Springer Optimization and Its Applications 58

My T. Thai
Panos M. Pardalos *Editors*

Handbook of Optimization in Complex Networks

Communication and Social Networks

Springer

Springer Optimization and Its Applications

VOLUME 58

Aims and Scope
Optimization has been expanding in all directions at an astonishing rate during the last few decades. New algorithmic and theoretical techniques have been developed, the diffusion into other disciplines has proceeded at a rapid pace, and our knowledge of all aspects of the field has grown even more profound. At the same time, one of the most striking trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field. Optimization has been a basic tool in all areas of applied mathematics, engineering, medicine, economics, and other sciences.

The series *Springer Optimization and Its Applications* publishes undergraduate and graduate textbooks, monographs and state-of-the-art expository work that focus on algorithms for solving optimization problems and also study applications involving such problems. Some of the topics covered include nonlinear optimization (convex and nonconvex), network flow problems, stochastic optimization, optimal control, discrete optimization, multi-objective programming, description of software packages, approximation techniques and heuristic approaches.

My T. Thai • Panos M. Pardalos
Editors

Handbook of Optimization in Complex Networks

Communication and Social Networks

Springer

Editors
My T. Thai
Department of Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611
USA
mythai@cise.ufl.edu

Panos M. Pardalos
Department of Industrial and Systems Engineering
University of Florida
303 Weil Hall
Gainesville, FL 32611
USA
pardalos@ufl.edu

Laboratory of Algorithms and Technologies for Networks Analysis (LATNA)
National Research University
Higher School of Economics 20
Myasnitskaya st. Moscow
101000
Russia

ISSN 1931-6828
ISBN 978-1-4614-0856-7 e-ISBN 978-1-4614-0857-4
DOI 10.1007/978-1-4614-0857-4
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2011938465

To our families!

Preface

The oldest, shortest words – "yes" and "no" – are those which require the most thought.

Pythagoras (Greek Philosopher 582 BC – 497 BC)

One common problem that spans several diverse applications is the management and derivation of knowledge from huge amounts of data, especially in scenarios involving human and social activities. In many practical situations, a real-life dataset can be represented as a large network (graph) – a structure that can be easily understood and visualized. Furthermore, special structures of graphs, when viewed in the context of a given application, provide insights into the internal structure and patterns of the data. Among the many examples of datasets that can be represented as graphs are the Web graph derived from the World Wide Web, the Call graph arising in telecommunications traffic data, and metabolic networks arising in biology. Of particular interest are social networks, in which vertices represent people or groups of people.

Although the concept of a network roots back to the ancient Greek philosopher Pythagoras in his theory of cosmos ($\kappa \acute{o} \sigma \mu o \varsigma$), the mathematical principles of networks were first developed in the last century. The first book in networks appeared in 1936 (D. König: Theory of Finite and Infinite Graphs). Since then, there has been a huge explosion of research regarding theoretical tools and algorithms in the analysis of networks.

One of the most exciting moments came at the dawn of the new Millennium in 1999 with the discovery of new types of graphs, called complex networks. Examples of such well-known classes of complex networks are scale-free networks and small-world networks. These classes of networks are characterized by specific structural features such as the power-law vertex degree distribution (scale-free networks) and for the short path lengths, small diameter and high clustering (small-world networks). Moreover, several other measures and features have been discovered, and are recently the focus of active research that related to the structural properties of complex networks. A new area of complex networks has been rapidly developing, spanning several disciplines such as mathematics, physics, computer science, social science, biology, and telecommunications.

In our two volume handbook, an attempt was made to present a wide spectrum of recent developments with emphasis in both theory and applications on complex networks. The first volume focuses on basic theory and properties of complex networks, on their structure and dynamics, and optimization algorithmic approaches. The last part of the volume concentrates on some feature applications. The second volume, this volume, deals with the emerging issues on communication networks and social networks. It covers material on vulnerability and robustness of complex networks. The second part is dedicated to complex communication networks, discussing several critical problems such as traffic activity graph analysis, throughput optimization, and traffic optimization. The last part of this volume focuses on recent research topics on online social networks such as security and privacy, social aware solutions, and people rank.

We would like to take this opportunity to thank all authors, the anonymous referees, and Springer for helping us to finalize this handbook. Our thanks also go to our students for their help during the processing of all contributions. We hope that this handbook will encourage research on the many intriguing open questions and applications on complex networks that still remain.

Gainesville, FL

My T. Thai
Panos M. Pardalos

Acknowledgements

M. T. Thai is partially supported by National Science Foundation (NSF) and Department of Defense grants.

P. M. Pardalos is partially supported by LATNA Laboratory, National Research University Higher School of Economics, RF government grant, ag. 11.G34.31.0057.

Contents

Part I Vulnerability and Robustness

1 Structural Vulnerability and Robustness in Complex Networks: Different Approaches and Relationships Between them 3
Regino Criado and Miguel Romance

2 Optimizing Network Topology for Cascade Resilience 37
Alexander Gutfraind

3 Optimizing Synchronization, Flow, and Robustness in Weighted Complex Networks 61
G. Korniss, R. Huang, S. Sreenivasan, and B.K. Szymanski

4 Joint Optimization of Resources and Routes for Minimum Resistance: From Communication Networks to Power Grids 97
Ali Tizghadam, Alireza Bigdeli, Alberto Leon-Garcia, and Hassan Naser

5 Clique Relaxation Models in Social Network Analysis 143
Jeffrey Pattillo, Nataly Youssef, and Sergiy Butenko

Part II Complex Communication Networks

6 Application Traffic Activity Graph Analysis 165
Yu Jin, Esam Sharafuddin, and Zhi-Li Zhang

7 Localized Bridging Centrality 197
Soumendra Nanda and David Kotz

8 On Throughput Maximization Problem for UWB-Based Sensor Networks via Reformulation–Linearization Technique 219
Yi Shi, Y. Thomas Hou, and Hanif D. Sherali

9 A Parallel Routing Algorithm for Traffic Optimization 241
M.L. Wang, K.H. Yeung, and F. Yan

10 Internet Based Service Networks 263
LiYing Cui, Soundar Kumara, and Réka Albert

Part III Online Social Networks and Security

11 On Detection of Community Structure in Dynamic Social Networks 307
Nam P. Nguyen, Ying Xuan, and My T. Thai

12 Path Formation in Human Contact Networks 349
Nishanth Sastry and Pan Hui

13 Social Forwarding in Mobile Opportunistic Networks: A Case of PeopleRank 387
Abderrahmen Mtibaa, Martin May, and Mostafa Ammar

14 Discount Targeting in Online Social Networks Using Backpressure-Based Learning 427
Srinivas Shakkottai and Lei Ying

15 Social-Aware Data Diffusion in Delay Tolerant MANETs 457
Yang Zhang, Wei Gao, Guohong Cao, Tom La Porta, Bhaskar Krishnamachari, and Arun Iyengar

16 Security and Privacy in Online Social Networks: Optimization Perspectives 483
Ling Ding, Hongjie Du, and Weili Wu

17 A Social Network Based Patching Scheme for Worm Containment in Cellular Networks 505
Zhichao Zhu, Guohong Cao, Sencun Zhu, Supranamaya Ranjan, and Antonio Nucci

Index 535

Part I
Vulnerability and Robustness

Chapter 1
Structural Vulnerability and Robustness in Complex Networks: Different Approaches and Relationships Between them

Regino Criado and Miguel Romance

Abstract The concept of vulnerability in the context of complex networks quantifies the capacity of a network to maintain its functional performance under random damages, malicious attacks, or malfunctions of any kind. Different types of networks and different applications suggest different approaches to the concept of networks structural vulnerability depending on the aspect we focus upon. In this introductory chapter, we discuss some different approaches and relationships amongst them.

1.1 Introduction

The study of complex networks has been found to be very productive in science and technology. Why? Because complex networks represent a natural alternative for representing, characterizing, and modeling the structure and non-linear dynamics of all discrete complex systems. In fact, many complex systems of the real world can be modeled using complex networks where nodes represent the different constituents of the system and edges depict the interactions between them. Different systems such as transport networks (underground, airline networks, road networks), communication networks (computer servers, internet), biochemical networks (metabolic, protein and genomic networks), social networks, infrastructure networks (electric power grids, water supply networks), and some others (including the World Wide Web) are known to have common characteristics in their behavior and structure [4,7,12,17,48,55,68,91,93,94,103,113]. Because of this reason they can be studied using non-linear mathematical models and computer modeling approaches.

R. Criado (✉) • M. Romance
Universidad Rey Juan Carlos, C/Tulipan s/n, 28933-Madrid, Spain
e-mail: regino.criado@urjc.es; miguel.romance@urjc.es

The study of structural properties of the underlying network may be very important in the understanding of the functions of a complex system as well as to quantify the strategic importance of a node (or set of nodes) in order to preserve the best functioning of the network as a whole. The improvements in computers performance in the last decades granted us the ability to analyze huge complex networks.

The concept of vulnerability in a network aims at quantifying the network's security and stability under the effects of all that kind of disfunctions. A series of different approaches from several branches of knowledge have been introduced to quantify the vulnerability of a complex network [1, 6, 11, 13, 15, 17, 18, 27, 35, 41, 67, 112].

Thus, for instance, in structural engineering the term "vulnerability" is often used to capture the susceptibility of a component or a system to some external action [2, 67]. In this way, a structure is vulnerable if any small damage produces disproportionately large consequences [2].

Several studies of critical infrastructure networks have focused on understanding the security of these networks and their susceptibility to damage, failures, attacks, disruptions, degradations or disfunctions of any type [3, 9, 28, 34, 40–42, 63, 64, 66, 74, 76–78, 89, 90, 106].

Another perspective is provided by the study of transportation systems. In this context, the vulnerability of a transportation system can be understood as the susceptibility to disruptions giving a considerable reduction in network serviceability as a result [13, 68]. Taylor and D'Este [68, 105] relate vulnerability to the degree of accessibility of a given node in the network, where accessibility is expressed as the travel cost needed to access the particular node, comparing optimal and alternative routes or detors.

A related concept which emerges in this context is the concept of reliability. It is important to remark, in any case, that vulnerability and reliability are two different concepts, not exactly opposite or complementary, since a measure of reliability is related to the concept of risk, which implies the use of a measure of the probability that guarantees the network will function under certain circumstances [13, 68, 104, 105]. So, reliability may thus be viewed as the degree of stability that a system offers certainly related to a measure of probability. Vulnerability means, in this sense, non-reliability or exhibiting a low degree of operability under certain circumstances.

In classic graph theory, the term "vulnerability" is related to a lack of resistance of the graph to the deletion of vertices and edges [11]. This point of view matches up with the structural vulnerability's approach, i.e., how the topology of a network is affected by the removal of a finite number of links and/or nodes. In the context of classic graph theory, the analysis of vulnerability is carried out by studying different versions of the connectivity of the graph [11, 65, 94]. The node-connectivity (edge-connectivity) is the smallest number of nodes (edges) whose removal disconnects the network (or in case of disconnected networks it increases the number of connected components). An alternative way to analyze connectivity is by considering the number of node-independent paths between two vertices or, in the same way, to the minimum number of other vertices in the network that must fail in order for those two vertices to become disconnected from one another [65,81,94].

On the other hand, complex networks analysis focuses on statistical graph measures, and numerical models (simulation), using a statistical approach to asses network vulnerability by measuring the fraction of the vertices or links to be removed before a complete disconnection happens in the network in order to study complex (large) networks. Specifically, many authors (see, for instance, [6, 13, 15–17, 27, 28, 33, 35, 41, 42, 66, 74, 76, 78]) have studied the structural vulnerability (so-called error and attack tolerance) of theoretical network models and empirical networks. So, structural vulnerability is related to the study of complex systems structure, represented as networks of multiple interconnected interacting parts and their susceptibility to damage by errors and attacks. This is how the term vulnerability is mainly used throughout this chapter.

Two measurements derived from the spectral analysis of the network connectivity may also be used to quantify the robustness and optimal connectivity of networks, independent from the network size. These measurements are algebraic connectivity and spectral gap. Algebraic connectivity was introduced in [53]. This measure, which depends on both the number of nodes and their respective configurations, indicates the level of connectivity in the graph [70, 85, 111]. So, the larger the algebraic connectivity is, the more difficult it is to cut the network into disconnected parts. Spectral gap is related to the so called "good expansion" properties [50]. The existence of good expansion properties (given by a sufficiently large value of spectral gap) together with uniform degree distribution result in higher structural sturdiness and robustness against node and link failures. On the contrary, low values of spectral gap indicate a lack of good expansion properties usually represented by bridges, cut vertices and network bottlenecks [50]. Both measurements, algebraic connectivity, and spectral gap, let us give alternative ways to quantify the network's well-connectedness.

The concept of vulnerability is also used to characterize a lack of robustness and resilience of a system. In everyday language, the word robustness is related to strength and sturdiness, but it is necessary to clarify, also in this case, that these concepts are not exactly the complement concepts of vulnerability. A system is robust if it will retain its system structure (function) intact (unchanged or nearly unchanged) when exposed to perturbations. The concept of resilience [17,18,55,91,94,100,102] is related to the capability of the system to regain a new stable position (close to it's original state) after perturbations.

In practice, real and different networks are vulnerable to many (external and internal) circumstances and events, so the task of finding a generally applicable measure of vulnerability is not an easy task. Nevertheless, in [39] the authors establish a new general framework, the family of (Ψ, p, q)-vulnerabilities, which comprises most of the (structural) vulnerability definitions appeared in complex network's literature and allows one to calculate relationships between different vulnerabilities. In any case, the purpose of this work is not to argue because of particular definitions of vulnerability or robustness, but rather to review the concept of structural vulnerability, that is, the concept of vulnerability based on the network's structure and topology [1, 2, 11, 76, 78, 79].

In general, under the perspective of structural vulnerability, two kinds of damages can be considered on error and attack tolerance in complex networks [6, 18, 27, 100, 102]: the removal of randomly chosen vertices (error tolerance) and the removal of deliberately chosen vertices (attack tolerance). Attacks are realized by removing vertices or edges in the order of decreasing importance, which can be quantified by properties such as degree (i.e. number of connected edges) of a node, connectivity, betweenness, etc. Depending on the concept of "importance" we consider, we will get different definitions of vulnerability. For example, to determine how important is a person within a social network, which is the appropriate criterion: to have many contacts or to have less but more important contacts? A specific way to measure the relative importance of a node within the network is by using the main measures of centrality: degree centrality, betweenness, closeness, and eigenvector centrality. A survey to review centrality measures as well as generalizations can be found in [73].

For instance, Latora and Marchiori in [76] studied the consequence and prevention of terrorism attacks in a given network, and suggested a method to detect critical vertices (i.e. the most important vertices for efficient network functioning). These authors show in [42] how the topology of a communication/transportation network may have important roles on error and attack tolerance of the system. In both cases, the importance of a particular vertex is given in terms of the change in network efficiency when this vertex is removed.

Under a general point of view, different types of networks and different applications suggest different approaches to the concept of networks (structural) vulnerability. In fact, by considering different ways of measuring the drop of performance of a network under malicious attacks or random damages, and depending on the nature of the problem, the pursued objective and the aspect we focus on, we can get different approaches. Some of these approaches, which depend on the concept of "importance" for vertices and edges we consider, are the following:

- Structural vulnerability related to the loss of connectivity (see, e.g., [46, 96]). This approach compares the vulnerability of networks of about the same size and structures by relating the concept of vulnerability to the loss of connectivity when we remove some nodes and edges to potentially disconnect the network. Under this point of view, the more homogeneous a network is (i.e., with all the nodes and links playing a similar role) the more robust that network is. An alternative approach, under this point of view, is given in [34–36].
- Structural vulnerability related to the variation of the network performance (see, e.g., [41, 66, 74, 76]). This approach relates the measure of vulnerability of a network to the fall of its efficiency when a damage occurs.
- Structural vulnerability related to betweenness [15, 16, 31, 33, 38, 96], another centrality measures and another concepts [17, 51, 84, 91, 97, 98, 106]. This approach attend to the strategic importance of specific links and nodes in order to preserve the functioning and performance of the network as a whole.
- Structural vulnerability based on spectral analysis [20, 21, 25, 32, 53, 82, 83, 85].

It's not worth to say that each one of these approaches has its advantages and disadvantages, and the most suitable approach for a specific problem may depend on the problem under investigation and the size of the network. As it is showed in [39] all of these approaches can be submerged in a general framework which give us a new perspective and formalism to the concept of network's vulnerability.

In short, this chapter is devoted to analyze different approaches to structural vulnerability, i.e., how different classes of network topologies are affected by the removal of a finite number of links and/or nodes.

The chapter is organized as follows: Sect. 16.2 introduces the basic concepts and notation to comprehensively analyze different approaches to the concept of vulnerability. Section 9.3 is devoted to establish an axiomatic support to the concept of vulnerability. In Sect. 9.4, some approaches to the concept of vulnerability are analyzed. Section 9.5 is devoted to introduce a common framework to several structural vulnerabilities. Finally, in Sect. 1.6, we collect some results about the numerical comparison between different types of network's structural vulnerability performed over random models and real life networks.

1.2 Basic Concepts and Notation

From a schematic point of view, a complex network is a mathematical object $G = (X,E)$ composed by a set of nodes or vertices $X = \{1,\dots,n\}$ that are pairwise joined by links or edges belonging to the set $E = \{\ell_1,\dots,\ell_m\}$. We consider the *adjacency matrix* $A(G) = (a_{ij})$ of $G = (X,E)$ determined by the conditions

$$a_{ij} = \begin{cases} 1 & if \quad \{i,j\} \in E \\ 0 & if \quad \{i,j\} \notin E. \end{cases}$$

Two vertices are called *adjacent* (or neighbors) if they are connected by an edge. The number of neighbors of a node i, denoted by d_i, is its *node degree*. Obviously, the degree of a node i can be easily calculated by the expression $\sum_{j=1}^{n} a_{ij}$. In the sequel, we will denote by $\delta(G)$ the minimum node degree of the nodes of G, and by $\Delta(G)$ the maximum node degree of the nodes of G. If all the nodes of G are pairwise adjacent, then G is called *complete* and a complete network of n nodes is denoted by K_n. K_3 is called a *triangle*. A network $G = (X,E)$ is called *q-partite* if X admits a partition into q classes such that every edge has its ends in different classes: nodes in the same partition class must not be adjacent. If $q = 2$, one usually says *bipartite*. A q-partite network in which every two nodes from different partition classes are adjacent is called complete. A *star* is a complete bipartite network such that one of the classes has exactly one element. A star of n nodes is denoted by S_n.

One we have introduced the concept of complex networks as the main object of the complex network analysis, we should give the basic parameters used in the literature in order to analyze these objects. There are plenty of mathematical functions that help to study and classify the behavior of complex network structure (see, e.g. [17] and [54]), including metric parameters, clustering, spectral functions and dynamical parameters among many others. In the rest of this chapter we will put mainly the stress in some metric and spectral functions that will help to analyze the structural vulnerability and robustness of a complex network. In this section, we will start by introducing the basic metric properties and parameters of networks while in Sect. 1.4.5 we will give the spectral functions used in the structural vulnerability analysis.

The metric structure of a complex network is related to the topological distance between nodes of the network, written in terms if walks and paths in the graph. A *walk* (of length k) in G is a non-empty alternating sequence

$$\{i_1, \ell_1, i_2, \ell_2, ..., \ell_{k-1}, i_k\}$$

of nodes and edges such that $\ell_r = i_r, i_r + 1$ for all $r < k$. If $i_1 = i_k$, the walk is *closed*. A *path* between two nodes is a walk through the network nodes in which each node is visited only once. A *cycle* is a closed walk that starts and ends at the same node, in which no edge is repeated. A cycle of n nodes is denoted by C_n. C_3 is a triangle. If it is possible to find a path between any pair of nodes the network is referred to as *connected*; otherwise, it is called disconnected. The *length* of a path is the number of edges of that path. If $i, j \in X$ a *geodesic* between i and j is a path of the shortest length that connects i and j. The *distance* d_{ij} between i and j is the length of a geodesic between i and j. The maximum distance $D(G)$ between any two vertices in G is called the *diameter* of G. By n_{ij} we will denote the number of different geodesics that join i and j. If $v \in X$ is a node and $\ell \in E$ is a link, then $n_{ij}(v)$ and $n_{ij}(\ell)$ will denote the number of geodesics that join i and j passing through v and ℓ respectively. A network $H = (Y, F)$ is a subnetwork of $G = (X, E)$ if $Y \subseteqq X$, $F \subseteqq Y$ and the edges in F connect nodes in X. A *connected component* is a maximal connected subgraph of G. Two paths connecting the same pair of vertices in a network are said to be vertex-independent if they share none of the same vertices other than their starting and ending vertices. A *k-component* is a maximal subset of the vertices of a network such that every vertex in the subset is connected to every other by k independent paths. For the special cases $k = 2$, $k = 3$, the k-components are called *bi-components* and *three-components* of the network. For any given network, the k-components are nested: every three-component is a subset of a bi-component, and so forth.

The *characteristic path length*, defined as

$$L(G) = \frac{1}{n(n-1)} \sum_{j=1}^{n} \sum_{\substack{k=1 \\ k \neq j}}^{n} d_{j,k} = \frac{1}{n(n-1)} \sum_{j \neq k \in X} d_{j,k},$$

is a way of measuring the performance of a network, but because of errors and attacks, networks can become disconnected. In fact, if the distance between two nodes is infinite, $L(G)$ becomes infinite. The concept of *efficiency*, introduced by

Latora and Marchiori in [74] is a well defined quantity also for non-connected networks. The efficiency of a network G is defined as

$$E(G) = \frac{1}{n(n-1)} \sum_{i,j \in X, i \neq j} \frac{1}{d_{ij}}.$$

The fall of the network's efficiency when a node fails is one of the main approaches to network's *structural vulnerability* as we will see in Sect. 9.4.

In [54], a panoramic view of the main existing measurements of complex networks can be found.

1.3 A Preliminary Axiomatic Approach to Structural Vulnerability

A first attempt to establish a mathematical axiomatic support to the somehow *"intuitive"* notion of vulnerability of a graph was given in [35]. The goal of the authors was to extract some intuitive properties which should be taken into account in every reasonable definition of vulnerability.

The invariance under isomorphisms is the first property which a vulnerability measure must fulfil. Otherwise, it would not make any sense the fact that the resulting value can depend on where the nodes are located, it must depend only on the edges that are present between them. Normalization (i.e., taking values within the unit interval $[0,1]$) may be another requirement: It seems reasonable that a measure of vulnerability can take values between 0 and 1. To do so in a reasonable way, we look at the most vulnerable graphs having values close to one while robust graphs have values close to zero. In fact, we would like the bounds to be attained at least asymptotically. In any case, note that we can always normalize any finite vulnerability measure in order to get a parameter contained in the $[0,1]$ interval.

Another requirement is the condition that vulnerability must be computable in polynomial time (necessary for practical reasons).

Once these basic conditions are established, the key property of a vulnerability measure is that it should never increase by adding edges. The rationale behind this assertion is that such an addition can only reinforce the structure of the network, because the worst situation we can face is the loss of that edge, either alone (in which case we are left with the original network) or in combination with other edges or nodes (which is as bad as losing those edges or nodes and having no extra edge anyway). Moreover, a new edge should generally strictly increase the robustness of a network although it is conceivable that, under special circumstances, adding a particular edge can have no impact on the vulnerability of the graph (a redundant edge, for instance).

Going into greater detail, it is natural to think that the complete graph must be the least vulnerable network for a given network size, since it cannot be strengthened in any way. Of course, this follows from the requirement that vulnerability must not increase as edges are added, but it deserves mentioning on its own. On the other hand, trees are relatively vulnerable, while a specific tree, the “star” network (which has a central node to which the rest is connected), should have the greatest vulnerability under attacks for a given size, since it exposes a clear weak spot whose failure makes the graph totally disconnected. Now, we can establish the following definition summarizing the above properties:

Definition 1.1. [35] Let $\mathscr{N}$ be the set of all possible networks with a finite number of vertices. A *vulnerability function* v is a function $v : \mathscr{N} \to [0, +\infty)$ verifying the following properties:

(i) *(Coherence)* v is invariant under isomorphisms of graphs.
(ii) *(Soundness)* $v(G') \geqslant v(G)$ if G is obtained from G' by adding edges.
(iii) *(Effective computation)* $v(G)$ is computable in polynomial time respect to the number of vertices of G.

The previous list of properties may be extended depending on the specific application, the aspect we focus on and the way of measuring the drop of performance of a network under malicious attacks or random damages we are studying.

1.4 Some Different Approaches to the Concept of Structural Vulnerability

As we have said in the introduction, depending on the kind of metrics and measurements which are considered to identify the importance of different vertices and edges inside the network we can get different approaches and definitions of structural vulnerability.

Some of these definitions are consistent with the underlying intuitive idea of vulnerability has been used in different contexts, but it is easy to check that in some cases they cannot distinguish networks that should have different vulnerabilities. For example, if we consider the cycle C_4 and the complete graph K_4, it is easy to check that both graphs have vulnerability zero in the sense considered in [76], but our intuition suggests that K_4 is more robust than C_4 (see Fig. 1.1).

In this section, we analyze several metrics and measurements usually employed in network’s literature. Above all we consider the measures related to connectivity, variation of network’s performance, variation of centrality measures such us betweenness and spectral measurements.

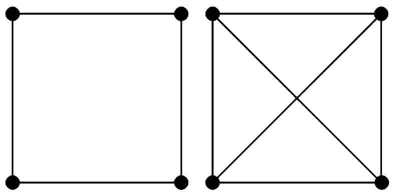


Fig. 1.1 The cycle C_4 and the complete graph K_4. The intuition suggests that K_4 is more robust than C_4, but some vulnerability measures do not distinguish between them

The measures related to connectivity are based on the analysis of network cohesion and adhesion and its responsiveness and tolerance to the removal of nodes and links. The measures related to betweenness and other centrality measures are based on the idea that the most important (in fact, critical) nodes and links for efficient network functioning are those which have the highest values of the network's centrality measure considered. Finally, the spectral measurements relates the topology of the network to the graph cohesion and connectivity strength through the study of the spectrum of the network's adjacency matrix.

1.4.1 Fall of Network Cohesion and Connectivity Type Vulnerabilities

In this approach, it is considered the impact of nodes and edges destruction in terms of potentially disconnecting the network.

There are two primary concepts of structural vulnerability based on connectivity in networks [46, 53]: The node connectivity and the edge connectivity, denoted, respectively, by $\kappa(G)$ and $\lambda(G)$. These two metrics are, respectively the minimal number of nodes (vertices together with adjacent edges) and the minimal number of edges whose removal disconnects the network. Both metrics are among basic most important indicators of network cohesion and robustness as they represent resistance to damage through quantifying the minimum number of failures/targeted attacks required to make the network disconnected.

The following well known theorem [46, 59, 81], due to Menger, give us an alternative formulation of node and edge connectivity:

Theorem 1.1. *For any network G we have*

(i) The node connectivity $\kappa(G)$ is the smallest number of node-distinct paths between any two nodes.

(ii) The edge connectivity $\lambda(G)$ is the smallest number of edge-distinct paths between any two nodes.

Another well known and useful theorem related to this metrics is the following [47] (recall that $\delta(G)$ is the minimum node degree of the nodes of G):

Theorem 1.2. *For any non-trivial network G the following inequalities between the node connectivity $\kappa(G)$, the edge connectivity $\lambda(G)$ and the minimum node degree $\delta(G)$ hold:*

$$\kappa(G) \leq \lambda(G) \leq \delta(G).$$

Dekker and Colbert in [46] define the concepts of node similarity and optimal connectivity in order to consider several strategies for designing optimally connected networks. They look at the degree of symmetry of the network as a way to analyze the network "robustness," in the sense that if a network has no distinguished nodes, an intentional attack may not cause important damages. This type of robustness is particularly interesting for military purposes and military networks of for civilian networks facing possibly terroristic activity. A network G is optimally connected if $\kappa(G) = \lambda(G) = \delta(G)$. The underlying idea is that a network is optimally connected if it is as robust as it could be, given the value of $\delta(G)$.

An alternative approach to this kind of robustness based on this idea of symmetry and regularity was given in [35], where the authors define two vulnerability functions V_1 and V_σ in order to quantify these features as follows:

$$V_1(G) = \exp\left(\frac{\Delta(G) - \delta(G)}{n} + n - m - 2 + \frac{2}{n}\right),$$

$$V_\sigma(G) = \exp\left(\frac{\sigma}{n} + n - m - 2 + \frac{2}{n}\right), \tag{1.1}$$

where σ is the standard deviation of the degree distribution, i.e.

$$\sigma = \left(\frac{1}{n}\sum_{i\in V}\left(d_i - \frac{2m}{n}\right)^2\right)^{1/2}. \tag{1.2}$$

Both definitions take into account the dispersion of the degree distribution and the number of nodes and links and satisfy the properties established in Definition 1.1. Moreover, a simple computation shows that the values of $V_1(G)$ for the complete graph K_n, the star S_n and the cycle C_n are, respectively,

$$\begin{aligned} V_1(K_n) &= \exp\left\{\frac{-n^3 + 3n^2 - 4n + 4}{2n}\right\}, \\ V_1(S_n) &= \exp\{0\} = 1, \\ V_1(C_n) &= \exp\left\{-2 + \frac{2}{n}\right\}. \end{aligned}$$

Note that $V_1(K_n)$ tends to zero and $V_1(C_n)$ tends to $\exp\{-2\}$ as n tends to infinity.

Nevertheless, $V_1(G)$ can be computed easily and gives a good estimation of the robustness of a complex network but it only cares about the nodes having extreme degrees, which makes the results not sharp enough. An example of this situation is shown in Fig. 1.2, where networks G and G' satisfies $V_1(G) = V_1(G')$ although our intuition suggests that G looks more robust than G'.

Hence, we will use the second vulnerability function, which gives better estimates for the security of a network.

It is easy to check that the values of $V_\sigma(G)$ for the complete graph K_n, the star S_n and the cycle C_n are, respectively,

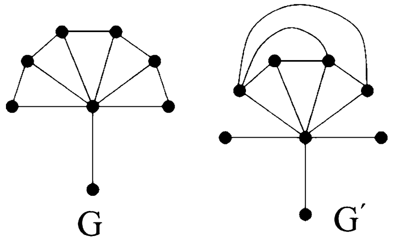


Fig. 1.2 G and G' have the same the same degree sequence (and therefore $V_1(G) = V_1(G')$), but the intuition suggests that G looks more robust than G'

$$V_\sigma(K_n) = \exp\left\{0 + n - |E| - 2 + \frac{2}{n}\right\}$$

$$V_\sigma(S_n) = \exp\left\{\frac{\left(\frac{(n-1)(n-2)^2}{n}\right)^{1/2}}{n} + n - (n-1) - 2 + \frac{2}{n}\right\}$$

$$V_\sigma(C_n) = \exp\left\{-2 + \frac{2}{n}\right\}.$$

Note that $V_\sigma(K_n)$ tends to zero and $V_\sigma(C_n)$ tends to $\exp\{-2\}$ as n tends to infinity.

Both measures $V_1(G)$ and $V_\sigma(G)$ provide values close to one for the most vulnerable graphs decreasing to values closer to zero for more robust graphs. Nevertheless, it seems that this type of vulnerability reasonably compare the vulnerability of networks possessing more or less the same size and structures, but it fails to properly rank vulnerability when networks of different sizes and structures are compared [114]. In any case, in [36], an extension of $V_\sigma(G)$ is given for directed networks and this measure is used to perform a comparative analysis of those performance measures over a significant sample of subway networks worldwide.

1.4.2 Fall of Efficiency Type Vulnerabilities

The concept of efficiency in a network plays the role of measuring its ability for the exchange of information and its response for the spread of perturbations in diverse applications [74, 75, 77, 78]. In fact, the efficiency is an indicator of the network performance, i.e., of its capability to have a short-path connection among nodes. The study of efficiency of a network is not only interesting in computer and communication networks but also in many other examples of complex networks, since it measures how optimally the dynamics of the network takes place and how its behavior can change due to some variations in the topology of the network. For example, it is crucial to quantify the stability of a cellular network when it is subject to random errors as a result of mutations, harsh extremal conditions that eliminate metabolites or protein misfolding [71], as in trophic networks it is important to analyze the response of the network to the removal, inclusion or mutation of species in an ecosystem. In [41, 76], by using as mathematical measures the global and the local efficiency, the authors study the effects of errors and attacks both on the global and the local properties of the network, showing that global efficiency is a better measure than the characteristic path length to describe the response of complex networks to external factors.

One of the main approaches to the vulnerability of networks relates this concept to the fall of networks efficiency when the failure of node happens [74, 77, 78]. Latora and Marchiori measure the vulnerability of a node as relative drop in performance (efficiency) after removal of the i-th node together with all the edges connected to it. Following this idea, the vulnerability of a network $G = (X, E)$ to a failure of a single node can be defined by several aggregation techniques (see [78]), mainly as it is shown in the following expressions:

$$V_{\max}(G) = \max\{E(G) - E(G \setminus \{v\});\ \ v \in X\}, \tag{1.3}$$

or

$$\overline{V}(G) = \frac{1}{n}\sum_{v\in X}\big(E(G) - E(G\setminus\{v\})\big), \tag{1.4}$$

where $E(G)$ denotes the efficiency of G [74] defined by

$$E(G) = \frac{1}{n(n-1)}\sum_{i\neq j\in X}\frac{1}{d_{ij}} \tag{1.5}$$

and $G \setminus \{v\}$ is the network G without node v and its incident links.

Obviously, the first one is suitable for intentional attacks and the second one for random failures or breakdowns. To normalize these expressions, i.e., to take values within the interval $[0, 1]$, it is enough to divide by the total efficiency $E(G)$. So, we can get the normalized $V_{\max_n}(G)$ and the normalized $\overline{V_n}(G)$ through the expressions:

$$V_{\max_n}(G) = \frac{V_{\max}(G)}{E(G)}, \tag{1.6}$$

and

$$\overline{V_n}(G) = \frac{\overline{V}(G)}{E(G)}. \tag{1.7}$$

If D is now a class of damages (for e.g., the deactivation of a node, the disfunction of a link, or a cascade failure of two or more nodes), and we consider a specific damage $d \in D$, the fall of efficiency of a network $G = (X, E)$ due to the presence of the damage d is given by

$$V(G, d) = E(G) - E(d(G)), \tag{1.8}$$

where $E(\cdot)$ is the efficiency and $d(G)$ is the graph obtained from G when the damage d is applied.

For example, if we consider the class of damages D of all possible single-node failures, we can identify each damage $d \in D$ with a node $i \in X$, and therefore we denote

$$V(G,i) = E(G) - E(G \setminus \{i\}). \tag{1.9}$$

Note that we will use a similar notation for the classes of damages of multiple-node failures, which can be identified with a certain family of subsets of X in such a way that, if the damage is the failure of the subset of nodes $I_k = \{1,\dots,k\}$, then the fall of efficiency $E(G) - E(G \setminus I_k)$ will be denoted by $V(G,I_k)$.

For a class of damages D, the vulnerability of a network $G = (V,E)$ to the class D can be defined by several aggregation techniques as

$$V_{\max}(G,D) = \max\{V(G,d);\ \ d \in D\} \tag{1.10}$$

or

$$\overline{V}(G,D) = \frac{1}{|D|} \sum_{d \in D} V(G,d), \tag{1.11}$$

where $|D|$ is the number of damages of the class D. Note that $V_{\max}(G,D)$ measures the maximal damage that can occur to the network G while $\overline{V}(G,D)$ is the average fall of efficiency of G under a damage of the class D. While its meaning is evident, such a definition of vulnerability as the fall of efficiency has several inconveniences, as it was stated in [16].

If we now consider the damage d that eliminates a single node $i_0 \in V$, the definition of vulnerability as the fall of efficiency presents several difficulties (a similar situation also occurs when we are dealing with multiple-node failures) as we will see now.

The fall of efficiency $E(G) - E(G \setminus \{i_0\})$ can be expressed as

$$\frac{1}{N(N-1)} \sum_{i \neq j \in V} \frac{1}{d_{ij}} - \frac{1}{(N-1)(N-2)} \sum_{\substack{i,j \in V \\ i,j \neq i_0}} \frac{1}{d'_{ij}},$$

where d'_{ij} is the distance in $G \setminus \{i_0\}$. If we take $i,j \in V,\ i,j \neq i_0$, such that the distance between i and j does not change when i_0 fails, we have that

$$\frac{1}{N(N-1)} \frac{1}{d_{ij}} - \frac{1}{(N-1)(N-2)} \frac{1}{d'_{ij}} = -\frac{2}{N(N-1)(N-2)} \frac{1}{d_{ij}} < 0.$$

Hence, the failure of i_0 produces a *collateral effect* in the fall of the efficiency even if this damage does not affect the distance between i and j. This problem comes from the fact that this difference of efficiency is *size-dependent*, and it has other inconveniences that should be avoided to give a smart parameter that measures the intuitive idea of vulnerability. To overcome this difficulty, a new measure of vulnerability is proposed in [16] that is not size-dependent and does not produce collateral effects. If we take the class of damages D of all possible single-node failures, and consider the damage of deactivating the node $i_0 \in V$. We then consider the square matrix E of size $N-1$ whose (i,j) entry $(i,j \neq i_0)$ is

$$E_{ij} = \begin{cases} \dfrac{1}{d_{ij}} - \dfrac{1}{d'_{ij}}, & \text{if } i \neq j \\ 0, & \text{if } i = j, \end{cases}$$

where d'_{ij} is the distance in $G \setminus \{i_0\}$.

This idea leads to the definition

$$W(G,i_0) = \sum_{\substack{i \neq j \in V \\ i,j \neq i_0}} \left(\frac{1}{d_{ij}} - \frac{1}{d'_{ij}} \right). \tag{1.12}$$

So, by using this idea the following definition to the vulnerability of the network $G = (X,E)$ under a class of damages D is given in [16] as

$$W_{\max}(G,D) = \max\left\{W(G,i_0);\ \ i_0 \in D\right\}, \tag{1.13}$$

or

$$\overline{W}(G,D) = \frac{1}{|D|} \sum_{i_0 \in D} W(G,i_0), \tag{1.14}$$

inspired by (1.10) and (1.11), respectively.

The following result is also obtained in [16]:

Theorem 1.3. *If $G = (X,E)$ is a complex network with $n \geq 3$ nodes, then*

$$\sum_{i_0 \in X} V(G,i_0) = \frac{1}{(n-1)(n-2)} \sum_{i_0 \in X} W(G,i_0), \tag{1.15}$$

where the damage i_0 is the failure of node i_0.

To finish this sub-section devoted to fall of efficiency type vulnerabilities, it is important to remark that Goldshtein et al. [63] introduce an additional parameter called the relative variance h. This parameter is a measure of the fluctuation level and it is used to describe the hierarchical properties of the network, and thus its vulnerability. In other words, they suggest that the ordered distribution of vertices with respect to their vulnerability is related to the network hierarchy; thus, the most vulnerable (critical) vertex occupies the highest position in the network hierarchy.

1.4.3 Betweenness Centrality (Node and Link) Based Vulnerabilities

Another approach to network's vulnerability is based on the idea that critical nodes and links stand between others, playing the role of an intermediary in the interactions. So, the greater the number of paths in which a node or edge participates, the higher the importance of this node or edge for the network. Thus, assuming that the interactions follow the shortest paths between two vertices, it is possible to quantify the importance of a node or an edge in terms of its betweenness centrality.

Node betweenness was first proposed by Freeman [56] in 1977 in the context of social networks. This concept has been considered more recently as an important parameter in the study of networks associated to complex systems [91]. Girvan and Newman [60] generalize this definition to edges and introduce the edge betweenness of an edge as the fraction of shortest paths between pairs of vertices that run along it.

Specifically, the betweenness approach to network's vulnerability is based on the concentration of the geodesic structure throughout the network.

The node betweenness centrality $B(G)$ of a network G [56] is

$$B(G) = \left(\frac{1}{n} \sum_{v \in X} b_v \right),$$

where b_v is the betweenness of the node $v \in X$ (see, e.g., [95, 109]) given by

$$b_v = \frac{1}{n(n-1)} \sum_{i,j \in X i \neq j} \frac{n_{ij}(v)}{n_{ij}}.$$

(Recall that n_{ij} is the number of different geodesics that join i and j, and $n_{ij}(v)$ is the number of geodesics that join i and j passing through v).

The maximum betweenness of the network G is

$$B_{\max}(G) = \max\{b_v : v \in X\}$$

The same parameters can be defined for edges exactly in the same way as before, obtaining the edge-betweenness $B_{\mathrm{E}}(G)$

$$B_{\mathrm{E}}(G) = \left(\frac{1}{m} \sum_{\ell \in E} b_\ell \right)$$

where, in the same way as before, b_ℓ is the betweenness of the link $\ell \in E$ given by

$$b_\ell = \frac{1}{n(n-1)} \sum_{i,j \in X i \neq j} \frac{n_{ij}(\ell)}{n_{ij}}$$

and the maximum edge betweenness centrality

$$B_{\mathrm{Emax}}(G) = \max\{b_\ell : \ell \in E\}.$$

In Sect. 1.4.5, some relationships between this measures and the eigenvalues of the Laplacian matrix of G will be shown.

In [15], it was introduced the link-based multi-scale vulnerability of a complex network $G = (X, E)$ as

$$V_{E,q}(G) = \left(\frac{1}{m} \sum_{\ell \in E} b_\ell^q \right)^{1/q},$$

for any $q \in [1, +\infty)$, where b_ℓ is the betweenness of the link $\ell \in E$ (see, e.g. [95,109]) given by

$$b_\ell = \frac{1}{n(n-1)} \sum_{li,j \in X i \neq j} \frac{n_{ij}(\ell)}{n_{ij}}.$$

A remarkable relationship between the characteristic path length $L(G)$ and $B_{\mathrm{E}}(G) = V_{E,1}(G)$ (in fact, the edge betweeness of $G = (X, E)$), as it is shown in [15] is the following:

$$\begin{aligned} V_{E,1}(G) &= \frac{1}{|E|} \sum_{\ell \in E} b_\ell = \frac{1}{|E|} \sum_{\ell \in E} \left(\sum_{j,k \in X} \frac{n_{jk}(\ell)}{n_{jk}} \right) \\ &= \frac{1}{|E|} \sum_{j,k \in X} \frac{1}{n_{jk}} \left(\sum_{\ell \in E} n_{jk}(\ell) \right). \end{aligned}$$

Notice that if $\mathscr{P}_{jk}$ is the set of all geodesics joining j and k then one has

$$n_{jk}(\ell) = \sum_{g \in \mathscr{P}_{jk}} \chi_g(\ell),$$

where $\chi_g(\ell)$ is 1 if ℓ belongs to the geodesic g and 0 otherwise. Hence if $d_{j,k}$ denotes the distance between j and k in the network then

$$\begin{aligned} V_{E,1}(G) &= \frac{1}{|E|} \sum_{j,k \in X} \frac{1}{n_{jk}} \left(\sum_{\ell \in E} n_{jk}(\ell) \right) \\ &= \frac{1}{|E|} \sum_{j,k \in X} \frac{1}{n_{jk}} \left(\sum_{g \in \mathscr{P}_{jk}} \sum_{\ell \in E} \chi_g(\ell) \right) \\ &= \frac{1}{|E|} \sum_{j,k \in X} \frac{1}{n_{jk}} \left(\sum_{g \in \mathscr{P}_{jk}} d_{j,k} \right) = \frac{n(n-1)}{|E|} L(G) \end{aligned}$$

and therefore $B_{\mathrm{E}}(G) = V_{E,1}(G)$ measures essentially the same global properties than the characteristic path length $L(G)$.

It is possible to overcome such a limitation by introducing (see [15]) the coefficient

$$V_{E,p}(G) = \left(\frac{1}{|E|} \sum_{\ell \in E} b_\ell^p \right)^{1/|p|}, \tag{1.16}$$

for each value of $p > 0$. Such a coefficient gives a *multi-scale measure* of the vulnerability of a graph in the following sense: if one wants to distinguish between two networks G and G', one first compute $V_{E,1}(G)$. If $V_{E,1}(G) < V_{E,1}(G')$ then G is more robust that G'. On the other hand, if $V_{E,1}(G) = V_{E,1}(G')$, then one takes $p > 1$ and compute $V_{E,p}(G)$ until $V_{E,p}(G) \neq V_{E,p}(G')$.

If, in the previous reasoning, we replace edges by nodes, one can obtain another concept of vulnerability related to what is written above [38]. In this case, it was considered the node-based multi-scale vulnerability of a complex network given for any $q \in [1, +\infty)$ as,

$$\begin{aligned} V_{X,q}(G) &= \left(\frac{1}{n} \sum_{v \in X} b_v^q \right)^{1/q} \\ &= \left(\frac{1}{n} \sum_{v \in X} \left[\frac{1}{n(n-1)} \sum_{\substack{i,j \in X \\ i \neq j}} \frac{n_{ij}(v)}{n_{ij}} \right]^q \right)^{\frac{1}{q}}. \end{aligned}$$

In [38], it was proved that there an analytical connection between the node-based and link-based approach of structural vulnerability as the following result shows:

Theorem 1.4 ([38]). *Let $G = (X, E)$ be a network with n nodes and m links. If $1 < p < \infty$, then*

$$2^{\frac{1}{p}-1} \left(\frac{m}{n} \right)^{1/p} V_{E,p}(G) \le V_{X,p}(G),$$

$$V_{X,p}(G) \le 2^{\frac{1}{p}-1} \left(\frac{m}{n} \right)^{1/p} (gr_{\max})^{1-\frac{1}{p}} V_{E,p}(G) + \frac{1}{n},$$

where $gr_{\max}$ denotes the maximal degree of the network G.

This analytical result is sharp and in [38] it was illustrated this relationship in the Erdős–Rénji model of random networks, as we will show in Sect. 1.6.

1.4.4 Bottleneck Type Vulnerabilities

This approach has been studied in [31, 33], and in [50]. In the first two cases the problem of locating a leader node on a complex network was considered. This problem is interesting due to its practical applications, such as the key transfer protocol design for multi-party key establishment (see [22]), where it is stated that it is not always adequate to consider all nodes in the network equally important, for instance to distribute a message from a central server to a group of users or key distribution for private communication. So, it is needed to require secure communication between the leader (or the *initiator*) and all other members in order to initially establish key. In any case, the keys (or a part) must be sent through a communication network which holds up the flow of information. In [50], the study of network robustness based on the removal of bottleneck nodes and edges is considered in order to analyze the property of certain complex networks that are simultaneously sparse and highly connected (established as "good expansion" (GE)).

The criterium considered in [31] and [33] for choosing such a leader in a complex network is related to spotting the node v_o that minimizes the expected number of disconnected nodes from v_o under a random breakdown of a node other than v_o. In order to compute these values the concept of bottleneck was introduced. If we take three nodes $x, y, z \in X$, we say that y is a *bottleneck from x to z* if every path from x to z goes through y. Note that if we denote by $\Pi(x,z)$ the set of all bottlenecks from x to z, then it was proved in [33] that the expected number of disconnected nodes from v under a random breakdown of a node other than v is exactly

$$D(v) = \frac{1}{n-1} \sum_{v \neq w \in X} \Big(|\Pi(v,w)| - 1 \Big),$$

where $|\Pi(v_o,w)|$ is the number of bottleneck from v_o to w and n is the number of nodes of the complex network. It is clear that this function $D(\cdot)$ gives a criterium for measuring the vulnerability, since the lower the average value of $D(\cdot)$ is the higher robustness of the network we get. Therefore, the bottleneck vulnerability of a complex network $G = (X,E)$ of n nodes is defined as

$$\begin{aligned} B(G) &= \frac{1}{n} \sum_{v \in X} D(v) \\ &= \frac{1}{n(n-1)} \sum_{v \in X} \sum_{v \neq w \in X} \Big(|\Pi(v,w)| - 1 \Big). \end{aligned} \tag{1.17}$$

This method introduced in [31] shows that for a given tree with an invulnerable node (leader) the D-measures on the different nodes help us to choose an optimal location for the leader. Another related problem is the following: assume that we have two possible tree networks for a given number of nodes and links, each with its distinguished node. Which one is preferable from the point of view of robustness? The following example where G and G' are two tree-shaped graphs illustrates the situation (Fig. 1.3).

Since G and G' have the same number of nodes and their corresponding incidence degrees coincide, there is no hope that any vulnerability function whose definition is based on those two parameters might be of use when it comes to preferring G over

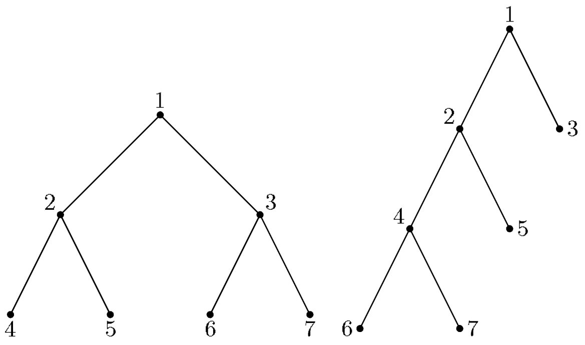


Fig. 1.3 An example of bottleneck-type vulnerability of two trees G (*on the left*) and G' (*on the right*) with a leader node. By using this $D(\cdot)$ we can select the best network from the point of view of robustness

G', or conversely. In contrast, the D-measures prove useful to our choice. In fact, G' should be preferred over G as the next table shows:

$$\begin{array}{llll} D^G(1)=10, & D^{G'}(1)=12, & D^G(5)=16, & D^{G'}(5)=14,\\ D^G(2)=11, & D^{G'}(2)=9, & D^G(6)=16, & D^{G'}(6)=15,\\ D^G(3)=11, & D^{G'}(3)=17, & D^G(7)=160, & D^{G'}(7)=15.\\ D^G(4)=16, & D^{G'}(4)=10, & & \end{array}$$

Note that the algorithm above yields node 1 as the best possible leader for G and node 2 as the best one for G'. Hence, selecting G' and node 2 in G' as the leader would be the wisest option since node 2 has the lowest D_1-vulnerability not only among the nodes of G' but among all nodes.

The concept of bottleneck and bottleneck vulnerability $B(\cdot)$ are very useful when we are dealing with complex networks which are tree-shaped (see [31]), but is it very restrictive when we consider general networks (see [33]) since the set $\Pi(i,j)$ is usually trivial. This is due to the fact that a node v is a bottleneck from i to j if all the possible paths from i to j go through v, which is too strong. In order to avoid this inconvenience in a general network, it is possible to consider a relative concept which gives a qualitative perspective of the geodesic structure of the complex network as follows [39]:

Definition 1.2. A node y is a geodesic bottleneck from node x to node z if every geodesic from x to z necessarily goes through y. We denote by $\Pi_g(x,z)$ the set of all geodesic bottlenecks from x to z.

If $v \in X$ and we denote by $D_g(v)$ the expected number of nodes that change their distance (inside the graph) to node v when a failure at some other node occurs, then we can prove that $D_g(v)$ is related to the geodesic bottlenecks from v, as the following result shows.

Proposition 1.1. *If $G=(X,E)$ is a complex network with n nodes and $v \in X$, then*

$$D_g(v) = \frac{1}{n-1}\sum_{z\in X}\Big(\big|\Pi_g(v,z)\big| - 1\Big).$$

Proof. First, note that if we fix $v \in X$, then the number of nodes that change their distance to node v when a failure in a node $y \neq v$ occurs is the number of nodes z such that $y \in \Pi_{\mathrm{g}}(v,z)$ and hence

$$\begin{aligned} D_{\mathrm{g}}(v) &= \frac{1}{n-1} \sum_{y \neq v} \left| \{ z \in X, y \in \Pi_{\mathrm{g}}(v,z) \} \right| \\ &= \frac{1}{n-1} \sum_{y \neq v} \sum_{z \in X} \chi_{\Pi_{\mathrm{g}}(v,z)}(y) \\ &= \frac{1}{n-1} \sum_{z \in X} \sum_{y \neq v} \chi_{\Pi_{\mathrm{g}}(v,z)}(y) \\ &= \frac{1}{n-1} \sum_{z \in X} \Big(|\Pi_{\mathrm{g}}(v,z)| - 1 \Big). \end{aligned}$$

By using this concept, the geodesic bottleneck vulnerability of a complex network $G = (X,E)$ of n nodes is introduced in [39] as

$$B_{\mathrm{g}}(G) = \frac{1}{n} \sum_{v \in X} D_{\mathrm{g}}(v) = \frac{1}{n(n-1)} \sum_{v \in X} \sum_{v \neq w \in X} \Big(|\Pi_g(v,w)| - 1 \Big). \tag{1.18}$$

It was proved in [33] that if $G = (X,E)$ is a tree then $|\Pi_g(r,z)| - 1$ is the distance between the nodes r and z, since there is a unique simple path from r to z and every node in that path is a bottleneck. Therefore, the last formula extends the results obtained in [31], but it is straightforward to prove that this phenomenon does not necessarily occur when $G = (X,E)$ is not a tree. Actually, given any node in G, it is possible to construct an auxiliary tree in G (the *bottleneck tree*) such that measuring distances in that tree gives us the same information about disconnected nodes in the original network (see [33]). The key-point to show this is the fact that the nodes in any bottleneck set $\Pi_g(x,y)$ are linearly ordered; in fact, every simple path from x to y runs through the nodes in $\Pi_g(x,y)$ in the same order. Thanks to this ordering, it is possible to define a *direct bottleneck of a node* for a given root [33]. If we have two different nodes r and $x \in X$, the last node in $\Pi_g(r,x)$, excluding x itself, will be called the direct bottleneck of x from r, and will be denoted by $\pi_r(x)$. Note that given distinct r, $x \in X$, we always have r, $x \in \Pi_g(r,x)$, so this definition always makes sense. The link that will help us establish a connection between the case of a general network and that of a tree-shaped one is the so called *bottleneck tree*. Given a node $r \in X$, the bottleneck tree of G rooted at r will be another network BT_r with the same set of nodes and exactly those edges of the form $(\pi_r(x),x)$, where $x \neq r$.

For example, if we consider the graph N_9, Fig. 1.4 shows the bottleneck tree rooted at 1.

The main reason why this bottleneck tree is useful in helping locate the right place for the leader is that measuring the distance from r to any other node is exactly

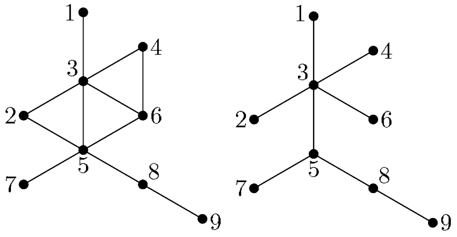


Fig. 1.4 A bottleneck tree for the graph N_9 (*on the left*) and its Bottleneck tree rooted at 1 (*on the right*)

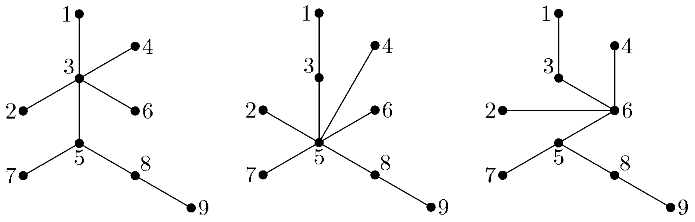


Fig. 1.5 All the bottleneck trees for the graph N_9, rooted at 1 or 3 (*on the left*); rooted at 5, 7, 8, or 9 (*on the center*); and rooted at 2,4, or 6 (*on the right*)

the same as counting how many bottlenecks there are in $\Pi_g(r,x)$. In other words, we have that

$$D(r) = \frac{1}{n-1}\sum_{x\in X} d_{rx}^{BT_r}, \qquad (1.19)$$

where $d_{rx}^{BT_r}$ is the distance from r to x in the bottleneck tree for G rooted at r. If we consider a non tree-shaped network, different leaders yield different bottleneck trees. For example, if we consider again the graph N_9 we can obtain essentially tree different bottleneck trees depending on which root we choose (see Fig. 1.5).

To end this subsection it is important to remark that there exist a strong relationship between the concept of bottleneck in a network G and the bi-components and k-components of G (see [96]).

1.4.5 Vulnerability and Algebraic Connectivity

The use of spectral methods in networks and graph theory have a long tradition. For example, the eigenvector-like centralities were introduced in sociology to measure the influence of each actor in a social group, taking into account the immediate effects, the mediative effects and the global effects of the social interaction [20], but they are also useful in other applications such as the web search engines like Google [23].

Specifically, spectral graph theory studies the eigenvalues of matrices that embody the graph structure. One of the main objectives in spectral graph theory is to deduce structural characteristics of a graph from such eigenvalue spectra.

Particularly, these methods are used in the study of the measures of vulnerability based on the fall of connectivity. In addition, spectral analysis has a lot of applications too numerous to be collected here. For example, spectral analysis allows characterizing models of real networks [80, 99], determine communities [92], to find the edges which connect different communities and remove them in a iterative form, breaking the network into disconnected groups of nodes [8,60] and even visualizing networks [101].

To introduce the eigenvalue spectra of a network some additional concepts are needed. The characteristic polynomial $\det(xI - A(G))$ of the adjacency matrix $A(G)$ is called the characteristic polynomial of G. The eigenvalues of $A(G)$ (i.e., the zeros of $\det(xI - A(G))$ are also called the eigenvalues of G. If μ is an eigenvalue of G, then a non-zero vector $\overrightarrow{v} \in \mathbb{R}^n$ satisfying $A(G)\overrightarrow{v} = \mu\overrightarrow{v}$, is called an eigenvector of $A(G)$ for μ; it is also called a μ-eigenvector. The relation $A(G)\overrightarrow{v} = \mu\overrightarrow{v}$ can be interpreted in the following way: if $\overrightarrow{v} = (v_1, ..., v_n)^t$, then for any vertex i we have that $\mu v_i = \sum_{j\sim i} v_j$, were the summation is over all neighbors j of i. If μ is an eigenvalue of G, then the set $\{\overrightarrow{v} \in \mathbb{R}^n : A(G)\overrightarrow{v} = \mu\overrightarrow{v}\}$ is a linear subspace of $\mathbb{R}^n$, called the eigenspace of G for μ.

On the other hand, since $A(G)$ is a real symmetric matrix, it is important to point out that all the eigenvalues of G are real. Moreover, $\mathbb{R}^n$ has a basis $\overrightarrow{v_1}, \dots, \overrightarrow{v_n}$ of n normal eigenvectors of $A(G)$ [25]. The eigenvalues $\mu_1(G) \leq \mu_2(G) \leq \cdots \leq \mu_n(G)$ of $A(G)$ are called the "spectrum" of G.

There is a large literature on algebraic aspects of spectral graph theory (see, e.g.,[14, 25, 43–45, 58, 61, 69, 70, 87, 88, 107]). The eigenvalue spectra of a network provide valuable information about the behavior of many dynamical processes running within the network, but in this section we only consider the applications of spectral analysis to static networks. For example, in [44] it is shown that diameter $D(G)$ of a network satisfies $D(G) \leq r - 1$, where r is the number of distinct eigenvalues.

The largest eigenvalue of the adjacency matrix $\mu_n(G)$ is called spectral radius of G. This eigenvalue is usually denoted by $\rho(G)$. It is important to remark that for $\rho(G)$, since $A(G)$ is non-negative, there exists an eigenvector whose all entries are non-negative.

This eigenvalue of $A(G)$ has received the most attention in this context, since this quantity refers to the speed of growth of walks in the graph (the number of walks of length k is, approximately, $\rho(G)^k$) [25]. A nice and useful property is given by the following inequality [44]:

$$\sqrt{\Delta(G)} \leq \mu_n(G) = \rho(G) \leq \Delta(G),$$

where $\Delta(G)$ is the maximum node degree of G. It is important to highlight in this subsection that the spectral radius of G plays an important role in modeling virus propagation in computer networks. The smaller the largest eigenvalue, the larger the robustness of a network against the spread of viruses. In fact, the epidemic threshold in spreading viruses is proportional to $\frac{1}{\rho(G)}$ [108]. Another example that remarks the importance of $\rho(G)$ is given by the Bonacich centrality of G (measure based on the eigenvectors associated to the spectral radius of G) [20, 21, 32].

The difference $s(G) = \rho(G) - \mu_{n-1}(G)$ between the spectral radius of G and the second eigenvalue of the adjacency matrix $A(G)$ is called the "spectral gap" of G [50]. A small value of $s(G)$ is usually observed through simultaneous sparseness and low connectivity, and the presence of bottlenecks and bridges whose removal cut the network into disconnected parts.

The spectral density of a network G is defined as

$$\rho(x) = \frac{1}{n}\sum_{j=1}^{n} \delta(x - \mu_j)$$

where

$$\delta(x) = \begin{cases} 1 & if x = 0 \\ 0 & if x \neq 0. \end{cases}$$

is the Kronecker or delta function.

The spectral density is one of most relevant tools used for studying the spectra of large complex networks [26, 58].

The Laplacian matrix of G is an $n \times n$ matrix $L(G) = D(G) - A(G)$, where $D(G) = \mathrm{diag}(d_i)$ and d_i denotes the degree of the node i. The matrix $L(G)$ is positive semi-definite, i.e., $\overrightarrow{v}^t \cdot L(G) \cdot \overrightarrow{v} \geq 0$ for any vector $\overrightarrow{v}$, and therefore its eigenvalues are non-negative. The least eigenvalue is always equal to 0 (note that $(1, 1, \ldots, 1)^t$ is an eigenvector corresponding to that eigenvalue); the second least eigenvalue is also called the algebraic connectivity of G. The eigenvalues of $L(G)$ are called the Laplacian eigenvalues of G. The Laplacian eigenvalues $\lambda_1(G) = 0 \leq \lambda_2(G) \leq \cdots \leq \lambda_n(G)$ are all real and nonnegative [85].

Many dynamical network processes (like synchronization) can be determined by the study of their Laplacian eigenvalues [58]. Furthermore, these eigenvalues are related to many basic topological invariants of networks such as diameter, betweenness centrality or mean distance. For example, the betweenness centrality $B(G)$ is closely related with the characteristic path length $L(G)$ as $B(G) = (n-1) \cdot (L(G) - 1)$ [29, 58].

The second smallest Laplacian eigenvalue $\lambda_2(G)$ is one of the most broadly extended measures of connectivity (see, for instance, [25, 44, 70, 72, 82, 83]). Larger values of algebraic connectivity represent higher robustness against efforts to disconnect the network, so the larger the algebraic connectivity is, the more difficult it is to cut a graph into independent components. In fact, the algebraic connectivity is only equal to zero if G is disconnected, and the multiplicity of zero as an eigenvalue of $L(G)$ is equal to the number of disconnected components of G. In [53] Fiedler proved the following upper bound on the algebraic connectivity

$$0 \leq \lambda_2(G) \leq \frac{n}{(n-1)}\delta(G).$$

Also, the following inequality can be found in [53]:

$$\lambda_2(G) \leq \kappa(G).$$

Hence, from Theorem 1.2 we can get that $\delta(G)$ and $\lambda(G)$ are also upper bounds of $\lambda_2(G)$.

However, these upper bounds in terms of the node and the link connectivity provides worst case robustness to node and link failures [53] so, as mentioned in [19], there are infinitely many graphs for which the algebraic connectivity is not a sharp lower bound.

In [69], the behavior of algebraic connectivity $\lambda_2(G)$ in the Erdös–Rénji random graph model is studied, obtaining that the algebraic connectivity increases with the increasing node and link connectivity in this model, showing that the larger the value of the algebraic connectivity $\lambda_2(G)$, the better the graph's robustness to node and link failures. Extensive simulations presented in this work show that the node and the link connectivity converge to a distribution identical to that of the minimal nodal degree, making $\delta(G)$ a valuable estimate of the number of nodes or links whose deflection results into disconnected random graph.

Mohar in [86] gave a bound that relates the algebraic connectivity $\lambda_2(G)$ to the diameter $D(G)$:

$$D(G) \geq \frac{4}{n\lambda_2(G)}.$$

Some other relationships between the inverse of algebraic connectivity $\lambda_2(G)$ and the ratio $\frac{\lambda_n(G)}{\lambda_2(G)}$ with other topological parameters like the diameter $D(G)$, the maximum and minimum degrees $\Delta(G)$ and $\delta(G)$, the characteristic path length $L(G)$, the number of cycles and the betweenness centrality can be found in [29, 30, 52, 62]. Specifically, some spectral bounds for either the node betweenness and edge betweenness, as the following, are presented in [29]:

$$\frac{n}{\sqrt{\lambda_2(G)(2\Delta - \lambda_2(G))}} \leq B_{\text{Emax}} \leq B_{\max} + 2$$

and

$$D(G) \leq 2\left(\frac{\ln(n/2)}{\ln \frac{B_{\text{Emax}}\Delta + n}{B_{\text{Emax}}\Delta - n}}\right).$$

To finish this subsection, it is important to remark, as it can be seen in [50], that a sufficiently large value of spectral gap $s(G)$ is considered as a necessary condition for the so-called "good expansion" properties. Furthermore, if $s(G)$ is low then G has no good expansion properties that are usually related to the number and quality of cut edges, bridges, and the existence of bottlenecks in the network [50, 70, 114].

1.5 A Common Framework to Several Structural Vulnerabilities

As we have seen in the previous section, many of the vulnerability functions introduced in complex network's literature are actually some kind of aggregation of local parameters such as the variation of performance, connectivity properties or betweenness; in [39], a general framework is introduced in order to give a new family of vulnerability functions that extends those known functions. Depending on the size of the network, the nature of the problem, the type of applications we are analyzing or even the target we are pursuing, we will have to decide which vulnerability function is best suited for that analysis. Many of these approaches are particular cases for specific values of the parameters p and q and for a specific function ψ of the general framework given by the concept of (ψ, p, q)-vulnerability [39]. So, inside this general framework, we have more information to decide which vulnerability function (the particular values of p and q, and the specific function ψ) is best suited to analyze a specific problem. Inside this general framework, we can obtain some bounds and relationships amongst these vulnerability functions and we show their sharpness through some relevant simulations (see [39]). This general framework gives us a relevant and useful tool to be applied to real-world complex networks.

To fix ideas, if G is a complex network, Y is a subset of ordered pairs of nodes or links, Z is a subset of nodes or links, and $\psi : Y \times Z \longrightarrow [0, +\infty)$ is a function, then the (ψ, p, q)-*vulnerability of* G is defined for any $p, q \in [0, \infty]$, as the value

$$V_{\psi,p,q}(G) = \left[\frac{1}{|Z|} \sum_{z \in Z} \left(\frac{1}{|Y|} \sum_{(i,j) \in Y} \psi(i,j,z)^p \right)^{q/p} \right]^{1/q}.$$

As we can see, $V_{\psi,p,q}(G)$ is an aggregation of the function $\psi(i,j,z)$ through all the possible values of $(i,j) \in Y$ and $z \in Z$. Let us notice that many of the different definitions for the vulnerability of a complex network are particular cases for the (ψ, p, q) -vulnerability.

For example, if we consider the vulnerabilities of a network based on the fall of efficiency due to a failure of a single node $V_{\max}(G)$ and $\overline{V}(G)$ as it was introduced in Sect. 1.4.2. These two definitions of vulnerabilities are particular cases of (ψ, p, q)-vulnerabilities simply by considering

$$Y = \{(i,j);\ \ i \neq j \in X\},$$

$Z = X$ and taking $\psi_1 : Y \longrightarrow [0,1]$ defined for every $i, j, v \in X$ $(i \neq j)$ by

$$\psi_1(i,j,v) = \begin{cases} \frac{1}{n(n-1)} \frac{1}{d_{ij}} - \frac{1}{(n-1)(n-2)} \frac{1}{d'_{ij}}, & \text{if } i \neq v \neq j, \\ \frac{1}{n(n-1)} \frac{1}{d_{ij}}, & \text{otherwise,} \end{cases}$$

where d'_{ij} is the geodesic distance in $G \setminus \{v\}$. By using these settings it is easy to check that $\overline{V}(G)$ is the $(\psi_1, 1, 1)$-vulnerability of G and $V_{\max}(G)$ is the $(\psi_1, 1, \infty)$-vulnerability of G. Then $V_{\psi_1,1,q}(G)$ interpolates between $\overline{V}(G)$ and $V_{\max}(G)$ in the range $q \in [1, \infty]$ (see [39]).

Following similar techniques, it can be checked that if we take the functions $\psi_2, \psi_3 : Y \longrightarrow [0,1]$ given by

$$\psi_2(i,j,\ell) = \frac{n_{ij}(\ell)}{n_{ij}},$$

$$\psi_3(i,j,v) = \frac{n_{ij}(v)}{n_{ij}},$$

then $V_{E,q}(G) = V_{\psi_2,1,q}(G)$ and $V_{X,q}(G) = V_{\psi_3,1,q}(G)$ for every $q \in [1, +\infty)$ (see [39]), where $V_{E,q}(G)$ and $V_{X,q}(G)$ are the multi-scale node-based and link-based vulnerabilities presented in Sect. 1.4.3. Furthermore, this general framework can be also applied to the bottleneck-type vulnerability, since if we take $\psi_5 : Y \longrightarrow [0,1]$ defined for every $i, j, v \in X$ $(i \neq j)$ as

$$\psi_5(i,j,v) = \chi_{\Pi_g(i,j)}(v) = \begin{cases} 1, \text{ if } v \in \Pi_g(i,j), \\ 0, \text{ otherwise}, \end{cases}$$

then it was proved in [39] that $V_{\psi_5,1,1}(G) = \frac{1}{n}(B_{\mathrm{g}}(G) + 1)$, where $B_{\mathrm{g}}(G)$ is the bottleneck-type vulnerability presented in Sect. 1.4.4.

The main advantages of these unified approach to vulnerability functions are that it allows to prove new analytical results that connects different vulnerability measures and also it helps to introduce new vulnerability functions easily (see [39]). By using some tools from the geometric functional analysis it can be proved sharp analytical relationships between the different vulnerability measures, as the following:

Theorem 1.5 ([39]). *Let G be a complex network with n nodes and let $1 \leq p, q \leq \infty$, then*

$$V_{\psi_1,p,q}(G) \leq C_n V_{\psi_5,p,q}(G),$$

and therefore $\tilde{V}(G) \leq C_n B_{\mathrm{g}}(G) + C_n$, where C_n is a positive constant only depending on n.

By using similar techniques, many other sharp analytical relationships can be stated between $V_{\psi_j,1,1}(G)$ for $j = 1, \ldots, 5$ and $p, q \in [1, +\infty]$ that were illustrated for the Erdös–Rénji and the Barabasi–Albert random models of complex networks in [39].

1.6 Structural Vulnerability: Some Results in Different Models

In the last years, several models of complex networks have been proposed after the pioneering random graph of Erdös–Rénji [49] as the small-world model of Watts and Strogatz [110] or the scale-free networks of Barabási and Albert [10]. The main reason for this was the discovery that real networks like the Internet graph have characteristics which are not explained by uniformly random connectivity. Instead, networks derived from real data may involve power law degree distributions and hubs among other structural features. Watts and Strogatz [110] proposed the first model that conciliated the existence of a large clustering with a small diameter or characteristic path length. They found out that many real world networks exhibit what is called the small-world property, i.e. most vertices can be reached from the others through a small number of edges, like in social networks.

In 1999 [5, 10], Barabási and coworkers found that the degree distribution of some complex systems follows "power laws" instead of being Poisson-like distribution, showing that the structure and the dynamics of that systems are strongly affected by nodes with a great number of connections and, additionally, many of that systems are strongly clustered with a big number of short paths between the nodes, i.e., they obey the small world property.

A network with degree power law distribution is called *scale-free*. A power-law function can be expressed as a polynomial $p(x) = ax^{-\gamma}$, where a and γ are constants and γ is called the power-law exponent. A power law distribution has no peak at its average value and is a relatively slow decreasing function, but the main property of power laws is their scale invariance, i.e., if we substitute the argument x by the same argument multiplied by a scaling factor c we get a proportionate scaling of the function itself, i.e., $p(cx) = a(cx)^{-\gamma} = c^{-\gamma}p(x) \propto p(x)$, which means that they are proportional and therefore it preserves the shape of the function itself. Moreover, by taking logarithms the following linear relation is obtained $\log p(x) = \log a - \gamma \log x$.

The model proposed by Barabási and Albert [10] is based on two observed facts in real networks: networks expand continuously by the addition of new vertices, and new vertices attach preferentially to sites that are already well connected. The model starts with a small number of nodes at step $t = 0$, and at every time step a new node is connected to a number of nodes of the existing graph. The probability of the new node to be connected to an existing node depends on the degree of that node, in the sense that nodes with higher degree have stronger ability to grab links added to the network.

A typical value for the degree power-law exponent in real networks is $2 \leq \gamma \leq 3$. The Barabási–Albert model produces a degree power-law distribution with exponent $\gamma = 3$, meanwhile the Watts–Strogatz and the Erdös–Rénji follow a Poisson distribution.

Despite the fact that, as we have said, various authors have observed that real-world networks have power-law degree distribution, the Erdös–Rénji random graph still has many modeling applications. The modeling of wireless ad-hoc and sensor-networks, peer-to-peer networks like Gnutella [24] and, generally, overlay-networks, provide some well-known examples [57].

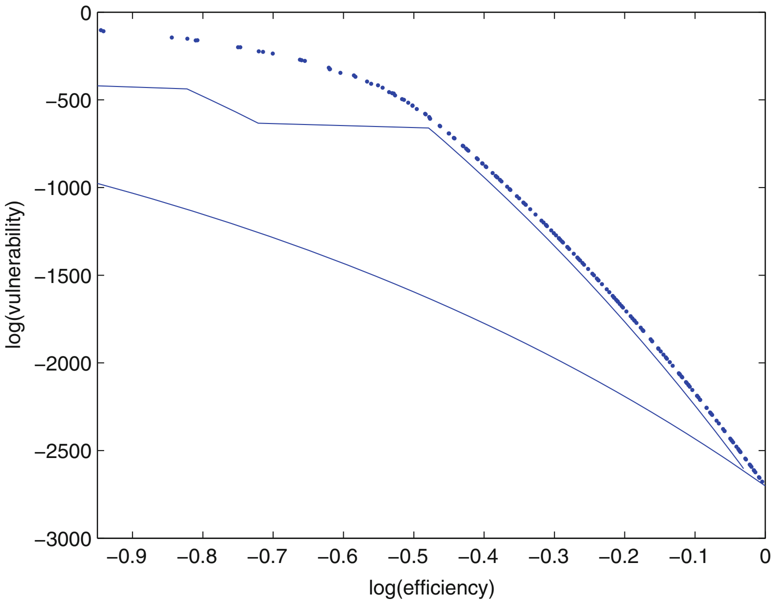


Fig. 1.6 Vulnerability $V_1(\cdot)$ vs efficiency in a simulation of 1,000 Erdös–Rénji random graphs with 100 nodes and variable p, and two different estimates

In [69], the authors study the algebraic connectivity in the Erdös–Rénji model in relation to the graphs robustness based on node connectivity $\kappa(G)$ and link connectivity $\lambda(G)$. Extensive simulations show that the node and the link connectivity converge to an identical distribution to that of the with the minimum nodal degree $\delta(G)$, already at small graph sizes. This makes $\delta(G)$ a valuable estimate of the number of nodes or links whose deletion results into a disconnected random graph. Simulations in [69] also show that, for large n, the distribution of the algebraic connectivity grows linearly with $\delta(G)$. The case of $V_1(\cdot)$ and $V_2(\cdot)$ was considered in [37] for the Erdös–Rénji model. In this paper, a probabilistic analytical concentration result for there vulnerability functions was proved in the classic Erdös–Rénji and some random test was proved for 1,000 random graphs for different values of p that strongly connects the vulnerability with the efficiency, as Fig. 9.2 shows

The multi-scale node and link-based vulnerabilities were studied in detail in [38]. In this paper, the authors proved a result that uses sharp inequalities related to finite dimensional Banach spaces and a numerical analysis of this sharp result was also presented in [38], where the relationship between the node based vulnerability and the link-based multi-level measure in the Erdös–Rénji model suggest a deep connection between these parameters, as Fig. 1.7 shows.

A similar study was performed in [39] for different kinds of vulnerability measures, but including on only the Erdös–Rénji model but also the Barabasi–Albert

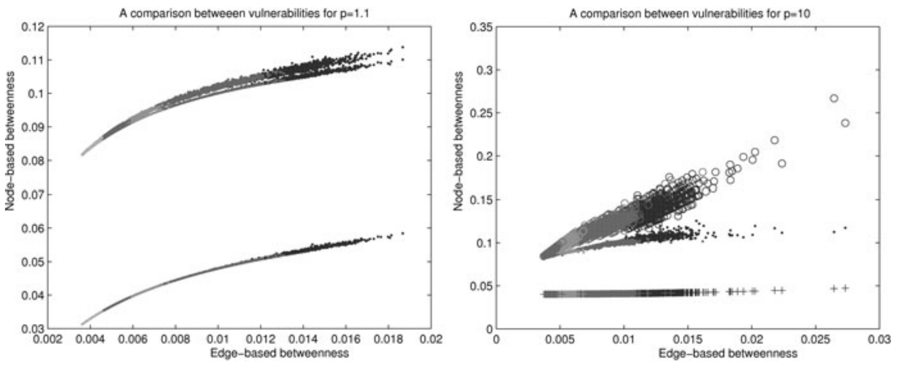


Fig. 1.7 Values for node-based and edge-based multi-scale vulnerabilities for $p = 1.1$ (*on the left*) and $p = 10$ (*on the right*) in a sample of 1,000 random graphs of Erdős–Rénji type with $n = 25$ and linking probabilities varying from 0.4 to 0.9 with a step of 0.1

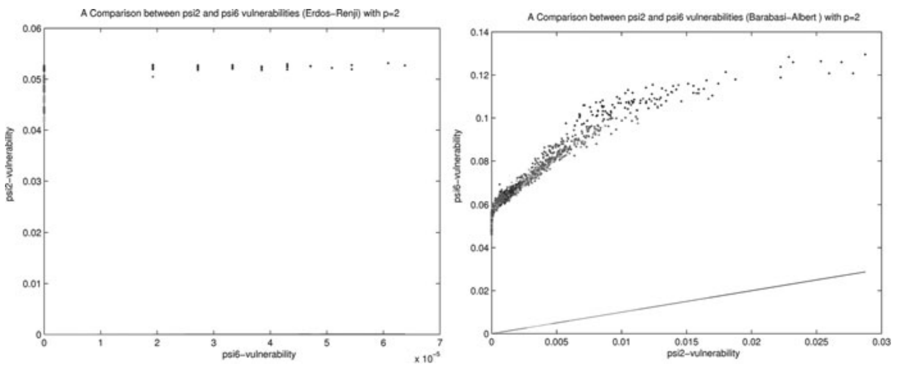


Fig. 1.8 Two comparisons between the fall of efficiency vulnerability (V_{ψ_2}) vs. the multilevel node-based vulnerability (V_{ψ_6}) for the Erdös–Rénji model (*on the left*) and the Barabasi–Albert model (*on the right*)

one, obtaining that the second model present a more heterogeneous behavior when we are dealing with vulnerability function, as Fig. 1.8 shows.

Nevertheless, different approaches to address networks structural vulnerability in the different complex networks models have been proposed by research community [4, 15, 63, 66, 77, 97, 106]. In general, it was concluded that the more heterogeneous a network is in terms of, e.g., degree distribution, the more robust it is to random failures, while, at the same time, it appears more vulnerable to deliberate attacks on highly connected nodes.

To finish this section let us observe that recently, in [84] a measure of network vulnerability called vulnerability index is introduced. This index was calculated for the Erdös–Rénji model, the Barabasi–Albert model of scale-free networks, the Watts–Strogatz model of small-world networks, and for geometric random networks. The model of small-world network appears to be the most robust network among all these models. This conclusion is due obviously to the fact that this model shows highest structural robustness when nodes or edges are removed from the network. Some other real-world networks were compared using this vulnerability index: two human brain networks, three urban networks, one collaboration network, and two power grid networks. The authors found that human brain networks were the most robust networks among all real-world networks studied in [84].

The empirical analysis of the structural vulnerability functions for some real-life networks (in addition to the results presented in [84]), includes the analysis of some technological networks, such as water distribution systems [113], Public Transportation Networks (see, e.g. [36] and [38]) and airport networks in Europe (see [15]). As an example, in [36] the authors presented a comparative analysis of more than 60 worlds city subways, according to several structural parameters that includes the structural vulnerability. Following with the analysis of public transportation systems (metro), in [38]) a detailed analysis of Madrid Underground is presented including the node-based and link-based vulnerability and a ranking of the more vulnerable stations according to the structural vulnerability of the system.

Acknowledgements This work has been supported by the Spanish Government Project MTM2009-13848.

References

1. Agarwal, J., Blockley, D.I. and Woodman, N.J.: Vulnerability of systems. Civil Eng. and Env. Syst. **18**, 14165 (2001)
2. Agarwal, J., Blockley, D.I. and Woodman, N.J.: Vulnerability of structural systems. Structural Safety **25**, 263286 (2003)
3. Albert, R., Albert, I., Nakarado, G.L.: Structural vulnerability of the North American power grid. Phys. Rev. E **69**, 025103 (2004)
4. Albert, R. and Barabási, A.L.: Statistical mechanics of complex networks. Rev. Mod. Phys. **74**, 47–97 (2002)
5. Albert, R., Jeong, H. and Barabási, A.L.: Diameter of the world-wide web. Nature **401**, 130–131 (1999)
6. Albert, R., Jeong, H. and Barabási, A.L.: Error and attack tolerance of complex networks. Nature **406**, 378 (2000)
7. Amaral, L. A. N. and Ottino, J. M.: Complex networks. European Physical Journal B, **38**, 147–162 (2004)
8. Arenas, A., Danon, L., Daz-Guilera, A., Gleiser, P.M. and Guimer, R.: Community analysis in social networks. Eur. Phys. Journal B, **38** 373–380 (2004)
9. Bao, Z.J., Cao, Y.J., Ding, L.J. and Wang, G.Z.: Comparison of cascading failures in small-world and scale-free networks subject to vertex and edge attacks. Physica A, **388**, 4491–4498 (2009)
10. Barabási, A.L. and Albert, R.: Emergence of scaling in random networks, Science **286**, 509–512 (1999)
11. Barefoot, C.A., Entringer, R. and Swart, H.: Vulnerability in graphs a comparative survey. J.Comb.Math.and Comb.Comput. **1**, 13–22 (1987)
12. Bar-Yam, Y.: Dynamics of ComplexSystems. Addison-Wesley, 1997.

13. Berdica, K.: An introduction to road vulnerability: what has been done, is done and should be done. Transport Policy **9**(2), 117–127 (2002)
14. Biggs, N.: Algebraic Graph Theory, 2nd Edition. Cambridge University Press, 1993.
15. Boccaletti, S., Buldú, J., Criado, R., Flores, J., Latora, V., Pello, J., Romance, M.: Multi-scale Vulnerability of Complex Networks. Chaos **17**, 043110 (2007)
16. Boccaletti, S., Criado, R., Pello, J., Romance, M., Vela-Pérez, M.: Vulnerability and fall of efficiency in complex networks: A new approach with computational advantages. Int. J. Bifurcat. Chaos **19**(2), 727–735 (2009)
17. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex Networks: Structure and Dynamics. Phys. Rep, **424**, 175 (2006)
18. Boguña, M., Serrano, M.: Generalized percolation in random directed networks. Phys. Rev. E **72**, 016106 (2005)
19. Bollobás, B.: Random graphs, 2nd edn. Cambridge University Press, Cambridge, 2001
20. Bonacich, P.: Factoring and weighing approaches to status scores and clique information. J. Math. Soc. **2**, 113 (1972)
21. Bonacich, P., Lloyd, P.: Eigenvectors-like measures of centrality for asymetric relations. Soc. Netw. **23**, 191 (2001)
22. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment (Information Security and Cryptography). Springer (2003)
23. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Comput. Netw. **30**, 107 (1998).
24. Castro, M., Costa, M., Rowstron, A.: Should We Build Gnutella on a Structured Overlay. ACM SIGCOMM Computer Communications Review **34**(1), 131136 (2004)
25. Chung, F.R.K.: Spectral Graph Theory. Conference Board of the Mathematical Sciences, AMS, Providence, RI, **92** (1997)
26. Chung, F.R.K., Lu, L., Vu, V.: The spectra of random graphs with given expected degrees. Proc. Natl. Acad. Sci. **100** 6313–6318, (2003)
27. Cohen, R., Erez, K., ben-Avraham, D., Havril, S.: Resilience of the internet to random breakdowns. Phys. Rev. Lett. **85**(21), 4626 (2000)
28. Cohen, R., Erez, K., ben-Avraham, D., Havril, S.: Breakdown of the internet under intentional attacks. Phys. Rev. Lett. **86**(16), 3682 (2001)
29. Comellas, F., Gago, S.: Spectral bounds for the betweenness of a graph. Linear Algebra Appl. **423**, 74–80 (2007)
30. Comellas, F., Gago, S.: Synchronizability of complex networks, J. Phys. A: Math. Theor. **40**, 4483–4492 (2007)
31. Criado, R., Flores, J., González-Vasco, M.I., Pello, J.: Locating a leader node on a complex network. J. Comput. Appl. Math. **204**, 10 (2007)
32. Criado, R., García del Amo, A., Flores, J., Romance, M.: Analytical relationships between metric and centrality measures of a network and its dual. J. Comput. Appl. Math. **235**, 1775–1780 (2011)
33. Criado, R., Flores, J., Pello, J., Romance, M.: Optimal communication schemes in a complex network: From trees to bottleneck networks. Eur. Phys. J.-Spec. Top. **146**, 145–157 (2007)
34. Criado, R., García del Amo, A., Hernández-Bermejo, B., Romance, M.: New results on computable efficiency and its stability for complex networks. J. Comput. Appl. Math. **192**, 59 (2006)
35. Criado, R., Flores, J., Hernández-Bermejo, B., Pello, J., Romance, M.: Effective measurement of network vulnerability under random and intentional attacks. J. Math. Model. Alg. **4**, 307–316 (2005)
36. Criado, R., Hernández-Bermejo, B., Romance, M.: Efficiency, vulnerability and cost: an overview with applications to subway networks worldwide. Int. J. Bifurcat. Chaos **17**(7), 2289 (2007)
37. Criado, R., Hernández-Bermejo, B., Marco-Blanco, J. Romance, M.: Probabilistic analysis of efficiency and vulnerability in the Erdös-Rénji model. Int.J. Comput. Math. **85**(3-4), 411–419 (2008)

38. Criado, R., Pello, J., Romance, M., Vela-Pérez, M.: A node based multi-scale vulnerability of Complex Networks. Int. J. Bifurcat. Chaos **19**(2), 703–710 (2009)
39. Criado, R., Pello, J., Romance, M., Vela-Pérez, M.: (ψ, p, q)-Vulnerabilities: An unified approach to network robustness. Chaos **19**, 013133 (2009)
40. Criado, R., Pello, J., Romance, M., Vela-Pérez, M.: Improvements in performance and security for complex networks. Int. J. of Comp. Math., **86**, 2, 209–218 (2009)
41. Crucitti, P., Latora, V., Marchiori, M., Rapisarda, A.: Efficiency of Scale-Free Networks: Error and Attack Tolerance. Physica A, **320**, 622 (2003)
42. Crucitti, P., Latora V., Marchiori, M.: Error and attack tolerance of complex networks. Physica A **340** 388–394 (2004)
43. Cvetkovic, D., Doob, M., Gutman, I., Torgasev, A.: Recent Results in the Theory of Graph Spectra, North-Holland, Amsterdam, 1988.
44. Cvetkovic, D.M., Doob, M., Sachs, H.: Spectra of Graphs, Theory and Applications, 3rd edn. Johann Ambrosius Barth, Heidelberg, 1995
45. Cvetkovic, D., Rowlinson, P.S.K. Simic: Eigenspaces of Graphs. Cambridge University Press, Cambridge, 1997.
46. Dekker, A.H., Colbert, B.D.: Network Robustness and Graph Topology. Proc. ACSC04, the 27th Australasian Computer Science Conference (18-22 January 2004), Dunedin, New Zealand (2004)
47. Diestel, R.: Graph Theory. Springer-Verlag (2005)
48. Dorogovtsev, S.N., Mendes J.F.F.: Evolution of networks. Adv. Phys. **51**, 10791187 (2002)
49. Erdös, P., Rénji, A.: On Random Graphs I, Publ. Math. **6**, 290–297 (1959)
50. Estrada, E.: Network robustness to targeted attacks. The interplay of expansibility and degree distribution. Eur. Phys. J.B., **52**, 563–574 (2006)
51. Estrada, E., Rodríguez-Velázquez, J.: Subgraph centrality in complex networks. Phys. Rev. E **71**, 056103 (2005)
52. Farkas, I.J., Derenyi, I., Barabási, A.-L. and Vicsek, T.: Spectra of realworld graphs: beyond the semicircle law. Physical Review E, **64**:026704 (2001)
53. Fiedler, M.: Algebraic Connectivity of Graphs. Czech. Math. J. **23**, 298 (1973)
54. Fontoura Costa, L. et al: Characterization of Complex Networks: A Survey of measurements. Advances in Physics, **56**, 167–242 (2007)
55. Fontoura Costa, L. et al: Analyzing and Modeling Real-World Phenomena with Complex Networks: A Survey of Applications. arXiv:0711.3199v3 [physics.soc-ph] (2008)
56. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**, 35–41, 1977.
57. Fuhrmann, T.: On the Topology of Overlay Networks. In Proceedings of 11th IEEE International Conference on Networks (ICON), pp. 271–276 (2003)
58. Gago, S.: Spectral Techniques in Complex Networks. Selectec Topics on Applications of Graph Spectra, Matematicki Institut SANU, Beograd, **14**(22), 63–84, 2011.
59. Gibbons, A.: Algorithmic Graph Theory. Cambridge University Press (1985)
60. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA **99**, 7821–7826 (2002)
61. Godsil, C.D. and Royle, G.: Algebraic Graph Theory. Springer, 2001.
62. Goh, K.-I. Kahng, B. and Kim, D.: Spectra and eigenvectors of scale-free networks. Physical Review E, **64**:051903 (2001)
63. Goldshtein, V., Koganov, G.A and Surdutovich, G.I.: Vulnerability and hierarchy of complex networks. cond-mat/0409298 (2004)
64. Guellner, C. and Costa, C.H.: A Framework for Vulnerability Management in Complex Networks. IEEE Ultra Modern Telecommunications, ICUMT.09, 1–8 (2009)
65. Harary, F.: Graph Theory. Perseus, Cambridge, MA. (1995)
66. Holme, P., Beom Jun Kim, Chang No Yoon, Seung Kee Han: Attack vulnerability of complex networks. Phys. Rev. E **65**, 056109 (2002)
67. Holmgren, J.: Using graph models to analyze the vulnerability of electric power networks. Risk Anal. **26**(4), (2006)

68. Husdal, J.: Reliability and vulnerability versus cost and benefits. Proc. 2nd Int. Symp. Transportation Network Reliability (INSTR). Christchurch, New Zealand, 180–186 (2004)
69. Jamakovic, A., Van Mieghem, P.: On the robustness of complex networks by using the algebraic connectivity. NETWORKING 2008, LCNS 4892, 183–194, 2008.
70. Jamakovic, A., Uhlig, S.: On the relationship between the algebraic connectivity and graphs robustness to node and link failures, Proc. 3rd EURO-NGI Conf. Next Generation Internet Network, Trondheim, Norway, 96–102 (2007)
71. Jeong, H., Mason, S., Barabási, A.L., Oltvai, Z.N.: Lethality and centrality in protein networks. Nature **411**, 41–42 (2001)
72. Juh$\tilde{a}$z, F.: The asymptotic behaviour of Fiedlers algebraic connectivity for random graphs. Discrete Mathematics **96**, 59–63 (1991)
73. Koschitzki,D., Lehmann, K. A., Peeters, L., Richter, S., Tenfelde-Podehl, D. and Zlotowski, O.: Centrality indices. LNCS 3418, 2005.
74. Latora, V., Marchiori, M.: Efficient Behavior of Small-World Networks. Phys. Rev. Lett. **87**, 198701 (2001)
75. Latora, V., Marchiori, M.: Economic small-world behaviour in weighted networks. Eur. Phys. J.B. **32**, 249–263 (2003)
76. Latora, V., Marchiori, M.: How the science of complex networks can help developing strategies against terrorism. Chaos Solitons Fract. **20**, 69 (2004)
77. Latora, V., Marchiori, M.:Vulnerability and protection of critical infrastructures. Phys Rev E **71**, 015103 (2004)
78. Latora, V., Marchiori, M.: A measure of centrality based on the network efficiency. New J. Phys. **9**, 188 (2007)
79. Lu, Z., Yu, Y., Woodman, N.J., Blockley, D.I.: A theory of structural vulnerability. Struct. Eng. **77**(18), 17–24 (1999)
80. Mehta, M.L.: Random Matrices. Academic Press, 1991.
81. Menger, K.: Zur allgemeinen Kurventheorie. Fund. Math. **10**, 96–115, (1927)
82. Merris, R.: Laplacian matrices of graphs: a survey. Linear Algebra Appl. **197-198**, 143–176 (1994)
83. Merris, R.: A survey of graph Laplacians. Linear and Multilinear Algebra **39**, 19–31 (1995)
84. Mishkovski, I., Biey, M. and Kocarev, L.: Vulnerability of Complex Networks. Commun Nonlinear Sci Numer Simulat **16**, 341–349 (2011)
85. Mohar, B.: The Laplacian spectrum of graphs. Graph Theory, Combinatorics and Applications **2**, 871–898 (1991)
86. Mohar, B.: Eigenvalues, diameter and mean distance in graphs. Graphs Combin. **7**, 53–64 (1991)
87. Mohar, B.: Laplace eigenvalues of graphs: a survey. Discrete Mathematics **109**, 198, 171–183 (1992)
88. Mohar, B., Hahn, G., Sabidussi, G.: Some applications of Laplace eigenvalues of graphs. Graph Symmetry: Algebraic Methods and Applications, NATO ASI Ser. C 497, 225–275 (1997)
89. Motter, A.E., Lai, Y-C.: Cascade-based attacks on complex networks. Phys. Rev. E **66**, 065102(R)(2002)
90. Motter, A.E.: Cascade control and defense in complex networks. Phys. Rev. Lett. **93**, 098701 (R)(2004)
91. Newman, M.E.J.: The structure and function of complex networks. SIAM Review **45**, 167–256 (2003)
92. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E 7**4**, 036104 (2006)
93. Newman, M.E.J.: Networks: An Introduction. Oxford Univ. Press, Oxford, 2010
94. Newman, M.E.J., Barabási, A.L., Watts, D.J.: The Structure and Dynamics of Networks. Princeton Univ. Press, Princeton, New Jersey (2006)
95. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**, 026113 (2004)

96. Newman, M.E.J., Goshal, G.: Bicomponents and the robustness of networks to failure, Phys. Rev. Lett. **100**, 138701 (2008)
97. Ouyang, M. et al.: A methodological approach to analyze vulnerability of interdependent infrastructures, Simulation Modelling Practice and Theory **17**, 817–828 (2009)
98. Rodríguez-Velázquez, J., Estrada, E., Gutiérrez, A.: Functional centrality in graphs. Linear and Multilinear Algebra **55**(3), 293–302 (2007)
99. Rosato, V. and Tiriticco,F.: Growth mechanisms of the AS-level internet network. Europhysics Letters, **66**(4):471–477 (2004)
100. Schwartz, N., Cohen, R., ben Avraham, D., Barabasi, A.L., Havlin, S.: Percolation in directed scale-free networks. Phys. Rev. E **66**, 015104(R) (2002)
101. Seary, A. J. and Richards, W.D.: Spectral methods for analyzing and visualizing networks: an introduction. In Dynamic Social Network Modeling and Analysis, pages 209–228. National Academy Press, 2003.
102. Serrano, M., Boguña, M.: Clustering in complex networks. ii. percolation properties. Phys. Rev. E **74**, 56115 (2006)
103. Strogatz, S.H.: Exploring complex networks. Nature **410**, 268–276 (2001)
104. Taylor, M.A.P., D'Este, G.M.D.: Network Vulnerability: An Approach to Reliability Analysis at the Level of National Strategic Transport Networks. Proc. 1st Int.Symp. on Transportation Network Reliability, 23–44 (2003)
105. Taylor, M.A.P., D'Este, G.M.D.: Concepts of network vulnerability and applications to the identification of critical elements of transport infrastructure. Paper presented at the 26th Australasian Transport Research Forum, Wellington, New Zealand,1-3 October 2003
106. Trpevski, D., Smilkov, D., Mishkovski, I. and Kocarev, L.: Vulnerability of labeled networks. Physica A **389**, 23, 5538–5549(2010)
107. Van Mieghem, P.: Performance Analysis of Communications Networks and Systems. Cambridge University Press, Cambridge, 2006.
108. Wang,Y., Chakrabarti, D., Wang, C., Faloutsos,C.: Epidemic spreading in real networks: An eigenvalue viewpoint. 22nd Symp. Reliable Distributed Computing, Florence, Italy, Oct. 68, 2003.
109. Wasserman, S., Faust, K.: Social Networks Analysis. Cambridge Univ. Press (1994)
110. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-world networks. Nature **393**, 440–442 (1998)
111. Wehmuth, K. et al: On the joint dynamics of network diameter and spectral gap under node removal. Latin-American Workshop on Dynamic Networks, Buenos Aires (2010)
112. Wu, J., Deng, H.Z., Tan, Y.J. and Zhu, D.Z.: Vulnerability of complex networks under intentional attack with incomplete information. Journal of Physics A: Mathematical and Theoretical, **40**, 11, 2665–2671 (2007)
113. Yazdani, A., Jeffrey, P.: Complex network analysis of water distribution systems, to appear in Chaos (2011)
114. Yazdani, A., Jeffrey, P.: A note on measurement of network vulnerability under random and intentional attacks. e-print: http://arxiv.org/abs/1006.2791 (2010)

Chapter 2
Optimizing Network Topology for Cascade Resilience

Alexander Gutfraind

Abstract Complex networks need resilience to cascades – to prevent the failure of a single node from causing a far-reaching domino effect. Such resilience can be achieved actively or topologically. In active resilience schemes, sensors detect the cascade and trigger responses that protect the network against further damage. In topological resilience schemes, the network's connectivity alone provides resilience by dissipating nascent cascades. Designing topologically resilient networks is a multi-objective discrete optimization problem, where the objectives include resisting cascades and efficiently performing a mission. Remarkably, terrorist networks and other "dark networks" have already discovered how to design such networks. While topological resilience is more robust than active resilience, it should not always be pursued because in some situations it requires excessive loss of network efficiency.

2.1 Introduction

Cascades are ubiquitous in complex networks and they have inspired much research in modeling, prediction and mitigation [11, 14, 20, 35, 53, 54, 57, 60, 62, 72]. For example, since many infectious diseases spread over contact networks a single carrier might infect other individuals with whom she interacts. The infection might then propagate widely through the network, leading to an epidemic. Even if no lives are lost, recovery may require both prolonged hospitalizations and expensive treatments. Similar cascade phenomena are found in other domains such as power distribution systems [22, 38, 43], computer networks such as ad-hoc

A. Gutfraind
Center for Nonlinear Studies and T-5/D-6, Los Alamos National Laboratory,
Los Alamos, NM 87545, USA
e-mail: mailto:agutfraind.research@gmail.com

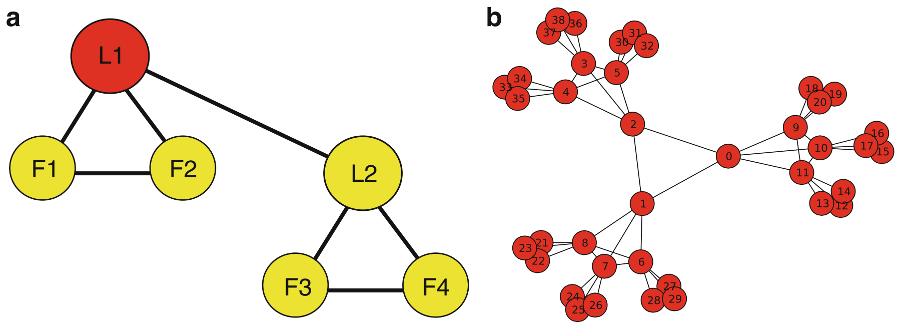


Fig. 2.1 The French World-War II underground network *Francs-tireurs et Partisans* (FTP) reconstructed by the author based on the account in [51]. Its organizational unit was the combat group (**a**). In an idealized case, nor always followed, this was divided into two "teams" of three fighters, where leader L1 was in overall command and in command of team 1. His lieutenant, L2, led team 2 and assumed overall command if L1 was captured. The small degree of the nodes ensured that the capture of any one node did not risk the exposure of a significant fraction of the organization. Each "group" is in a command hierarchy (**b**) where three groups (bottom-level nodes) made a "section," three sections made a "company," and finally three companies made a "battalion"

wireless networks [60], financial markets [8, 36], and socio-economic systems [40]. A particularly interesting class are "dark" or clandestine social networks, such as terrorist networks, guerrilla groups [65], espionage, and crime rings [5, 52]. In such networks, if one of the nodes (i.e. individuals) is captured by law enforcement agencies, he may betray all the nodes connected to him leading to their likely capture.

Dark networks are therefore designed to operate in conditions of intense cascade pressure. As such they can serve as useful prototypes of networks that are cascade-resilient because of their connectivity structure (topology) alone. Their nodes are often placed in well-defined cells – closely-connected subnetworks with only sparse connections to the outside (for an example from World War II see Fig. 2.1) [51]. The advantages of cells are thought to be that the risk from the capture of any person is mostly limited to his or her cell mates, thereby protecting the rest of the network [29, 48]. Modern terrorist groups retain this cellular structure, but increasingly use networks made of components with no connections between them, thus caging cascades within each component [67, 69, 73].

2.1.1 *Active vs. Topological Cascade Resilience*

Networks could be endowed with cascade resilience using two complementary approaches: "active" and "topological." In active resilience, the network is monitored for cascades, and if a cascade is detected, attempts are made to stop it *while it progresses*. For example, in case of human pathogens, health authorities may continuously monitor hospital records for contagious diseases. If the records begin to show anomalous increases, various responses are initiated, including distribution of medicines and alerts to the public. Similarly, in power distribution systems, special devices monitor the network for signs of cascades, such as high currents or phase changes. Those may indicate failures in lines, short circuits and other phenomena that threaten to disrupt the system or damage its components. The power system includes a variety of automated controls tasked with stopping the nascent cascade [53], such as "relays." Those relays can automatically shut down faulty lines or nodes of the network so as to isolate them from the rest of the network [63, 75].

Those two examples of active cascade resilience must be contrasted with "topological" approach to resilience where only the topology (i.e. the pattern of connections) is used to increase cascade resilience. For example, the network could be structured into modules, where any two modules are connected to each other through long paths. As a result, in certain types of cascades, a failure in one module might dissipate before it reaches any other module. When topological cascade resilience is possible, it offers two advantages over active resilience: simplicity and robustness. In topological resilience, the network protects itself, requiring no real-time automated decisions or difficult-to-achieve rapid response during the cascade.

2.2 What is a Cascade-Resilient Network?

The words Network, Cascade, and Resilience have many domains of application, so much so that no universal definition of these terms exists. Therefore, this section briefly surveys some of the recurrent applications and meanings of those terms. It also introduces specific definitions that are appropriate for some applications. Later in the paper these definitions serve as an example of optimizing networks for cascade resilience.

2.2.1 Network as an Unweighted Graph

Complex Networks is the study of real world systems using ideas of graph theory. Specifically, here and in most other studies the network is represented using simple unweighted graph G: a tuple (V,E) where V is a set called "nodes" or "vertices" and E are unordered two-element subsets of V termed "edges." Such an approach offers simplicity and can employ the well-developed tools of graph theory. Ultimately, though, models of networks must consider their evolving nature, fuzzy boundaries, and multiplicities of node classes and diverse relationships.

The simplification is often unavoidable given the lack of data on networks. For example, in dark social networks only the connectivity is known, if that. Fortunately, the loss of information involved in representing networks as simple rather than say, as weighted graphs, could be evaluated. It is shown in Sect. 2.3.1 that at least in two examples where the weights are known, the error in key metrics when using simple graphs has no systematic bias and is usually small.

2.2.2 Cascades

There is a very extensive literature on both cascades and resilience. The classic literature on cascades includes two basic models: percolation cascades and capacity cascades. The former originate in Physics but are often applied to Epidemiology, where they are termed "contagions" or "epidemics" (see e.g. [58]). In percolation phenomena, nodes are assigned states which change because of the influence of their neighbors. For example, an infected node can pass the infection to its contacts in the network, and the infection could then be passed to more and more nodes. Another variant of such percolations are the case where nodes change their state only when a certain fraction of their neighbors exert influence (see e.g. [14, 72]). Percolation phenomena are exceptionally well-studied, and in many variants analytic expressions exist for the final extent of the cascade as a function of the network topology (see e.g. [26, 57]). The capacity cascades are characteristic of capacitated networks, such as power transmission systems and supply chains. Classically, in those systems the edges are assigned capacities and thus carry flows from supply nodes to demand nodes. Cascades occur when due to failures the flow can no longer be carried by the edges within their capacities or when some of the supply nodes fail [3, 21, 43, 53]. In capacity cascades the failure can jump to nodes that are many hops away from the initial failures possibly skipping the neighbors.

2.2.3 Resilience

A vast number of studies attempted to define resilience, often in very different ways. Perhaps the most common meaning refers to the connectivity of the network under disruption or failures in its components. Such definitions are motivated by applications in telecommunications where it is desired that nodes are able to find a path to each other even if some of the components in the networks are damaged or destroyed [1, 6, 7, 12, 15, 16, 23, 33, 34, 41].

The idea of damaging networks has attracted a lot of research in the area of Sociology of secret societies such as terrorist networks [4, 25, 29, 32, 48, 49, 52, 74]. In fact, many secret societies are benign, including non-governmental organizations and dissident movements operating in hostile political environments. In those networks, if the network is penetrated by its enemy, it must be able to minimize the damage. Economists too have recently analyzed the problem of organizational design when the organization is being attacked [27]. Related problems have also been studied by epidemiologists, where the question focused on immunization strategies (e.g. [64]) but apparently not as a question of optimal network design.

Recently, resilience has became associated with the ability to quickly recover from damage rather than to absorb it [10]. Indeed, in many applications disruptions and failures are not rare singular events, but rather occur regularly and even continuously. For example, there are continuous demand spikes in communication networks [19] and voltage fluctuations in power systems.

It is to be expected that no notion of resilience would be useful universally across different applications. Similarly, many networks experience cascades, but the details vary. This paper will investigate cascade resilience under a particularly important and well-characterized class of cascades known as "susceptible-infected-recovered" (SIR). SIR are a type of cascades where any failed node leads to the failure of each neighboring node independently with probability τ [58]. This τ represents the network's propensity to experience cascades, expressing both the susceptibility of components and the environment in which the network operates. Using the SIR model, resilience $R(G)$ could be defined as the average fraction of the network that does not fail in the cascade:

$$R(G) = 1 - \frac{1}{n-1}\mathbb{E}[\text{extent of a cascade}]\,, \tag{2.1}$$

where "extent of a cascade" refers to the ultimate number of new cases created by a single failed node (the initial node does not count) and where $n = |V|$ is the number of nodes. For simplicity, cascades are assumed to start at all nodes with uniform probability.

Observe that under this definition the most cascade-resilient network ($R(G) = 1$) is the network with no edges. But such a network cannot carry any information from node to node! It is not surprising that the objective of designing cascade resilience conflicts with other features of the network. In other cascade types, such as cascades on capacitated networks, the most cascade-resilient network might be the network with infinite capacities, which obviously would conflict with the objective of minimizing cost. It follows then that optimization of networks requires specifying a notion of value or efficiency.

2.2.4 Measuring Efficiency

Notions of network efficiency attempt to quantify the value of a network, and this problem has a long history. For example, an influential early work on communication networks suggested that a network's value increases as $O\left(\frac{n(n-1)}{2}\right)$, because each node can connect to $n-1$ other nodes [9]. However, this measure ignores the difficulty of connecting to other nodes (as well as, e.g. cost). Indeed, it is often desired that the distances between the nodes are short: when nodes are separated by short distances they can, e.g., more easily communicate and distribute resources to each other. Therefore, many authors invoke measures based on the distances between pairs of nodes in the network (see e.g. [44, 49, 55]).

In the following we will consider a version of distance-based efficiency, termed "distance-attenuated reach" metric [44]. For all pairs of nodes $u, v \in V$, weigh each pair by the inverse of its internal distance (the number of edges in the shortest path from u to v) taken to power g:

$$W(G) = \frac{1}{n(n-1)} \sum_{u\in V} \sum_{v\in V\setminus\{u\}} \frac{1}{d(u,v)^g}, \tag{2.2}$$

Normalization by $n(n-1)$ ensures that $0 \leq W(G) \leq 1$, and only the complete graph achieves 1. As usual, for any node v with no path to u, set $\frac{1}{d(u,v)^g} = 0$. The parameter g, "connectivity attenuation" represents the rate at which distance decreases the connectivity between nodes. Unless stated otherwise $g = 1$. A valuable property of $W(G)$ is that it is well-defined and non-singular even on networks that have multiple components with no connections to each other. As will be shown, such a separation into components provides a very powerful mechanism for cascade resilience.

2.3 Evaluating Real Networks

Significant insight into cascade resilience can be derived from comparing the cascade resilience of networks from different domains. We will see that dark networks like terrorist networks are more successful in the presence of certain cascades than other complex networks. Their success stems not from cascade resilience alone but from balancing resilience with efficiency.

To make those comparisons, define the overall "fitness," $F(G)$, of a network by aggregating resilience and efficiency through a weight parameter r:

$$F(G) = rR(G) + (1-r)W(G).$$

The parameter r depends on the application and represents the damage from a cascade – from light ($r \to 0$) to catastrophic ($r \to 1$). Note that it is possible to include in fitness other metrics such as construction cost.

We will compare the fitnesses of several complex networks, including communication, infrastructure and scientific networks to the fitnesses of dark networks. The class of dark networks will be represented by three networks: the 9/11, 11M and FTP networks. The 9/11 network links the group of individuals who were directly involved in the September 11, 2001 attacks on New York and Washington, DC [49]. Similarly the 11M network links those responsible for the March 11, 2004 train attacks in Madrid [67]. Both 9/11 and 11M were constructed from press reports of the attacks. Edges in those networks connect two individuals who worked with each other in the plots [49, 67]. The FTP network is an underground group from World War II (Fig. 2.1), whose network was constructed by the author from a historical account [51].

Figure 2.2 shows that the dark networks attain the highest fitness values of all networks, except for extreme levels of cascade risk ($\tau > 0.6$) This is to be expected:

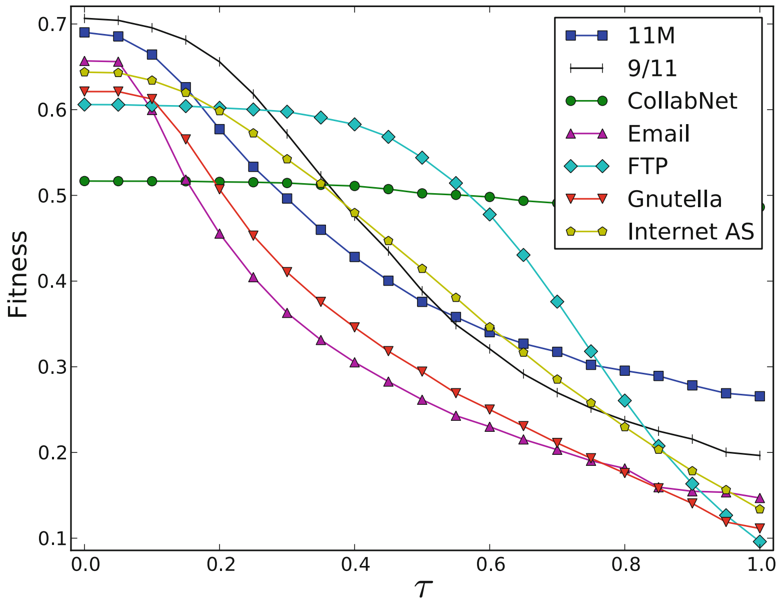


Fig. 2.2 Fitnesses of various networks at $r = 0.51$ and various values of τ. 11M is the network responsible for the March 11, 2004 attacks in Madrid (70 nodes, 240 edges). 9/11 [49] is the network responsible for the 9/11 attacks (62 nodes, 152 edges). CollabNet [59] is a scientific co-authorship network in the area of network science (1,589 nodes, 2,742 edges). E-Mail [28] is a university's e-mail contact network, showing its organizational structure (1,133 nodes, 5,452 edges). FTP is the network in Fig. 2.1 (174 nodes, 300 edges). Gnutella [37,66] is a snapshot of the peer-to-peer network (6,301 nodes, 20,777 edges). Internet AS [47] is a snapshot of the Internet at the autonomous system level (26,475 nodes, 53,381 edges). Except for $\tau > 0.6$ dark networks (11M, 9/11 and FTP) attain the highest fitness

only 11M, 9/11, and the FTP networks have been designed with cascade resilience as a significant criterion – a property that makes them useful case studies. For high cascade risks ($\tau > 0.6$) the CollabNet network exceeds the fitnesses of the dark networks. CollabNet was drawn by linking scientists who co-authored a paper in the area of network science [59]. It achieved high fitness because it is partitioned into research groups that have no publications with outside scientists. Like some terrorist networks, it is separated into entirely disconnected cells.

It is interesting to compare the empirical networks to each other in their efficiency and resilience (Fig. 2.3). Note that FTP and 9/11 networks are not the most resilient, but they strike a good balance between resilience and efficiency. The advantages of the two networks over other networks are not marginal, implying that their advantages in fitness are not sensitive to the choice of r. Of course, they are optimized for particular combinations of r and τ, and will no longer be very successful outside that range. For instance, in the range of high r and high τ networks with multiple connected components would have higher fitness because they are able to isolate cascades in one component.

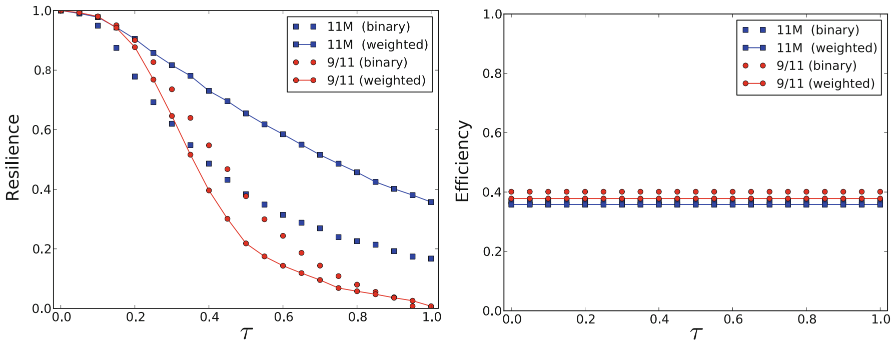


Fig. 2.3 Resilience and efficiency of the real networks. The fittest networks are not always the most resilient

The 9/11 and the 11M networks are very successful for low values of τ (<0.2), but then rapidly deteriorate because of a jump in the extent of cascades – the so-called percolation transition [24]. Past this threshold, cascades start affecting a large fraction of the network, resilience collapses and the fitness declines rapidly. The pattern of onset of failure can be clearly seen in most of the networks. For violent secret societies this transition means that the network might be initially hard to defeat, but there is a point after which efforts against it start to pay off. Because τ is representative of the security environment, the 9/11 network is found to be relatively ill-adapted to the more stringent security regime implemented after the attacks. Indeed, it is likely that the 9/11 attacks would have been thwarted under the current security regime since some of the nodes were captured before the attacks, but not interrogated in time to discover and apprehend the rest of the network [71]. In contrast, the cellular tree hierarchy of the FTP network is more suitable for an intermediate range of cascade risks. However, the pair-wise distances in it are too long to provide high efficiency. Therefore, its fitness is comparatively poor under very low and very high values of τ.

2.3.1 Resilience and Efficiency of Weighted Networks

In some networks, each edge (u,v) carries a distance weight $D_{uv} > 0$. The smaller the distance, the closer the connection between u and v. We now explain in some detail how to compute the fitness of those networks. We will introduce generalizations of resilience and efficiency, that reduce to the original definitions for unweighted networks when $D_{uv} = 1$, while capturing the effects of weights in the weighted networks.

The original definition of resilience was built on a percolation model where the failure of any node leads to the failure of its neighbor with probability τ. In the weighted network, more distant nodes should be less likely to spread the cascade. Thus, we make the probability of cascade through (u,v) to be $\min(\tau/D_{uv},1)$.

The efficiency was originally defined as the sum of all-pairs inverse geodesic distances, normalized by the efficiency of the complete graph. In the weighted network, both the distance and the normalization must be generalized. To compute the distance $d(u,v)$ we consider the weights on the edges D and apply Dijkstra's algorithm to find the shortest path. Normalization too must consider D because a weighted graph with sufficiently small distances could outperform the complete graph (if all the edges of the latter have $D_{ij}=1$). Therefore, we weigh the efficiency by the harmonic mean H of the edges (E) of the graph:

$$W(G) = \frac{H(G)}{n(n-1)} \sum_{u \in V} \sum_{v \in V \setminus \{u\}} \frac{1}{d(u,v)^g}, \tag{2.3}$$

where

$$H(G) = \frac{|E|}{\sum_{(u,v)\in E} \left(\frac{1}{D_{uv}}\right)^g}.$$

The harmonic mean ensures that for any D, the complete graph has $W(G)=1$.

Having defined generalized resilience and efficiency we can evaluate the standard approach to dark networks, which represents them as binary graphs $D_{uv} \in \{0,1\}$, rather than as weighted graphs. The former approach is often taken because the information about dark networks is limited and insufficient to estimate edge weights.

Fortunately, in two cases, the 9/11 network and the 11M network [49, 67] the weights could be estimated. The 9/11 data labels nodes as either facilitators or hijackers. Hijackers must train together and thus should tend to have a closer relationship. Thus set $D_{uv}=2,1,0.5$ if the pair u,v includes zero, one or two hijackers, respectively. The 11M network is already weighted ($Z_{uv}=1,2,3\ldots$) based on the number of functions each contact (u,v) serves (friendship, kin, joint training etc.). We mapped those weights to D by $D_{uv}=2/Z_{uv}$. In both networks, the transformation was so that the weakest ties have weight 2, giving them greater distance than in the binary network, while the strongest ties are shorter than in the binary network.

Figure 2.4 compares the fitnesses, resiliences and efficiencies of the weighted and binary representations. It shows that for both networks, the fitnesses of the binary representation lies within 0.15 of the fitness of the weighted representation and for some τ much closer. The efficiency measures are even more close (within 0.05). The behavior of resilience is intriguing: for the 9/11 network the weighted representation shows more gradual decline as a function of cascade risk when compared to the binary representation. For the 11M network, the decline is actually slightly more sharp in the weighted representation. Structurally, the 11M network has a center (measured by betweenness centrality) of tightly knit-nodes (very short distances), while the 9/11 network is more sparse at its center, increasing its cascade resilience. This effect explains the direction of the error in the binary representation. Based on those two examples, it appears that the binary representation does not have a systematic bias, and may even underestimate the fitness of dark networks.

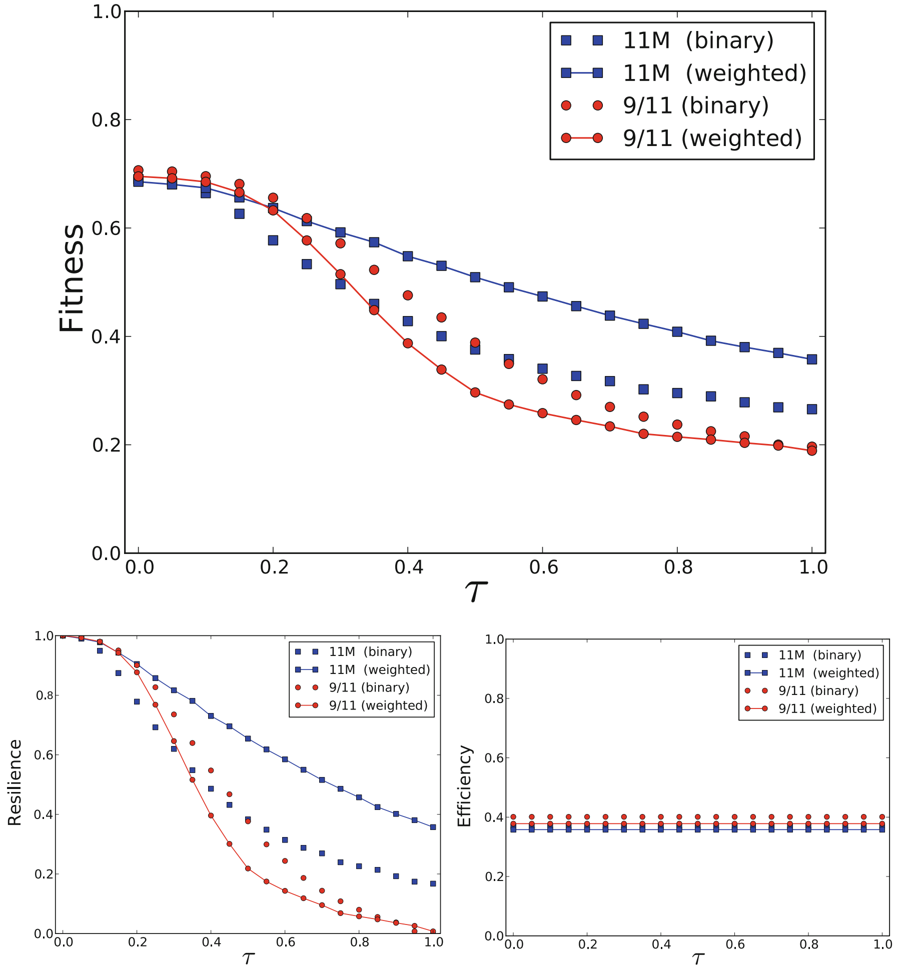


Fig. 2.4 Fitness, resilience, and efficiency of two dark networks ($r = 0.51$), comparing binary and weighted representations. The binary representation matches the weighted representation within 0.15, and typically closer

2.4 Designing Networks

The success of dark networks must be due to structural elements of those networks, such as cells. If identified, those elements could be used to design more resilient networks and to upgrade existing ones. Thus, by learning how dark networks organize, it will be possible to make networks such as communication systems, financial networks, and others more resilient and efficient.

Those identification and design problems are our next task: both will be solved using an approach based on discrete optimization. Let a set of graphs $\mathbb{G}$ be called a

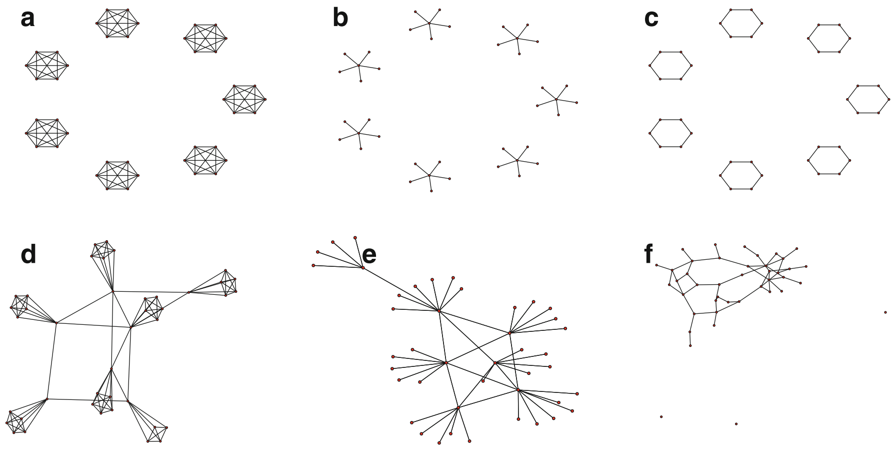


Fig. 2.5 Graphs illustrating the 6 network designs. *Cliques* (**a**), *Stars* (**b**), *Cycles* (**c**), *Connected Cliques* (**d**), *Connected Stars* (**e**), and *Erdos–Renyi "ER"* (**f**). Each design is configured by just one or two parameters (the number of individuals per cell and/or the random connectivity). This enables rapid solution of the optimization problem. In computations the networks were larger ($n = 180$ nodes)

"network design" if all the networks in it share a structural element. Since dark networks are often based on dense cliques, we consider a design where all the networks consist of one or multiple cliques. We consider also designs based on star-like cells, cycle-based cells and more complex patterns (see Fig. 2.5).

In the first step, we will find the most successful network within each design. Namely, consider an optimization problem where the decision variable is the topology G of a simple graph taken from a design $\mathbb{G}$. The objective is the fitness $F(G)$:

$$\max_{G \in \mathbb{G}} F(G) . \tag{2.4}$$

In the second step, we will compare the fitnesses across designs, thus identifying the topological feature with the highest fitness (e.g. star vs. clique).

This optimization problem introduces a method for designing cascade-resilient networks for applications such as vital infrastructure networks. To apply this to a given application, one must make the design $\mathbb{G}$ the set of all feasible networks in that domain, to the extent possible by computational constraints. For a related approach using game-theoretic ideas see Lindelauf et al. [48, 49].

A complementary approach is to consider the multi-objective optimization problem in which $R(G)$ and $W(G)$ are maximized simultaneously:

$$\max_{G \in \mathbb{G}} \{R(G), W(G)\} . \tag{2.5}$$

The multi-objective approach cannot find the optimal network but instead produces the Pareto frontier of each design – the set of network configurations that cannot be improved without sacrificing either efficiency or resilience. The decision maker can use the frontier to make the optimal trade-off between resilience and efficiency.

The fitness and the multi-objective optimization approaches could be easily generalized to consider additional design objectives and constraints. For example, research on social networks indicates that resilience and efficiency might be just two of several design criteria that also include, e.g. "information-processing requirements," that impose additional constraints on network designs [5]. In the original context, "information-processing" refers to the need to have ties between individuals involved in a particular task, when the task has high complexity. Each individual might have a unique set of expertise into which all the other agents must tap directly. Generalizing from sociology, such "functional constraints" might considerably limit the flexibility in constructing resilient and efficient networks. For example, in the context of terrorism, this constraint significantly decreased the quality of attacks that could be successfully carried out in the post 9/11 security environment [73]. Such functional constraints could be addressed by looking at a palette of network designs which already incorporate such constraints. In engineering applications, such as infrastructure or communication networks, the financial cost of building the network is another key objective.

2.4.1 Properties of the Solution

The solution to the scalarized objective problem, (2.4) has a number of useful properties: Its fitness is continuous in the parameter r and changes predictably with other parameters: Notice that the claim is not about the continuity of fitness of a single configuration as a function of r but rather about the set of optimal solutions.

Proposition 2.1. *$f(r) = \max_{G\in\mathbb{G}} F(G,r)$ is Lipschitz-continuous for $r \in [0,1]$.*

Proof. The argument constructs a bound on the change in f in terms of the change in r. Consider an optimal configuration C_1 of a design for $r = r_1$ and let its fitness be $f_1 = F(C_1, r_1)$ (there is slight abuse of notation since C is a configuration, whose fitness is the average fitness of an ensemble of graphs).
Observation 1: Consider the fitness of C_1 at $r = r_2$. Because C_1 is fixed and the metrics are bounded ($0 \le R \le 1$ and $0 \le W \le 1$), the fitness change is bounded by the change in r:

$$\begin{aligned}|f_1 - F(C_1, r_2)| &= |r_1 R(C_1) + (1-r_1)W(C_1)\\ &\quad -r_2 R(C_1) - (1-r_2)W(C_1)|\\ &= |(r_1-r_2)R(C_1) - (r_1-r_2)W(C_1)|\\ &\le |r_1 - r_2|\,.\end{aligned}$$

Observation 2: Let C_2 be the optimal configuration for $r = r_2$ and let $f_2 = F(C_2, r_2)$. Since C_2 is optimal for $r = r_2$ it satisfies: $f_2 \geq f(C_1, r_2)$, and so $-f_2 \leq -F(C_1, r_2)$. It follows that $f_1 - f_2 \leq f_1 - F(C_1, r_2)$. Take the absolute value of the right hand side and apply Observation 1 to get the bound: $f_1 - f_2 \leq |r_1 - r_2|$.
Observation 3: Applying the argument of Observations 1&2 but reversing the roles of C_1 and C_2 implies that $f_2 - f_1 \leq |r_1 - r_2|$.

Observations 2 and 3 give $|f_1 - f_2| \leq |r_1 - r_2|$, proving the result.

Proposition 2.2. *Let $f(\tau)$ be the highest attainable fitness within a fixed network design $\mathbb{G}$, for cascade probability τ:*

$$f(\tau) = \max_{G \in \mathbb{G}} \left[\underbrace{rR(G,\tau) + (1-r)W(G)}_{F(G,\tau)} \right]$$

Then $f(\tau)$ is a non-increasing function of τ.

Proof. The proof relies on the simple claim that resilience of networks does not increase when τ increases [31]. The claim is equivalent to the result that for a given graph G increasing τ does not decrease the expected extent of cascades. The remainder is almost trivial: it is the claim that when the fitness of all the points on the space (all graphs) has been made smaller or kept the same (by increasing τ), the new maximum value would not be greater than in the old space.

The argument is easy to generalize. One could apply this method to the parameter g of attenuation, showing that fitness is non-increasing when attenuation is increased.

2.4.2 General Approaches to Large-Scale Networks

Our study did not involve solving the general optimization problem of finding the optimal network on n nodes in (2.4), but in some cases solving the general problem would be required. Clearly, the multi-objective problem and the scalarized model are hard: both are discrete optimization problems with non-linear objective functions. In general, solutions could be obtained using derivative-free optimization methods [18] and approximations such as [38,68]. Promising approaches also exist for finding the Pareto front [45,50,56]. Whether those methods are fast and accurate enough would depend on the definitions of $R(G)$, $W(G)$ and the set $\mathbb{G}$.

In small instances, it might also be possible to use the following approach based on bilevel stochastic integer programming. Given a specified network size (e.g. 180 nodes), one has integer decision variables $E_{ij} \in \{0,1\}$ for all $i \neq j$ where $i, j \in V$. The objective contains a stochastic term, $R(G)$ and a deterministic term, $W(G)$. The former is a linear function of the expected extent of percolation cascades. The cascade extents could be computed by generating stochastic starting points

$s \sim \text{Uniform}[V]$ and stochastic edge connectivity values $B_{ij} \sim \text{Bin}(\tau)$ for all i, j with $E_{ij} = 1$. Given the starting point and connectivities, the cascade extent could be found in each stochastic realization by solving the maximum flow problem: connect all nodes to a special target node t with edges of capacity $= 1.$ and assign capacities $|V|E_{ij}$ to all i, j pairs. On this network, the maximal $s-t$ flow numerically equals the set of nodes affected by the cascade that originated in s. The latter term, efficiency, could be computed by finding the all-pairs distances in the graph defined by $E_{ij} = 1$, by solving a linear program for every pair. It would be advantageous to use an efficiency function that depends linearly on distances, if possible, rather than the non-linear definition in (2.2) above.

2.4.3 Computational Implementation

To investigate the cascade-resilience of dark networks, we used computational methods described in this section. We considered networks on $n = 180$ nodes constructed through 6 simple designs, chosen both based on empirical findings (see e.g.[2, 13]) as well as the possibility of analytic tractability in some cases. When more data becomes available on dark networks, it will become possible to extract additional subgraphs with statistical validity.

Three of the designs are based on identical "cells": each cell is either (a) a clique (a complete graph), (b) a star (with a central node called "leader"), and (c) a cycle (nodes connected in a ring). Each of these have a single parameter, k – the number of nodes in the cell. Recent research suggests that under certain assumptions constructing networks from identical cells is optimal [27]. Let us also consider n-node graphs consisting of (d) randomly-connected cliques (sometimes termed "cavemen"), and (e) randomly-connected stars, in both cases according to probability p. Consider also (f) the simpler and well-studied Erdos–Renyi (ER) random graph with probability p (see figure in main text). By considering different structures for the cells we determine which of those structures provides the best performance.

The solution to the optimization problem is found by setting each of the parameters k (and when possible p) to various values. Each design D has "configurations" $C_1^D, C_2^D, \ldots$ each specifying the values of the parameters. Each configuration C_i^D is inputted to a program that generates an ensemble of $1-10$ networks, whose average performance provides an estimate of the fitness of C_i^D. The number of networks was ten for networks with parameter p because there is higher variability between instances. The coefficient of variation (CV) in the fitness of the sample networks was monitored to ensure that the average is a reliable measure of performance. Typically CV was <0.2 except near phase transitions of connectivity and percolation.

Optimization was performed using grid search. Alternative methods (e.g. Nelder–Mead) were considered but grid search was chosen despite its computational cost because it suffers no convergence problems even in the presence of noise (present due to variations in topology and contagion extent), and collects data useful for sensitivity analysis and multi-objective optimization. The sampling grid was as follows. In designs consisting of cells of size k, cell size was set to all integer values in $[1,180]$. If k did not divide 180, a cell of size $< k$ was added to ensure that the number of nodes in the graph is 180. The number of nodes is 180 because 180 is a highly-composite number and so it offers many networks of equally-sized cells. In general, normalization by n in the definitions of resilience and efficiency ensures that even when the number of nodes is tripled the effect of network size on fitness is very small for the above designs (around ± 0.05 in numerical experiments). In designs containing a parameter of connectivity p, it was set to all multiples of 0.05 in $[0,1]$, with some extra points added to better sample phase transitions. The grid search algorithm results are readily used to compute the Pareto frontier using the ε-balls method [45] ($\varepsilon = 0.01$).

The resilience metric is most easily computed by simulation where a node is selected at random to be "infected," and the simulation is run until all nodes are in states S or R, and none is in state I. In the simplest version of the SIR cascade model, which we adopt, each node in the graph can be in one of three states "susceptible," "infected" and "removed" designated S,I, and R, respectively (these names are borrowed from Epidemiology). Time is described in uniform discrete steps. A node in S state at time t stays in this state, unless a neighbor "infects" the node, causing it to move to state I at time $t+1$. Specifically, a node in state S at time t has probability τ of turning to I state at time $t+1$ for each adjacent node in state I at time t. Finally, a node in I state at time t always becomes R at time $t+1$. Once in state R, the node remains there for all future times. It is possible to consider an alternate model where the rate of transition $I \to R$ takes more than one time step, but adding this effect would mostly serve to increase the probability of transmission, which is already parametrized by τ [57,61].

A cascade/contagion that starts at a single node would run for up to n steps, but usually much fewer since typically $\tau < 1$ and/or the graph is not connected. To achieve good estimate of the average extent, the procedure was replicated 40 times, and then continued as long as necessary to achieve an error of under ± 0.5 node with a 95% confidence interval [46].

An analytic computation of the cascade extent metric was investigated. It is possible in theory because the contagion is a Markov process with states in the superset of the set of nodes, 3^n. Unfortunately, such a state space is impractically large. When G is a tree, then an analytic expression exists,[1] and it might be feasible when the treewidth is small [17, 57]. However, for many graph designs the tree approximation is not suitable. Another possible approach is to represent the contagion approximately as a system of differential equations which can be integrated numerically [39] . These possibilities were not pursued since the simulation approach could be applied to all graphs, while the errors of the analytic approaches are possibly quite large.

[1]Specifically, the mean contagion size is $1 + \frac{pG_0'(1)}{1-pG_1'(1)}$, where $G_0(x)$ generates the degree distribution and $G_1(x) = \frac{G_0'(x)}{G_0'(1)}$ generates the probability of arrival to a node [57].

2.5 Topological Cascade Resilience in the SIR-Reach Model

In this section and the rest of the paper, we will use the SIR-Reach model (R and W follow (2.1), (2.2)) The two models are attractive because they have real applications: Social Networks and Epidemiology. They have also been extensively explored by network scientists, which make them ideal as a case study in topological cascade resilience.

2.5.1 Optimal Network

The first set of experiments compares the designs against each other under different cascade risks (τ), Fig. 2.6. At each setting of τ, each design is optimized to its best configuration, i.e. the best cell size, and connectivity if applicable. The curves indicate the fitness of the optimal network in each design. Typically, at each τ the optimal network is different from the optimal network at another τ. Observe that within each design, as τ increases the fitness decreases – one cannot win when fighting cascades, only delay (see [30] for the proof). In certain applications, it

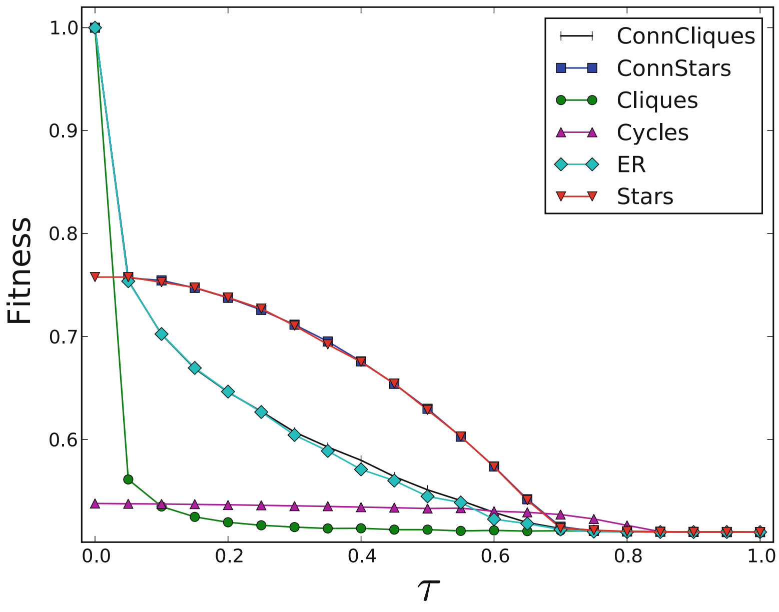


Fig. 2.6 Fitness at $r = 0.51$ of various network designs. The *Connected Stars* design is the best design at all cascade risks, τ. *Cliques and Connected Cliques* are competitive only for extreme ranges of τ. The superiority of *Connected Stars* over the *ER* (random graph) confirms the hypothesis that cells give fitness gains against cascades. The fitness of a design at each value of τ is defined as the fitness of the optimal configuration (network ensemble) within that design

is possible to invest in reducing the cascade propagation probability, τ. Then the curves in Fig. 2.6 could also be viewed as expressing the gain from efforts to reduce cascades by reducing τ and also adapting the network structure. If the slope is steep, then the gains are large.

Comparing designs to each other reveals that Connected Stars is superior to all others in fitness (Fig. 2.6). The design also outperforms any of the empirical networks in Fig. 2.2 in part because for each value of τ we selected the optimal network. The simpler Stars design is almost as fit, deteriorating only at extreme ranges of τ. The rankings of the designs are of course dependent on the parameter values, but not strongly (see [30] for the proof). Star-like designs are successful because the central node in a star acts as a cascade blocker while keeping the average distance in the star short ($\sim$2). Only for sufficiently low r, the Cliques, Connected Cliques and Connected Stars designs are superior to the Stars design. For such values of r efficiency is the dominant contributor to fitness. High weighting for efficiency benefits the former designs where efficiency can be 1 by constructing a fully connected (complete) graph. In the star design, efficiency is lower, reaching $\sim 1/2$ (when all nodes are placed in a single large star).

It has been long conjectured that cells provide dark networks with high resilience. Indeed, this is probably the reason why we found that dark networks have higher fitnesses than other networks. But cells also reduce the efficiency of a network since they isolate nodes from each other. To rigorously determine the net effect of cells, we compare the ER design (random graphs) to the Connected Stars design. ER is a strict subset of Connected Stars but only Connected Stars has cells. Therefore it is notable that Connected Stars has a higher fitness than ER, often significantly so. Indeed, cells must be the cause of higher fitness because cells are the only feature in Connected Stars that ER lacks.

2.5.2 *Properties of Optimal Networks*

Many properties of the optimal networks such as resilience, efficiency and edge density show rapid phase transitions as r is changed. For example, in the Cliques design when $r < 0.5$ the optimal network has high density that maximizes efficiency, whereas for $r > 0.5$ it is sparse and maximizes resilience (Fig. 2.7).

Intuition may suggest that the networks grow more sparse as cascade risk grows. Instead, the trend was non-monotonic (Fig. 2.7). For $\tau \gg 0$ and $r < 0.5$ Cliques, Connected Cliques, and Connected Stars became denser, instead of sparser, and for them the most sparse networks were formed in the intermediate values of τ where the optimal networks achieve both relatively high resilience and high efficiency. At higher τ values, when $r < 0.5$ it pays to sacrifice resilience because fitness is increased when efficiency is made larger through an equal or lesser sacrifice in resilience. The Stars design does not show a transition at $r = 0.5$ because it is hard to increase efficiency with this design.

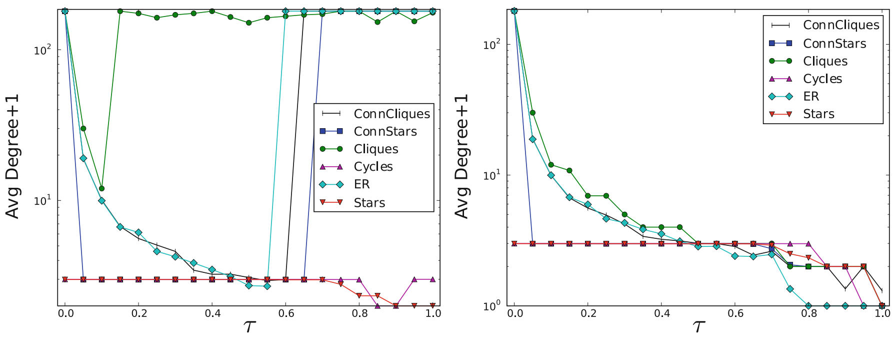


Fig. 2.7 Average degree in the optimal configuration of each design. At $r = 0.49$ (**a**) the optimization prefers networks that have high efficiency while at $r = 0.51$ (**b**) the preference is for resilience. In (**b**) the average degree diminishes monotonically to compensate for increasing cascade risk. In (**a**) most designs have a threshold τ at which they jump back to a completely–connected graph because structural cascade resilience becomes too expensive in terms of efficiency

2.5.3 Multi-objective Optimization

A complementary perspective on each design is found from its Pareto frontier of resilience and efficiency (Fig. 2.8). Typically a design is dominant in a part of the Resilience–Efficiency plane but not all of it. The Stars and Connected Stars designs can access most of the high resilience-low efficiency region. In contrast, the Cliques and Connected Cliques can make networks in the medium resilience-high efficiency regions.

The sharp phase transitions discussed earlier are seen clearly: along most of the frontiers, if we trace a point while decreasing resilience, there is a threshold at which a small sacrifice in resilience gives a major gain of efficiency. More generally, consider the points where the frontier is smooth. By taking two nearby networks on the frontier one can define a rate of change of efficiency with respect to resilience: $|\Delta W/\Delta R|$. The ratio can be used to optimize the network without using the parameter r. When $|\Delta W/\Delta R| \gg 1$ the network optimizer should choose to reduce to the resilience of the network in order to achieve great gains in efficiency; when $|\Delta W/\Delta R| \ll 1$ efficiency should be sacrificed to improve resilience.

2.6 Discussion

The analysis above considered both empirical networks and synthetic ones. The latter were constructed to achieve structural cascade resilience and efficiency. In contrast, in many empirical networks the structure emerges through an unplanned growth process or results from optimization to factors such as cost rather than blocking cascades. Without exception the synthetic networks showed higher fitness values

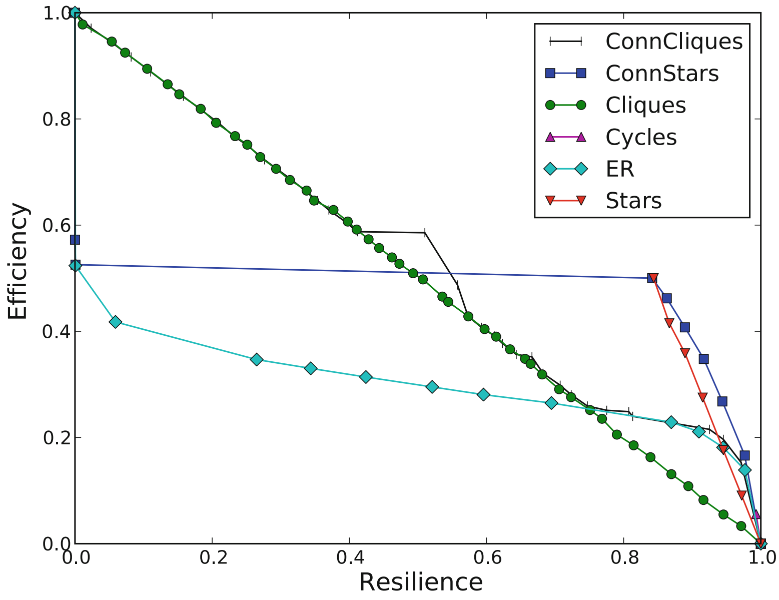


Fig. 2.8 The Pareto frontiers of various network designs ($\tau = 0.4$). The configurations of the *Connected Stars* design dominate over other designs when the network must achieve high resilience. However, designs based on *cliques* are dominant when high efficiency is required. Several designs show sharp transitions where at a small sacrifice of efficiency it is possible to achieve large increases in cascade resilience

despite the fact that they were based on very simple designs. This suggests that network optimization can significantly improve the fitness and cascade resilience of networks. It follows that an optimization process can be applied to design a variety of networks and to protect existing networks from cascades.

Many empirical networks also have power-law degree distributions [58]. Unfortunately, this feature significantly diminishes their cascade resilience: the resulting high-degree hubs make the networks extremely vulnerable to cascades once τ is slightly larger than 0 [20, 62].

In some successful synthetic networks, the density of edges increased when the cascade risk τ was high. This phenomenon has interesting parallels in non-violent social movements which are often organized openly rather than as secret underground cells even under conditions of severe state repression [70]. This openness greatly facilitates recruitment and advocacy, justifying the additional risk to the participants, just like the sacrifice of resilience to gain higher efficiency is justified under $r < 0.5$ conditions.

There are other important applications of this work, such as the design of power distribution systems. For power networks, the definition of resilience and efficiency will need to be changed. It would also be necessary to use much broader designs and optimization under design constraints such as cost. Furthermore, this work could also be adapted to domains of increasing concern such as financial credit networks, whose structure may make them vulnerable to bankruptcies [8, 36].

Acknowledgements This work has benefited from discussions with Aaron Clauset, Michael Genkin, Vadas Gintautas, Shane Henderson, Jason Johnson, and Roy Lindelauf, and anonymous reviewers. This paper extends the analysis originally published in [30]. Part of this work was funded by the Department of Energy at the Los Alamos National Laboratory (LA-UR 10-08349) under contract DE-AC52-06NA25396 through the Laboratory Directed Research and Development program, and by the Defense Threat Reduction Agency.

References

1. Albert, R., Jeong, H., Barabasi, A.L.: Error and attack tolerance of complex networks. Nature (London) **406**, 378–381 (2001)
2. Arquilla, J., Ronfeld, D.: Networks and Netwars: The Future of Terror, Crime, and Militancy. RAND Corporation, Santa Monica, CA (2001)
3. Ash, J., Newth, D.: Optimizing complex networks for resilience against cascading failure. Physica A: Statistical Mechanics and its Applications **380**, 673–683 (2007). DOI 10.1016/j.physa.2006.12.058
4. Baccara, M., Bar-Isaac, H.: How to organize crime. Review of Economic Studies **75**(4), 1039–1067 (2008)
5. Baker, W.E., Faulkner, R.R.: The social organization of conspiracy: Illegal networks in the heavy electrical equipment industry. American Sociological Review **58**(6), 837–860 (1993)
6. Ball, M.O.: Computing Network Reliability. Operations Research **27**(4), 823–838 (1979). DOI 10.1287/opre.27.4.823
7. Ball, M.O., Colbourn, C.J., Provan, J.S.: Network reliability. Tech. Rep. TR 1992-74, University of Maryland (1992)
8. Battiston, S., Gatti, D.D., Gallegati, M., Greenwald, B., Stiglitz, J.E.: Credit chains and bankruptcy propagation in production networks. Journal of Economic Dynamics and Control **31**, 2061–2084 (2007)
9. Briscoe, B., Odlyzko, A., Tilly, B.: Metcalfe's law is wrong. IEEE Spectrum (2006)
10. Bruneau, M., Chang, S.E., Eguchi, R.T., Lee, G.C., O'Rourke, T.D., Reinhorn, A.M., Shinozuka, M., Tierney, K., Wallace, W.A., von Winterfeldt, D.: A framework to quantitatively assess and enhance the seismic resilience of communities. Earthquake Spectra **19**(4), 733–752 (2003). DOI 10.1193/1.1623497
11. Buldyrev, S.V., Parshani, R., Paul, G., Stanley, H.E., Havlin, S.: Catastrophic cascade of failures in interdependent networks. Nature **464**(7291), 1025–1028 (2010). DOI 10.1038/nature08932
12. Callaway, D.S., Newman, M.E.J., Strogatz, S.H., Watts, D.J.: Network robustness and fragility: Percolation on random graphs. Phys. Rev. Lett. **85**(25), 5468–5471 (2000). DOI 10.1103/PhysRevLett.85.5468
13. Carley, K.M.: Destabilization of covert networks. Comput Math Organiz Theor **12**, 51–66 (2006)
14. Centola, D., Macy, M.: Complex contagions and the weakness of long ties. American J. Sociology **113**(3), 702–734 (2007)
15. Cohen, R., Erez, K., ben Avraham, D., Havlin, S.: Resilience of the internet to random breakdowns. Phys. Rev. Lett. **85**(21), 4626–4628 (2000). DOI 10.1103/PhysRevLett.85.4626
16. Colbourn, C.J.: Network resilience. SIAM Journal on Algebraic and Discrete Methods **8**(3), 404–409 (1987). DOI 10.1137/0608033
17. Colcombet, T.: On families of graphs having a decidable first order theory with reachability. In: P. Widmayer, S. Eidenbenz, F. Triguero, R. Morales, R. Conejo, M. Hennessy (eds.) Automata, Languages and Programming, *Lecture Notes in Computer Science*, vol. 2380, pp. 787–787. Springer Berlin / Heidelberg (2002)
18. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2009)

19. Cowie, J.H., Ogielski, A.T., Premore, B., Smith, E.A., Underwood, T.: Impact of the 2003 blackouts on internet communications: Preliminary report. Tech. rep., Renesys Corporation (2004). Www.renesys.com
20. Crepey, P., Alvarez, F.P., Barthelemy, M.: Epidemic variability in complex networks. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) **73**(4), 046131 (2006). DOI 10.1103/PhysRevE.73.046131
21. Crucitti, P., Latora, V., Marchiori, M.: Model for cascading failures in complex networks. Phys. Rev. E **69**(4), 045,104 (2004). DOI 10.1103/PhysRevE.69.045104
22. Dobson, I., Carreras, B.A., Lynch, V.E., Newman, D.E.: Complex systems analysis of series of blackouts: Cascading failure, critical points, and self-organization. Chaos: An Interdisciplinary Journal of Nonlinear Science **17**(2), 026103 (2007). DOI 10.1063/1.2737822
23. Doyle, J.C., Alderson, D.L., Li, L., Low, S., Roughan, M., Shalunov, S., Tanaka, R., Willinger, W.: The ”robust yet fragile” nature of the Internet. Proceedings of the National Academy of Sciences **102**(41), 14,497–14,502 (2005). DOI 10.1073/pnas.0501426102
24. Draief, M., Ganesh, A., Massoulié, L.: Thresholds for virus spread on networks. Annals of Applied Probability **18**(2), 359–378 (2008). DOI 10.1214/07-AAP470
25. Finbow, A.S., Hartnell, B.L.: On designing a network to defend against random attacks of radius two. Networks **19**(7), 771–792 (1989). DOI 10.1002/net.3230190704
26. Gleeson, J.P., Cahalane, D.J.: Seed size strongly affects cascades on random networks. Phys. Rev. E **75**(5), 056,103 (2007). DOI 10.1103/PhysRevE.75.056103
27. Goyal, S., Vigier, A.: Robust networks (2010). Working paper http://sticerd.lse.ac.uk/seminarpapers/et11032010.pdf
28. Guimerà, R., Danon, L., Díaz-Guilera, A., Giralt, F., Arenas, A.: Self-similar community structure in a network of human interactions. Phys. Rev. E **68**(6), 065,103 (2003). DOI 10.1103/PhysRevE.68.065103
29. Gunther, G., Hartnell, B.L.: On minimizing the effects of betrayals in resistance movements. In: Proceedings of the Eighth Manitoba conference on Numerical Mathematics and Computing, pp. 285–306 (1978)
30. Gutfraind, A.: Optimizing topological cascade resilience based on the structure of terrorist networks. PLoS ONE **5**(11), e13,448 (2010). DOI 10.1371/journal.pone.0013448
31. Gutfraind, A.: Monotonic and Non-Monotonic Epidemiological Models on Networks. http://arxiv.org/abs/1005.3470
32. Hartnell, B.L.: The optimum defense against random subversions in a network. In: Proceedings of the Tenth Southeast conference on Combinatorics Graph Theory and Computing, pp. 494–499 (1979)
33. Holme, P.: Efficient local strategies for vaccination and network attack. Europhys. Lett. **68**(6), 908–914 (2004)
34. Holme, P., Kim, B.J., Yoon, C.N., Han, S.K.: Attack vulnerability of complex networks. Phys. Rev. E **65**(5), 056,109 (2002). DOI 10.1103/PhysRevE.65.056109
35. Huang, W., Li, C.: Epidemic spreading in scale-free networks with community structure. J Stat Mech **P01014** (2007)
36. Iori, G., Masi, G.D., Precup, O.V., Gabbi, G., Caldarelli, G.: A network analysis of the italian overnight money market. Journal of Economic Dynamics and Control **32**, 259–278 (2008)
37. J. Leskovec, J.K., Faloutsos, C.: Graph Evolution: Densification and Shrinking Diameters. ACM Transactions on Knowledge Discovery from Data (ACM TKDD) **1**(1) (2007)
38. Johnson, J.K., Chertkov, M.: A majorization-minimization approach to design of power transmission networks. In: Proceedings of the 49th IEEE Conference on Decision and Control (CDC ’10) (2010)
39. Keeling, M.J.: The effects of local spatial structure on epidemiological invasions. Proc R Soc Lond B **266**, 859–867 (1999)
40. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD ’03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 137–146. ACM, New York, NY, USA (2003)
41. Klau, G.W., Weiskircher, R.: Robustness and resilience. In: Network Analysis, Lecture Notes in Computer Science 3418, pp. 417–437. Springer-Verlag (2005)

42. Krebs, V.E.: Mapping networks of terrorist cells. Connections **24**(3), 43–52 (2002)
43. Lai, Y.C., Motter, A., Nishikawa, T.: Attacks and cascades in complex networks. In: Complex Networks: Lecture Notes in Physics 650, pp. 299–310. Springer-Verlag (2004)
44. Latora, V., Marchiori, M.: Efficient behavior of small-world networks. Phys. Rev. Lett. **87**(19), 198,701 (2001). DOI 10.1103/PhysRevLett.87.198701
45. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. Evolutionary Computation **10**(3), 263–282 (2002). DOI 10.1162/106365602760234108. PMID: 12227996
46. Law, A., Kelton, W.D.: Simulation Modeling and Analysis, 3 edn. McGraw-Hill Higher Education, New York (1999)
47. Leskovec, J., Kleinberg, J.M., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: KDD, pp. 177–187 (2005)
48. Lindelauf, R.H., Borm, P.E., Hamers, H.: On Heterogeneous Covert Networks. SSRN eLibrary (2008)
49. Lindelauf, R.H., Borm, P.E., Hamers, H.: The Influence of Secrecy on the Communication Structure of Covert Networks. Social Networks **31**(2) (2009)
50. Marler, R., Arora, J.: Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization **26**, 369–395(27) (April 2004). DOI doi:10.1007/s00158-003-0368-6
51. Miksche, F.O.: Secret Forces, 1st edn. Faber and Faber, London, UK (1950)
52. Morselli, C., Petit, K., Giguere, C.: The Efficiency/Security Trade-off in Criminal Networks. Social Networks **29**(1), 143–153 (2007)
53. Motter, A.E.: Cascade control and defense in complex networks. Phys. Rev. Lett. **93**(9), 098,701 (2004). DOI 10.1103/PhysRevLett.93.098701
54. Motter, A.E., Lai, Y.C.: Cascade-based attacks on complex networks. Phys. Rev. E **66**(6), 065,102 (2002). DOI 10.1103/PhysRevE.66.065102
55. Motter, A.E., Nishikawa, T., Lai, Y.C.: Range-based attack on links in scale-free networks: Are long-range links responsible for the small-world phenomenon? Phys. Rev. E **66**(6), 065,103 (2002). DOI 10.1103/PhysRevE.66.065103
56. Mueller-Gritschneder, D., Graeb, H., Schlichtmann, U.: A successive approach to compute the bounded pareto front of practical multiobjective optimization problems. SIAM Journal on Optimization **20**(2), 915–934 (2009). DOI 10.1137/080729013
57. Newman, M.E.J.: Spread of epidemic disease on networks. Phys. Rev. E **66**(1), 016,128 (2002). DOI 10.1103/PhysRevE.66.016128
58. Newman, M.E.J.: The structure and function of complex networks. SIAM Review **45**(2), 167–256 (2003). DOI 10.1137/S003614450342480
59. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E **74**(3), 036,104 (2006). DOI 10.1103/PhysRevE.74.036104
60. Newman, M.E.J., Forrest, S., Balthrop, J.: Email networks and the spread of computer viruses. Phys. Rev. E **66**(3), 035,101 (2002). DOI 10.1103/PhysRevE.66.035101
61. Noël, P.A., Davoudi, B., Brunham, R.C., Dubé, L.J., Pourbohloul, B.: Time evolution of epidemic disease on finite and infinite networks. Phys. Rev. E **79**(2), 026,101 (2009). DOI 10.1103/PhysRevE.79.026101
62. Pastor-Sarorras, R., Vespignani, A.: Epidemic spreading in scale-free networks. Phys Rev Lett **86**(14), 3200–3203 (2001)
63. Phadke, A., Thorp, J.: Expose hidden failures to prevent cascading outages [in power systems]. Computer Applications in Power, IEEE **9**(3), 20 –23 (1996). DOI 10.1109/67.526849
64. Pourbohloul, B., Meyers, L., Skowronski, D., Krajden, M., Patrick, D., Brunham, R.: Modeling control strategies of respiratory pathogens. Emerg. Infect. Dis. **11**(8), 1246–56 (2005)
65. Raab, J., Milward, H.B.: Dark Networks as Problems. J Public Adm Res Theory **13**(4), 413–439 (2003). DOI 10.1093/jopart/mug029
66. Ripeanu, M., Foster, I., Iamnitchi, A.: Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. IEEE Internet Computing Journal **6**(1) (2002)

67. Rodriguez, J.: The march 11th terrorist network: In its weakness lies its strength (2004). Working Papers EPP-LEA, University of Barcelona
68. Ron, D., Safro, I., Brandt, A.: Relaxation-based coarsening and multiscale graph organization. SIAM Multiscale Modeling and Simulations (under revision) (2010). Preprint ANL/MCS-P1696-1009
69. Sageman, M.: Leaderless Jihad - Terror Networks in the Twenty-First Century. University of Pennsylvania Press, Philadelphia, PA (2008)
70. Sharp, G.: From dictatorship to democracy: A conceptual framework for liberation. The Albert Einstein Institution, East Boston, Massachusetts (2003)
71. U.S. Government: The 9/11 Commission Report. US Government Printing Office, Washington, DC (2007)
72. Watts, D.J.: A simple model of global cascades on random networks. Proceedings of the National Academy of Sciences of the United States of America **99**(9), 5766–5771 (2002). DOI 10.1073/pnas.082090499
73. Woo, G.: Mathematical Methods in Counterterrorism, chap. Intelligence Constraints on Terrorist Network Plots, pp. 205–214. Springer-Verlag (2009). Nasrullah Memon and Jonathan D. Farley and David L. Hicks and Torben Rosenorn, Eds.
74. Zawodny, J.: Internal organization problems and the sources of tensions of terrorist movements as catalysts of violence. Terrorism: An International Journal (continued as Studies in Conflict and Terrorism) **1**(3/4), 277–285 (1978)
75. Zhang, Y., Prica, M., Ilic, M., Tonguz, O.: Toward smarter current relays for power grids. In: Power Engineering Society General Meeting, 2006. IEEE, p. 8 (2006). DOI 10.1109/PES.2006.1709580

Chapter 3
Optimizing Synchronization, Flow, and Robustness in Weighted Complex Networks

G. Korniss, R. Huang, S. Sreenivasan, and B.K. Szymanski

Abstract Complex biological, social, and technological systems can be often modeled by weighted networks. The network topology, together with the distribution of available link or node capacity (represented by weights) and subject to cost constraints, strongly affect the dynamics or performance of the networks. Here, we investigate optimization in fundamental synchronization and flow problems where the weights are proportional to $(k_i k_j)^\beta$ with k_i and k_j being the degrees of the nodes connected by the edge. In the context of synchronization, these weights represent the allocation of limited resources (coupling strength), while in the associated random walk and current flow problems, they control the extent of hub avoidance, relevant in routing and search. In this chapter, we review fundamental connections between stochastic synchronization, random walks, and current flow, and we discuss optimization problems for these processes in the above weighted networks.

3.1 Introduction

Synchronization [1–6] and transport [7–11] phenomena are pervasive in natural and engineered complex interconnected systems with applications ranging from neurobiology and population dynamics to social, communication, and information networks. In the recent wave of research on complex networks [12–18], the focus has shifted from structure to various dynamical and stochastic processes on networks [19, 20], synchronization and transport are being one of them.

G. Korniss (✉) • R. Huang • S. Sreenivasan
Department of Physics and Social and Cognitive Networks Academic Research Center, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180–3590, USA
e-mail: korniss@rpi.edu; huangr3@gmail.com

S. Sreenivasan • B.K. Szymanski
Department of Computer Science and Social and Cognitive Networks Academic Research Center, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180–3590, USA
e-mail: sreens@rpi.edu; szymansk@cs.rpi.edu

The common question addressed by most studies within their specific context is how the collective response of locally-coupled entities is influenced by the underlying network topology.

Here, by network synchronization, we refer to the generic problem where individuals or agents attempt to locally coordinate their actions with their network neighbors or within some spatial neighborhood, in an attempt to improve global performance or reach global agreement [6, 21]. In the broader context, these problems are also referred to as consensus problems [6, 22, 23]. In this chapter, we will use the terms synchronization and coordination synonymously. Classic examples for coordination phenomena are animal flocking [24–26] and cooperative control of vehicle formation [27], where individual animals or units are adjusting their position, speed, and headings (the relevant local state variables) based on the state of their neighborhood, potentially leading to tight formations. Fundamental synchronization problems have also numerous applications to neurobiology [28–32], population dynamics [33, 34], and load balancing and task allocation problems in distributed computing [21,35–39].

Research on flow optimization in networks has been around since at least the first data sets on transportation networks became available (for a brief historical review, see [11,40]). Perhaps, among the first ones was a study on transportation planning on the Soviet railway network, as early as in 1930 [41], followed by others in the 1940s [42–44]. Flow optimization and network interdiction problems also attracted significant interest during the Cold War years [45,46] and have been a main thrust in operations research since [7,47,48].

The increasing availability of data on real-life complex biological, information, social, and infrastructure networks, and the emerging novel type of network structures have triggered a recent wave on fundamental research on transport and flow in networks [49–83]. Connections between random walks and resistor networks have been discussed in detail in several works [84–86]. Furthermore, we have recently explored fundamental connections and relations (governed by the same underlying network Laplacian) between stochastic synchronization problems and resistor networks, current flow, and random walks [10, 87]. In this Chapter, in parallel with reviewing synchronization phenomena in noisy environments, we will discuss some natural and fundamental connections with idealized transport and flow problems on complex networks, in particular, connections with some simplified local and global routing and search schemes [67,68,72].

The ultimate challenge in network optimization (of synchronization and flow) is when both the network structure and the link qualities (represented by weighted links) can change or evolve [8,67], subject to cost constraints. Here, we review and discuss a simpler set of problems, where the network structure is fixed but the link weights (or coupling strengths) can be allocated. In particular, we consider a specific and symmetric form of the weights on uncorrelated scale-free (SF) networks, being proportional to $(k_i k_j)^\beta$ where k_i and k_j are the degrees of the nodes connected by the link [10, 88–91]. The above general form has been suggested by empirical studies of metabolic [50] and airline transportation networks [51]. We discuss the effects of such a weighting scheme in our synchronization and flow problems. Then the task becomes maximizing the synchronization efficiency, throughput, or robustness as a function of β.

The setup of this chapter is as follows. In Sect. 3.2 we review optimization of synchronization in a noisy environment [10]. In Sects. 3.3 and 3.4, we present results for optimization of resistor networks and random walks, respectively, together with reviewing fundamental connections between the relevant observables in synchronization, resistor networks, and random walks. In Sect. 3.5 we discuss current-flow betweenness and optimization of throughput in weighted complex networks [92]. In Sect. 3.6 we present results on shortest-path betweenness, cascading failures, and cascade control in weighted complex networks.

3.2 Synchronization in a Noisy Environment in Weighted Networks

A large number of studies investigated the Kuramoto model of coupled oscillators [4, 93], naturally generalized to complex networks [94–96]. The common feature of the findings is the spontaneous emergence of order (synchronous phase) on complex networks, qualitatively similar to that observed on fully-connected networks (also referred to as complete graphs), in contrast to regular networks in low dimensions. Another large group of studies addressed synchronization in coupled nonlinear dynamical systems (e.g., chaotic oscillators) [3] on small-world (SW) [97] and scale-free (SF) [88, 98–101] networks. The analysis of synchronization in the latter models can be carried out by linearization about the synchronous state and using the framework of the master stability function [102]. In turn, the technical challenge of the problem is reduced to the diagonalization of the Laplacian on the respective network, and calculating or estimating the eigenratio [97] (the ratio of the largest and the smallest non-zero eigenvalue of the network Laplacian), a characteristic measure of synchronizability (smaller eigenratios imply better synchronizability). Along these lines, a number of recent studies considered not only complex, possibly heterogeneous, interaction topologies between the nodes, but also weighted (heterogeneities in the strength of the couplings)[49, 88, 99, 100] and directed networks [103–105].

In a more general setting of synchronization problems, the collective behavior/response of the system is obviously strongly influenced by the nonlinearities, the coupling/interaction topology, the weights/strength of the (possibly directed) links, and the presence and the type of noise [3, 101]. Here, we study synchronization in weighted complex networks with linear coupling in the presence of delta-correlated white noise. Despite its simple formulation, this problem captures the essential features of fundamental stochastic synchronization, consensus, and coordination problems with application ranging from coordination and load balancing causally-constrained queuing networks [106, 107] to e-commerce-based services facilitated by interconnected servers [108], and certain distributed-computing schemes on computer networks [21,36–39]. This simplified problem is the Edwards–Wilkinson (EW) process [109] on the respective network [10, 87, 110–115], and is described by the Langevin equation

$$\partial_t h_i = -\sum_{j=1}^{N} C_{ij}(h_i - h_j) + \eta_i(t), \tag{3.1}$$

where $h_i(t)$ is the general stochastic field variable on a node (such as fluctuations in the task-completion landscape in certain distributed parallel schemes on computer networks [21, 111, 112]) and $\eta_i(t)$ is a delta-correlated noise with zero mean and variance $\langle\eta_i(t)\eta_j(t')\rangle = 2\delta_{ij}\delta(t-t')$. Here, $C_{ij} = C_{ji} > 0$ is the symmetric coupling strength between the nodes i and j ($C_{ii} \equiv 0$). Note that without the noise term, the above equation is also referred to as the consensus problem [6, 22, 23] on the respective network (in the sense of networked agents trying to reach an agreement, balance, or coordination regarding a certain quantity of interest). Defining the network Laplacian,

$$\Gamma_{ij} \equiv \delta_{ij}C_i - C_{ij}, \tag{3.2}$$

where $C_i \equiv \sum_l C_{il}$, we can rewrite (3.1)

$$\partial_t h_i = -\sum_{j=1}^{N} \Gamma_{ij} h_j + \eta_i(t). \tag{3.3}$$

For the steady-state equal-time two-point correlation function one finds

$$G_{ij} \equiv \langle(h_i - \bar{h})(h_j - \bar{h})\rangle = \hat{\Gamma}^{-1}_{ij} = \sum_{k=1}^{N-1} \frac{1}{\lambda_k}\psi_{ki}\psi_{kj}, \tag{3.4}$$

where $\bar{h} = (1/N)\sum_{i=1}^{N} h_i$ and $\langle\ldots\rangle$ denotes an ensemble average over the noise in (3.3). Here, $\hat{\Gamma}^{-1}$ denotes the inverse of Γ in the space orthogonal to the zero mode. Also, $\{\psi_{ki}\}_{i=1}^{N}$ and λ_k, $k = 0, 1, \ldots, N-1$, denote the kth normalized eigenvectors and the corresponding eigenvalues, respectively. The $k = 0$ index is reserved for the zero mode of the Laplacian on the network: all components of this eigenvector are identical and $\lambda_0 = 0$. The last form in (3.4) (the spectral decomposition of $\hat{\Gamma}^{-1}$) can be used to directly employ the results of exact numerical diagonalization.

For the EW process on any network, the natural observable is the steady-state width or spread of the synchronization landscape [87, 111, 112, 115–117]

$$\langle w^2\rangle \equiv \left\langle \frac{1}{N}\sum_{i=1}^{N}(h_i - \bar{h})^2 \right\rangle = \frac{1}{N}\sum_{i=1}^{N} G_{ii} = \frac{1}{N}\sum_{k=1}^{N-1}\frac{1}{\lambda_k}. \tag{3.5}$$

The above observable is typically self-averaging (confirmed by numerics), i.e., the width $\langle w^2\rangle$ for a sufficiently large, single network realization approaches the width averaged over the network ensemble. A network is said to be synchronizable if the width has a finite steady-state value; the smaller the width, the better the synchronization. Finite and connected (single component) networks are always synchronizable. In the limit of infinite network size, however, network ensembles with a vanishing (Laplacian) spectral gap may become unsynchronizable, depending on the details of the small-λ behavior of the density of eigenvalues [5, 21].

The focus of this section is to optimize synchronization (i.e., minimize the width) on (a) weighted uncorrelated networks with SF degree distribution, (b) subject to a fixed cost. In the context of this work, we define the total cost C_{tot} simply to equal to the sum of weights over all edges in the network

$$\sum_{i<j} C_{ij} = \frac{1}{2}\sum_{i,j} C_{ij} = C_{\text{tot}}. \tag{3.6}$$

The elements of the coupling matrix C_{ij} can be expressed in terms of the network's adjacency matrix A_{ij} and the assigned weights W_{ij} connecting node i and j as $C_{ij} = W_{ij}A_{ij}$. Here, we consider networks where the weights are symmetric and proportional to a power of the degrees of the two nodes connected by the link, $W_{ij} \propto (k_i k_j)^{\beta}$. We choose our cost constraint to be such that it is equal to that of the unweighted network, where each link is of unit strength.

$$\sum_{i,j} C_{ij} = 2C_{\text{tot}} = \sum_{i,j} A_{ij} = N\bar{k}, \tag{3.7}$$

where $\bar{k} = \sum_i k_i/N = \sum_{i,j} A_{ij}/N$ is the mean degree of the graph, i.e., the average cost per edge is fixed. Thus, the question we ask, is how to allocate the strength of the links in networks with heterogeneous degree distributions with a fixed total cost in order to optimize synchronization. That is, the task is to determine the value of β which minimizes the width (3.5), subject to the constraint (3.7).

Combining the form of the weights, $W_{ij} \propto (k_i k_j)^{\beta}$, and the constraint (3.7) one can immediately write for the coupling strength between nodes i and j

$$C_{ij} = N\bar{k}\frac{A_{ij}(k_i k_j)^{\beta}}{\sum_{l,n} A_{ln}(k_l k_n)^{\beta}} \tag{3.8}$$

From the above it is clear that the distribution of the weights is controlled by a single parameter β, while the total cost is fixed, $C_{\text{tot}} = N\bar{k}/2$.

Before tackling the above optimization problem for the restricted set of heterogeneous networks and the specific form of weights, it is useful to determine the minimum attainable value of the width of the EW synchronization problem in any network with symmetric couplings. This value will serve as a "baseline" reference for our problem. In Appendix 1, we show that this absolute minimum value of the width is

$$\langle w^2 \rangle_{\text{min}} = \frac{(N-1)^2}{2NC_{\text{tot}}} \tag{3.9}$$

and can be realized by the fully connected network.

If one now considers the synchronization problem on any network with N nodes, with average degree $\overline{k}$ and with total cost $C_{\text{tot}} = N\overline{k}/2$ to be optimized in some fashion [e.g., with respect to a single parameter β, (3.8), the above result provides an absolute lower bound for the optimal width

$$\langle w^2(\beta)\rangle_{\min} \geq \frac{(N-1)^2}{N^2}\frac{1}{\overline{k}} \simeq \frac{1}{\overline{k}}. \tag{3.10}$$

3.2.1 Mean-Field Approximation on Uncorrelated SF Networks

First, we approximate the equations of motion (3.1) by replacing the local weighted average field $(1/C_i)\sum_j C_{ij}h_j$ with the global average $\overline{h}$ (the mean–height)

$$\begin{aligned}\partial_t h_i &= -\sum_{j=1}^{N} C_{ij}(h_i - h_j) + \eta_i(t) = -C_i\left(h_i - \frac{\sum_j C_{ij}h_j}{C_i}\right) + \eta_i(t)\\ &\approx -C_i\left(h_i - \overline{h}\right) + \eta_i(t).\end{aligned} \tag{3.11}$$

Note that $C_i \equiv \sum_j C_{ij}$ is the weighted degree. As can be directly seen by summing up (3.1) over all nodes, the mean height $\overline{h}$ performs a simple random walk with noise intensity $\mathscr{O}(1/N)$. Thus, in the mean-field (MF) approximation (see details in Appendix 2), in the asymptotic large-N limit, fluctuations *about the mean* decouple and reach a stationary distribution with variance

$$\left\langle (h_i - \bar{h})^2 \right\rangle \approx 1/C_i, \tag{3.12}$$

yielding

$$\langle w^2 \rangle = \frac{1}{N}\sum_{i=1}^{N}\left\langle (h_i - \bar{h})^2\right\rangle \approx \frac{1}{N}\sum_i \frac{1}{C_i}. \tag{3.13}$$

Now we consider uncorrelated weighted SF networks, with a degree distribution

$$P(k) = (\gamma - 1)m^{\gamma-1}k^{-\gamma}, \tag{3.14}$$

where m is the minimum degree in the network and $2 < \gamma \leq 3$. The average and the minimum degree are related through $\langle k\rangle = m(\gamma-1)/(\gamma-2)$. Using the approximation for the weighted degree $C(k)$ of a node with degree k in uncorrelated (UC) weighted SF graphs (see Appendix 3),

$$C(k) \approx \frac{\gamma - 2 - \beta}{\gamma - 2}\frac{k^{\beta+1}}{m^{\beta}}, \tag{3.15}$$

and assuming self-averaging for large enough networks, one obtains for the width of the synchronization landscape

$$\langle w^2(\beta)\rangle \approx \frac{1}{N}\sum_i \frac{1}{C_i} \approx \int_m^\infty \mathrm{d}kP(k)\frac{1}{C(k)} = \frac{1}{\langle k\rangle}\frac{(\gamma-1)^2}{(\gamma-2-\beta)(\gamma+\beta)}, \tag{3.16}$$

where using infinity as the upper limit is justified for $\gamma+\beta > 0$. Elementary analysis yields the main features of the above expression for the average width:

1. $\langle w^2(\beta)\rangle$ is minimum at $\beta = \beta^* = -1$, *independent* of the value of γ.
2. $\langle w^2\rangle_{\min} = \langle w^2(\beta^*)\rangle = 1/\langle k\rangle$

The above approximate result is consistent with using infinity as the upper limit in all integrals, in that the optimal value $\beta^* = -1$ falls inside the interval $-\gamma < \beta < \gamma - 2$ for $2 < \gamma \leq 3$. Interestingly, one can also observe that in this approximation, the minimal value of the width is equal to that of the global optimum (3.10), realized by the fully-connected network of the same cost $N\langle k\rangle/2$, i.e. with identical links of strength $\langle k\rangle/(N-1)$.

We emphasize that in obtaining the above result (3.16) we employed two very strong and distinct assumptions/approximations: (a) for the dynamics on the network, we neglected correlations (in a MF fashion) between the local field variables and approximated the local height fluctuations by (3.12); (b) we assumed that the network has no degree–degree correlations between nodes which are connected (UC), so that the "weighted degree" of a node with degree k, $C(k)$ can be approximated with (3.15) for networks with $m \gg 1$.

3.2.2 *Numerical Results*

For comparison with the above mean-field results, we considered Barabási–Albert (BA) SF networks [13, 14], "grown" to N nodes,[1]where $P(k) = 2m^2/k^3$, i.e., $\gamma = 3$. While growing networks, in general, are not uncorrelated, degree–degree correlations are anomalously (marginally) weak for the BA network [18, 118].

We have performed exact numerical diagonalization and employed (3.4) to find the local height fluctuations and (3.5) to obtain the width for a given network realization. We carried out the above procedure for 10–100 independent network realizations. Finite-size effects (except for the $m = 1$ BA tree network) are very weak for $-2 < \beta < 0$; the width essentially becomes independent of the system size in this interval. Figure 3.1 displays result for the local height fluctuations as a

[1]For the BA scale-free model [13] (growth and preferential attachment), each new node is connected to the network with m links, resulting in $\langle k\rangle \simeq 2m$ in the large-N limit. Here, we employed a fully-connected initial cluster of $m+1$ nodes.

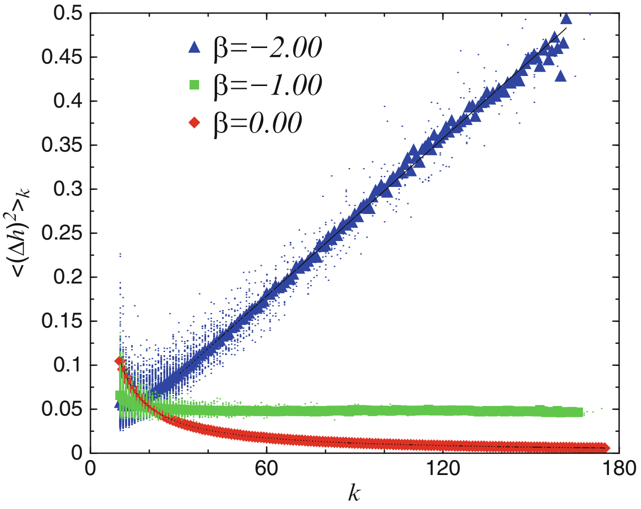


Fig. 3.1 Height fluctuations as a function of the degree of the nodes for $N = 1{,}000$, $\langle k \rangle = 20$, and for $\beta = -2.00$, $\beta = -1.00$, and $\beta = 0.00$ (*from top to bottom*). Data, represented by *filled symbols*, are averaged over all nodes with degree k. Scatter plot (*dots*) for individual nodes is also shown from ten network realizations. *Solid lines* correspond to the MF+UC scaling $\langle(\Delta h)^2\rangle_k \sim k^{-(\beta+1)}$

function of the degree of the node. We show both the fluctuations averaged over all nodes with degree k and the scattered data for individual nodes. One can observe that our approximate results for the scaling with the degree [combining (3.12) and (3.58), $\left\langle (h_i - \bar{h})^2 \right\rangle \approx 1/C_i \sim k_i^{-(\beta+1)}$, work very well, except for very low degrees. The special case $\beta = 0$, is exceptionally good, since here $C_i = \sum_j A_{ij} = k_i$ exactly, and the only approximation is (3.12).

In Fig. 3.2, we show our numerical results for the width and compare it with the approximate (MF+UC) results (3.16). The divergence of the approximate result (3.16) at $\beta = -3$ and $\beta = 1$ is the artifact of using infinity as the upper limit in the integrals performed in our approximations. The results for the width clearly indicate the existence of a minimum at a value of β^* somewhat greater than -1. Further analysis reveals [10] that as the minimum degree m is increased, the optimal β approaches -1 from above. This is not surprising, since in the limit of $m \gg 1$ (large minimum degree), both the MF and the UC part of our approximations are expected to work progressively better. For $\beta = 0$, our approximation (3.16) is within 8%, 4%, and 1% of the results extracted from exact numerical diagonalization through (3.5), for $m = 10$, $m = 20$, and $m = 100$, respectively [10]. For $\beta = -1$, it is within 15%, 7%, and 3% of the numerical results for $m = 10$, $m = 20$, and $m = 100$, respectively [10]. Thus, our approximation works reasonably well for large uncorrelated SF networks with sufficiently large minimum (and consequently, average) degree, i.e., in the $1 \ll m \ll N$ limit. Although for sparse networks with small average degree the MF+UC approximation fails to locate the minimum and

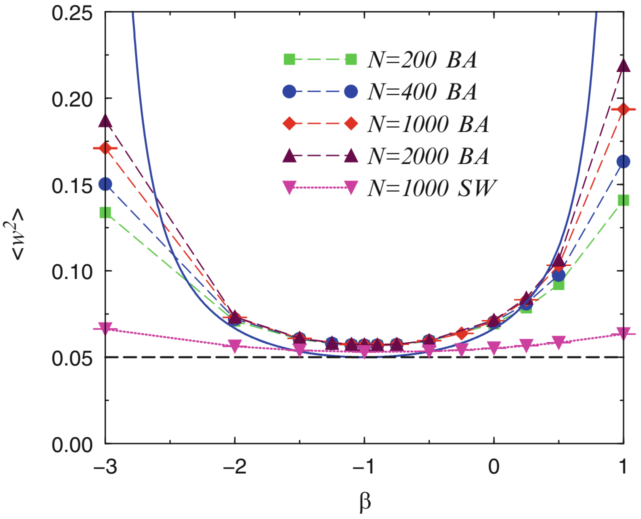


Fig. 3.2 Steady-state width of the EW synchronization landscape as a function of the weighting parameter β for the BA networks with $m = 10$ ($\langle k\rangle \simeq 2m = 20$) for various network sizes. The *solid curve* is the approximate (MF+UC) result (3.16). For comparison, numerical results for a SW networks with $N = 1{,}000$ and with the same degree is also shown. The *horizontal dashed line* indicates the absolute lower bound (3.10), as achieved by the fully connected network with the same cost $N\langle k\rangle/2$

the value of the width precisely; nevertheless, it provides insight for an efficient optimization of the global performance of weighted heterogeneous networks with a single parameter β, as opposed to a computationally prohibitive exhaustive search. For a detailed quantitative analysis of the error of the MF+UC approximation in the context of the closely related random walks on weighted SF networks (Sect. 3.4) see [91].

The above optimal link-strength allocation at around the value $\beta^* \approx -1$ seems to be present in all random networks where the degree distribution is different from a delta-function. For example, in SW networks,[2]although the degree distribution has an exponential tail, $\langle w^2\rangle$ also exhibits a minimum, but the effect is much weaker, as shown in Fig. 3.2. Further, a point worthwhile to mention, a SW network with the same number of nodes and the same average degree (corresponding to the same cost) always "outperforms" its SF counterpart (in terms of minimizing the width). The difference between their performance is smallest around the optimal value, where both are very close to that of the lowest possible value, realized by the FC network of the same cost.

[2]Here, we constructed SW networks by *adding* random links [111, 119, 120] on top of a regular ring with two nearest neighbors. The density of random links per node is p, resulting in an average degree $\langle k\rangle = 2 + p$.

3.3 Weighted Resistor Networks

Resistor networks have been widely studied since the 1970s as models for conductivity problems and classical transport in disordered media [121, 122]. Amidst the emerging research on complex networks, resistor networks have been employed to study and explore community structures in social networks [123–126] and centrality measures in information networks [127]. Also, electrical networks with directed links (corresponding to diodes) have been used to propose novel page-ranking methods for search engines on the World-Wide-Web [128].

Most recently, simple resistor networks were utilized to study transport efficiency in SF [79, 80] and SW networks [87]. The work by López et al. [80] revealed that in SF networks [13, 14] anomalous transport properties can emerge, displayed by the power-law tail of distribution of the network conductance. Now, we consider weighted resistor networks subject to a fixed total cost (the cost of each link is associated with its conductance). As we have shown [10,87] the relevant observables in the EW synchronization problem and in (Ohmic) resistor networks are inherently related through the spectrum of the network Laplacian. Consider an arbitrary (connected) network where C_{ij} is the conductance of the link between node i and j, with a current I entering (leaving) the network at node s (t). Kirchhoff's and Ohm's laws provide the relationships between the stationary currents and voltages [87, 129]

$$\sum_j C_{ij}(V_i - V_j) = I(\delta_{is} - \delta_{it}), \tag{3.17}$$

or equivalently,

$$\sum_j \Gamma_{ij} V_j = I(\delta_{is} - \delta_{it}), \tag{3.18}$$

where Γ_{ij} is the network Laplacian, as defined in the context of the EW process (3.2). Introducing the voltages measured from the mean at each node, $\hat{V}_i = V_i - \bar{V}$, where $\bar{V} = (1/N)\sum_{i=1}^{N} V_i$, one obtains [87]

$$\hat{V}_i = I(G_{is} - G_{it}). \tag{3.19}$$

Here, G is the same network propagator discussed in the context of the EW process, i.e. the inverse (3.4) of the network Laplacian (3.2) in the space orthogonal to the zero mode. Applying (3.19) to nodes s and t, where the voltage drop between these nodes is $V_{st} = \hat{V}_s - \hat{V}_t$, one immediately obtains the effective two-point resistance of the network between nodes s and t [87, 129],

$$R_{st} \equiv \frac{V_{st}}{I} = G_{ss} + G_{tt} - 2G_{st} = \sum_{k=1}^{N-1} \frac{1}{\lambda_k}(\psi_{ks}^2 + \psi_{kt}^2 - 2\psi_{ks}\psi_{kt}). \tag{3.20}$$

The spectral decomposition in (3.20) is, again, useful to employ the results of exact numerical diagonalization. Comparing (3.4) and (3.20), one can see that the two-point resistance of a network between node s and t is the same as the steady-state *height-difference* correlation function of the EW process on the network [87],

$$\langle (h_s - h_t)^2 \rangle = \langle [(h_s - \bar{h}) - (h_t - \bar{h})]^2 \rangle = G_{ss} + G_{tt} - 2G_{st} = R_{st}\,. \tag{3.21}$$

For example, using the above relationship and then employing the MF+UC approximation,[3]one can immediately obtain the scaling of the typical value of the effective two-point resistance in weighted resistance networks, between two nodes with degrees k_s and k_t,

$$R_{st} \simeq G_{ss} + G_{tt} \sim \left[k_s^{-(1+\beta)} + k_t^{-(1+\beta)} \right] = \frac{k_s^{1+\beta} + k_t^{1+\beta}}{(k_s k_t)^{1+\beta}}\,. \tag{3.22}$$

A global observable, measuring transport efficiency, analogous to the width of the synchronization landscape, is the average two-point resistance [80,87] (averaged over all pairs of nodes, for a given network realization). Using (3.21) and exploiting the basic properties of the Green's function, one finds

$$\bar{R} \equiv \frac{2}{N(N-1)} \sum_{s<t} R_{st} = \frac{1}{N(N-1)} \sum_{s \neq t} R_{st} = \frac{N}{N-1} 2\langle w^2 \rangle \simeq 2\langle w^2 \rangle, \tag{3.23}$$

i.e., in the asymptotic large system-size limit the average system resistance of a given network is twice the steady-state width of the EW process on the same network. Note that the above relationships, (3.21) and (3.23), are exact and valid for any graph.

The corresponding optimization problem for resistor networks then reads as follows: For a fixed total cost, $C_{\text{tot}} = \sum_{i<j} C_{ij} = N\langle k \rangle / 2$, where the link conductances are weighted according to (3.8), what is the value of β which minimizes the average system resistance $\overline{R}(\beta)$? Based on the above relationship between the average system resistance and the steady-state width of the EW process on the same graph (3.23), the answer is the same as was discussed in Sect. 3.2 (3.16): $\beta^* = -1$ and $\overline{R}_{\min} = 2N/[(N-1)\langle k \rangle] \simeq 2/\langle k \rangle$ in the mean-field approximation on uncorrelated random SF networks. Numerical results for $\overline{R}(\beta)$ are also provided for "free" as $\bar{R}(\beta) \simeq 2\langle w^2(\beta) \rangle$, by virtue of the connection (3.23) [Fig. 3.2].

[3]In the context of resistor networks, while there are no "fields," we carry over the terminology "mean-field" (MF) from the associated EW synchronization problem. In terms of the network propagator, the assumptions of the MF approximation can be summarized as $G_{st} \ll G_{ss}$ for all $s \neq t$, and $G_{ss} \simeq 1/C_s$.

3.3.1 Transport Optimization for Heterogeneous Source/Target Frequencies

As suggested by Lopez et al. [80], the effective (electrical) conductance provides a powerful measure to characterize transport in complex networks. This observable, strongly influenced by the number of disjoint (and possibly weighted) paths between a source and a target, is also closely related to the max-flow problem in networks [7, 11, 40, 63, 81]. The effective two-point conductance is the inverse of the effective two point resistance (3.20), $g_{st} = 1/R_{st}$. If each node is equally likely to be a target or a source, a simple average over all source and target pairs provides the average system conductance, $\bar{g} = \sum_{s \neq t} g_{st}/N(N-1)$. In real systems, however, nodes are not created equal; their relative frequency to be a source or target can greatly vary. In the simplest phenomenological model, we assume that nodes are sources or targets with a frequency proportional k_i^{ρ} $(\rho \geq 0)$ [65, 80]. Also, as previously, we allow the edges (conductivities) to be weighted, controlled by the parameter β according to (3.8), subject to a fixed total edge cost (3.7). Then, naturally, the relevant global measure is the appropriately weighted system conductance

$$\bar{g}(\beta) = \frac{\sum_{s \neq t} (k_s k_t)^{\rho} g_{st}(\beta)}{\sum_{s \neq t} (k_s k_t)^{\rho}}. \tag{3.24}$$

Then, we consider *optimizing the allocation limited resources* in the above simplified transport problem. That is, for a given source/target distribution controlled by ρ, what is the value of β which minimizes the system conductance $\bar{g}(\beta)$?

In Fig. 3.3, we show numerical results for BA scale-free networks. When the source/target profile is uniform ($\rho = 0$), the system conductance exhibits a maximum at around $\beta \approx -1$ (in synch with the system resistance exhibiting a minimum around the same β, Fig. 3.2). For increasing positive values of ρ, the optimal value of β shifts to the right; the location of the maximum of the $\bar{g}(\beta)$ curve for a given ρ quantifies the extent to which resources should be allocated around hubs (or away from hubs) for optimal global performance.

Figure 3.3 also indicates that the conductance curves for all ρ intersect at around $\beta \approx -1$. Indeed, our previous approximation (3.22) predicts that at this point the effective two-point conductance $g_{st} = 1/R_{st}$ becomes independent of the degree of the source and target nodes, hence the system conductance (3.24) become ρ-invariant.

In Fig. 3.3, we also plot the same system conductance cure for SW networks with the same network size and average degree for two values of ρ. For $\rho = 0$ (uniform source/target profile), a SW graph (with a close-to-homogeneous degree distribution) outperforms its BA SF counterpart (with heterogeneous degree distribution) of the same cost for every β. For strongly heterogeneous source/target frequencies ($\rho = 1$), the performance of a SW network is better for $\beta < -1$ and $\beta > 2$, while the BA SF network performs better in the $-1 < \beta < 2$ interval.

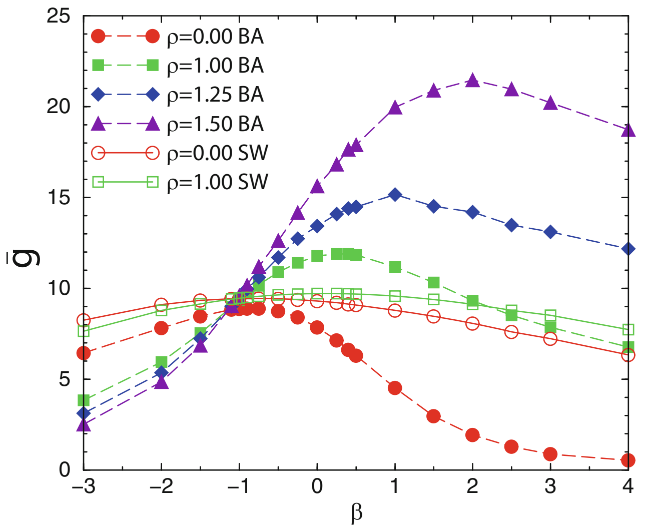


Fig. 3.3 System conductance vs. the edge weight parameter β for different source/target distributions controlled by ρ for BA networks with $m = 10$ ($\langle k \rangle \simeq 2m = 20$) and $N = 400$ (*solid symbols*). For comparison, numerical results for a SW networks with the same network size and average degree is also shown for two ρ values (*open symbols*)

3.4 Random Walks in Weighted Networks

Investigating random walks (RW) on networks and resistor networks can provide invaluable insights into fundamental properties and characteristics of transport and flow on networks [10, 54–56, 80, 84, 87, 91, 130, 131]. In these models, with direct application to search, routing, and information retrieval on networks [132, 133], the connection between network structure and function becomes explicit, so one can address the problems of designing network structures to minimize delivery times, or for a fixed structure, allocating resources (queuing capacity) to minimize load and delays [10, 70, 90].

Here, we consider weights $\{C_{ij}\}$ employed in the previous sections and define a discrete-time random walk (RW) with the transition probabilities [84]

$$P_{ij} \equiv \frac{C_{ij}}{C_i} \tag{3.25}$$

(recall that $C_i = \sum_l C_{il}$ is the weighted degree). P_{ij} is the probability that the walker currently at node i will hop to node j in the next step. Note that because of the construction of the transition probabilities (being a normalized ratio), the issue of cost constraint disappears from the problem. That is, any normalization prefactor associated with the conserved cost [as in (3.8)] cancels out, and the only relevant information is $C_{ij} \propto A_{ij}(k_ik_j)^{\beta}$, yielding

$$P_{ij} = \frac{C_{ij}}{C_i} = \frac{A_{ij}(k_ik_j)^{\beta}}{\sum_l A_{il}(k_ik_l)^{\beta}} = \frac{A_{ij}k_j^{\beta}}{\sum_l A_{il}k_l^{\beta}}. \tag{3.26}$$

Then the results are invariant for any normalization/constraint, so for convenience one can use the normalized form of the C_{ij} coefficients as given in (3.8). As is clear from the above RW transition probabilities, the parameter β controls to what extent "hubs" should be avoided.

Having a random walker starting at an arbitrary source node s, tasked to arrive at an arbitrary target node t, the above weighted RW model can be associated with a simple *local* routing or search scheme [67] where packets are independently forwarded to a nearest neighbor, chosen according to the transition probabilities (3.26), until the target is reached. These probabilities contain only limited local information, namely the degree of all neighboring nodes. By construction, the associated local (stochastic) routing problem (Sect. III.B.3) does not concern link strength (bandwidth) limitations but rather the processing/queuing capabilities of the nodes, so the cost constraint, associated with the links, disappears form the problem.

3.4.1 Node Betweenness for Weighted RWs

In network-based transport or flow problems, the appropriate betweenness measure is defined to capture the amount of traffic or information passing through a node or a link, i.e., the load of a node or a link [15, 18, 52–54, 126, 134, 135]. Here, our observable interest is the *node betweenness* B_i for a given routing scheme [67] (here, purely local and characterized by a single parameter β): *the expected number of visits* to node i for a random walker originating at node s (the source) before reaching node t (the target) $E_i^{s,t}$, summed over all source-target pairs. For a general RW, as was shown by Doyle and Snell [84], $E_i^{s,t}$ can be obtained using the framework of the equivalent resistor-network problem (discussed in Sect. 3.3). More specifically,

$$E_i^{s,t} = C_i(V_i - V_t), \tag{3.27}$$

while a *unit* current is injected (removed) at the source (target) node. Utilizing again the network propagator and (3.19), one obtains

$$E_i^{s,t} = C_i(V_i - V_t) = C_i(\hat{V}_i - \hat{V}_t) = C_i(G_{is} - G_{it} - G_{ts} + G_{tt}). \tag{3.28}$$

For the node betweenness, one then obtains

$$\begin{aligned} B_i &= \sum_{s\neq t} E_i^{s,t} = \frac{1}{2}\sum_{s\neq t}(E_i^{s,t}+E_i^{t,s}) = \frac{1}{2}\sum_{s\neq t} C_i(G_{ss}+G_{tt}-2G_{ts}) \\ &= \frac{C_i}{2}\sum_{s\neq t} R_{st} = \frac{C_i}{2}N(N-1)\overline{R}. \end{aligned} \tag{3.29}$$

Note that the above expression is valid for any graph and for an arbitrary weighted RW defined by the transition probabilities (3.25). As can be seen from (3.29), the node betweenness is proportional to the product of a local topological measure, the weighted degree C_i, and a global flow measure, the average system resistance $\overline{R}$. As a specific case, for the unweighted RW ($\beta = 0$) $C_i = \sum_l A_{il} = k_i$, thus, the node betweenness is exactly proportional to the degree of the node, $B_i = k_i N(N-1)\overline{R}/2$.

Using our earlier approximations and results for uncorrelated SF graphs (3.58) and (3.16), and the relationship between the width and the average system resistance (3.23), for weighted RW, controlled by the exponent β, we find

$$B_i(\beta) = \frac{C_i}{2}N(N-1)\overline{R} = C_i N^2\langle w^2\rangle \approx N^2\frac{\gamma-1}{\gamma+\beta}\frac{k_i^{1+\beta}}{m^{1+\beta}}. \tag{3.30}$$

First, we consider the average "load" of the network

$$\overline{B} = \frac{1}{N}\sum_i B_i = \frac{\sum_i C_i}{2}(N-1)\overline{R}. \tag{3.31}$$

Similar to (3.29), the above expression establishes an exact relationship between the average node betweenness of an arbitrary RW [given by (3.25)] and the observables of the associated resistor network, the total edge cost and the average system resistance. For example, for the $\beta = 0$ case, $\overline{B} = \overline{k}N(N-1)\overline{R}/2$. As noted earlier, for calculation purposes one is free to consider the set of C_{ij} coefficients given by (3.8), which also leads us to the following statement:

For a RW defined by the transition probabilities (3.25), the average RW betweenness is minimal when the average system resistance of the associated resistor network with fixed total edge cost (and the width of the associated noisy synchronization network) is minimal.

Utilizing again our earlier approximations and results for uncorrelated SF graphs and the relationship between the width and the average system resistance, we find

$$\overline{B}(\beta) = \frac{\sum_i C_i}{2}(N-1)\overline{R} = \left(\sum_i C_i\right)N\langle w^2\rangle \approx N^2\frac{(\gamma-1)^2}{(\gamma-2-\beta)(\gamma+\beta)}. \tag{3.32}$$

The average node betweenness is minimal for $\beta = \beta^* = -1$, for all γ.

3.4.2 Commute Times and Hitting Times for Weighted RWs

The hitting (or first passage) time τ_{st} is the expected number of steps for the random walker originating at node s to reach node t for the first time. Note that using Doyle and Snell's result [84] for the expected number of visits (3.27), expressed in term of the network propagator (3.28), one can immediately obtain an expression for the expected first passage time (see Appendix 4). The commute time is the expected number of steps for a "round trip" between nodes s and t, $\tau_{st} + \tau_{ts}$. Relationships between the commute time and the effective two-point resistance have been explored and discussed in detail in several works [85, 130, 131]. In its most general form, applicable to weighted networks, it was shown by Chandra et al. [130] (see also Appendix 4) that

$$\tau_{st} + \tau_{ts} = \left(\sum_i C_i\right) R_{st}\,. \tag{3.33}$$

For the average hitting (or first passage) time, averaged over all pairs of nodes, one then obtains

$$\begin{aligned}\overline{\tau} &\equiv \frac{1}{N(N-1)}\sum_{s\neq t}\tau_{s,t} = \frac{1}{2N(N-1)}\sum_{s\neq t}(\tau_{s,t}+\tau_{t,s})\\ &= \frac{\sum_i C_i}{2N(N-1)}\sum_{s\neq t}R_{st} = \frac{\sum_i C_i}{2}\overline{R}.\end{aligned} \tag{3.34}$$

Comparing (3.31) and (3.34), the average hitting time (the average travel time for packets to reach their destinations) then can be written as $\overline{\tau} = \overline{B}/(N-1)$. Note that this relationship is just a specific realization of Little's law [136, 137], in the context of general communication networks, stating that the average time needed for a packet to reach its destination is proportional to the total load of the network. Thus, the average hitting time and the average node betweenness (only differing by a factor of $N-1$) are minimized *simultaneously* for the same graph (as a function of β, in our specific problem).

3.4.3 Network Congestion due to Queuing Limitations

Consider the simplest local "routing" or search problem [67,70,72] in which packets are generated at *identical* rate ϕ at each node. Targets for each newly generated packet are chosen uniformly at random from the remaining $N-1$ nodes. Packets perform independent, weighted RWs, using the transition probabilities (3.25), until they reach their targets. Further, the queuing/processing capabilities of the nodes are limited and are identical, e.g. (without loss of generality) each node can send out one packet per unit time. From the above it follows that the network is congestion-free as long as

$$\phi \frac{B_i}{N-1} < 1, \tag{3.35}$$

for *every* node i [10, 66, 67, 70, 71, 73]. As the packet creation rate ϕ (network throughput per node) is increased, congestion emerges at a critical value ϕ_c when the inequality in (3.35) is first violated. Up to that point, the simple model of independent random walkers (discussed in the previous subsections), can self-consistently describe the average load landscape in the network. Clearly, network throughput is limited by the most congested node (the one with the maximum betweenness); thus,

$$\phi_c = \frac{N-1}{B_{\max}}, \tag{3.36}$$

a standard measure to characterize the efficiency of communication networks [10, 66, 67, 70, 71, 73].

To enhance or optimize network throughput (limited by the onset of congestion at the nodes), one may scale up the processing capabilities of the nodes [70], optimize the underlying network topology [67], or optimize routing by finding pathways which minimize congestion [10, 71–73]. The above RW routing, with the weighting parameter β controlling "hub avoidance," is an example for the latter, where the task is to maximize global network throughput by locally directing traffic. In general, congestion can also be strongly influenced by "bandwidth" limitations (or collisions of packets), which are related to the edge betweenness, and not considered here.

According to (3.36), the network throughput is governed and limited by the largest betweenness in the network. Further, the RW betweenness of the nodes is proportional to the weighted degree, which approximately scales as a power law with the degree in SF networks (3.30). Employing the known scaling behavior of the degree cut-off (the scaling of the largest degree) in uncorrelated SF networks [15, 118, 138], one can show that the maximum RW betweenness and network throughput exhibit a minimum and a maximum, respectively, at around $\beta^* = -1$ [10]. Here, we show numerical results for the RW betweenness and the network throughput in BA SF networks. Figure 3.4 demonstrates that the RW betweenness is strongly correlated with the degree in SF networks. In particular, except for nodes with very small degrees, $B(k_i) \sim k_i^{\beta+1}$ (3.30). For $\beta \approx -1$, the load (RW betweenness) becomes balanced [Fig. 3.4] and the network throughput exhibits a maximum [Fig. 3.5]. Thus, RW weights with $\beta \approx -1$ correspond to the optimal hub avoiding weighting scheme.

In a recent, more realistic network traffic simulation study of a congestion-aware routing scheme, Danila et al. [72] found a qualitatively very similar behavior to what we have observed here. In their network traffic simulation model, packets are forwarded to a neighbor with a probability proportional to a power β of the *instantaneous queue length* of the neighbor. They found that there is an optimal value of the exponent β, close to -1.

We also show numerical results for the network throughput for SW networks with the same degree [Fig. 3.5a]. In particular, an optimally weighted SW network always

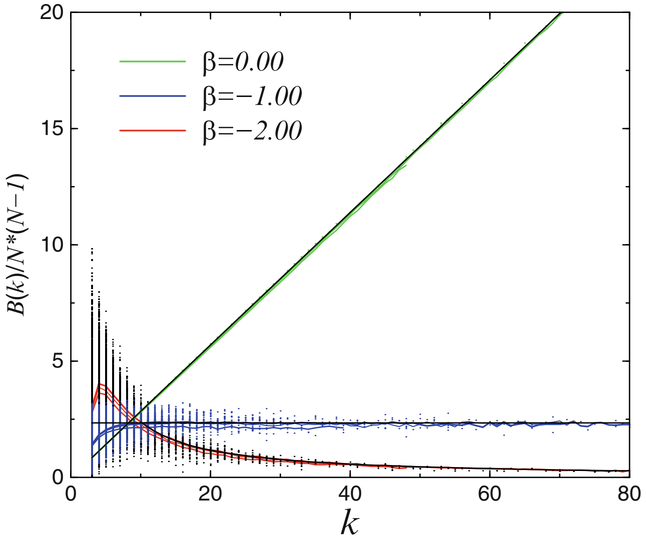


Fig. 3.4 Normalized RW node betweenness on BA networks with $m = 3$ as a function of the degree of the nodes for four system sizes [$N = 200$ (*dotted*), 400 (*dashed*), 1,000 (*long-dashed*), 2,000 (*solid*)] and for three different β values, $\beta = 0.00$, $\beta = -1.00$, and $\beta = -2.00$ (*from top to bottom*). Data point represented by *lines* are averaged over all nodes with degree k. Data for different system sizes are essentially indistinguishable. Scatter plot (*dots*) for the individual nodes is also shown from ten network realizations for $N = 1{,}000$. *Solid curves*, corresponding to the MF+UC scaling $B(k) \sim k^{\beta+1}$ (3.30), are also shown

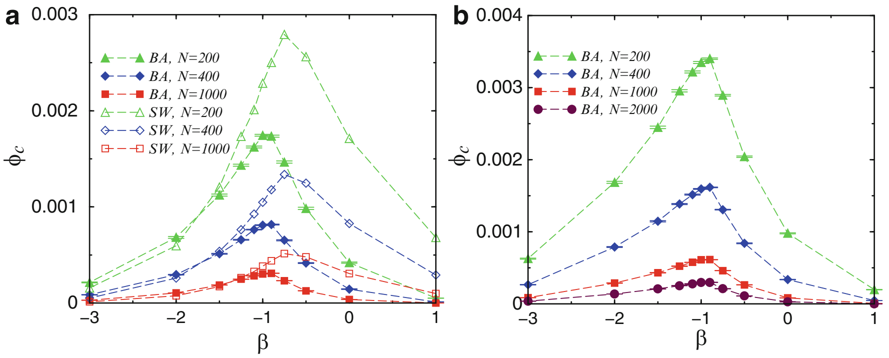


Fig. 3.5 Network throughput per node as a function of the weighting parameter β for BA networks (*solid symbols*) for various system size for (**a**) $m = 3$ and for (**b**) $m = 10$ ($\langle k \rangle \simeq 2m$). Figure (**a**) also shows the same observable for SW networks with the same average degree for the same system sizes (the same respective *open symbols*)

outperforms its BA scale-free counterpart with the same degree. Qualitatively similar results have been obtained in actual traffic simulation for networks with exponential degree distribution [72].

To summarize, the above simple weighted RW model for local routing on SF networks indicates that the routing scheme is optimal around the value $\beta^* \approx -1$. At this point, the load is balanced (3.30) and Fig. 3.4], both the average load and the average packet delivery time are minimum, and the network throughput is maximum [Fig. 3.5].

From a viewpoint of network vulnerability [139–143], the above results for the weighted RW routing scheme also implies the following. Network failures are often triggered by large load fluctuations at a specific node, then subsequently cascading through the system [142]. Consider a "normal" operating scenario (i.e., failure is *not* due to intentional/targeted attacks), where one gradually increases the packet creation rate ϕ and the overloaded nodes (ones with the highest betweenness) gradually removed from the network [143]. For $\beta > \beta^* \approx -1$ (including the unweighted RW with $\beta = 0$), these nodes are the ones with the highest degrees, but uncorrelated SF networks are structurally vulnerable to removing the hubs. At the optimal value of β, not only the network throughput is maximal, and the average packet delivery time is minimal, but the load is balanced: overloads are essentially equally likely to occur at any node and the underlying SF structure is rather resilient to random node removal [139, 140]. Thus, at the optimal value of β, the local weighted RW routing simultaneously optimizes network performance and makes the network less vulnerable against inherent system failures due to congestions at the processing nodes.

3.5 Current Flow in Weighted Networks

Current flow in resistor networks provides the simplest distributed flow model in complex networks [92]. This flow is directed and distributed, as the current flows from the highest potential node (source) to the lowest potential node (target). While current can run along all (possibly weighted) paths between the source and target nodes, more current is carried along shorter paths (with smaller resistance). Further, hanging dead ends (i.e., nodes which does not lie on a path between the source and target) will carry zero current. Thus, currents running through the nodes or the links, averaged over all source–target pairs (referred to as the current-flow betweenness), provide a good measure for information centrality, also referred to as current-flow betweenness [126, 127].

Using the same resistor network model as in Sect. 3.3 where an edge between nodes i and j has conductivity C_{ij}, for a given source (s) and target (t) pair, we can write the potential difference between nodes i and j as

$$V_i - V_j = \hat{V}_i - \hat{V}_j = I(G_{is} - G_{it} - G_{js} + G_{jt}). \tag{3.37}$$

Here, G_{ij} is the propagator (or pseudo inverse, operating in the space orthogonal to the zero mode) of the network Laplacian. If nodes i and j are connected by an edge in the network, and assuming unit current ($I = 1$) entering and leaving the network, then the current through this edge can be expressed as

$$I_{ij}^{st} = C_{ij}(V_i - V_j) = C_{ij}(G_{is} - G_{it} - G_{js} + G_{jt}). \tag{3.38}$$

Thus, exploiting the conservation of currents, the net current running through node i for a given source–target pair, can be written as

$$I_i^{st} = \frac{1}{2}\sum_j |I_{ij}^{st}| = \frac{1}{2}\sum_j C_{ij}|G_{is} - G_{it} - G_{js} + G_{jt}|. \tag{3.39}$$

Finally, considering all source–target pairs (where all nodes can simultaneously be sources and send one unit of current per unit time to a randomly chosen target), one finds the current-flow betweenness or information centrality [126, 127],

$$l_i = \frac{1}{N-1}\sum_{s,t} I_i^{st} = \frac{1}{2(N-1)}\sum_j\sum_{s,t} C_{ij}|G_{is} - G_{it} - G_{js} + G_{jt}|. \tag{3.40}$$

Despite the similarities between (3.28) and (3.38), here the summation over source and target pairs does not yield internal cancelations and simplifications, and the result for the current-flow betweenness is not amenable to simple analytic (mean-field-like) approximations. Therefore, we present only numerical results for the resulting current flow betweenness (the local load for unit input currents) l_i. Our numerical scheme was based on the exact numerical diagonalization [144] of the network Laplacian and constructing the pseudo inverse (propagator) G_{ij} using straightforward spectral decomposition. In addition to the local loads at the nodes l_i, and average system load

$$\langle l \rangle = \frac{1}{N}\sum_i l_i = \frac{1}{2N(N-1)}\sum_{i,j}\sum_{s,t} C_{ij}|G_{is} - G_{it} - G_{js} + G_{jt}|, \tag{3.41}$$

we also measured the largest current flow betweenness $l_{\max} = \max_{i=1,N}\{l_i\}$ in a given network, and then averaged over many network realizations within the same random network ensemble.

We analyzed the above observables for weighted random networks with $C_{ij} \propto (k_j k_j)^\beta$. Figure 3.6a shows that the loads (current-flow betweenness) at the nodes are strongly correlated with their degree in BA scale-free networks for $\beta = 0$, while they become much more balanced for $\beta = -1$. Also, for $\beta = 0$ (unweighted network) the load distribution exhibits fat tails, while it decays faster than any power law for $\beta = -1$ [Fig. 3.6b]; consequently, the largest load is significantly reduced for $\beta = -1$. This balanced load for $\beta = -1$, however, is achieved at the expense of a somewhat increased average load [Fig. 3.6a]. In general, we observe that reducing β leads to an increasing average load [Fig. 3.7a]. Nevertheless, the largest load in a network, potentially triggering cascading load-based failures, exhibits a minimum

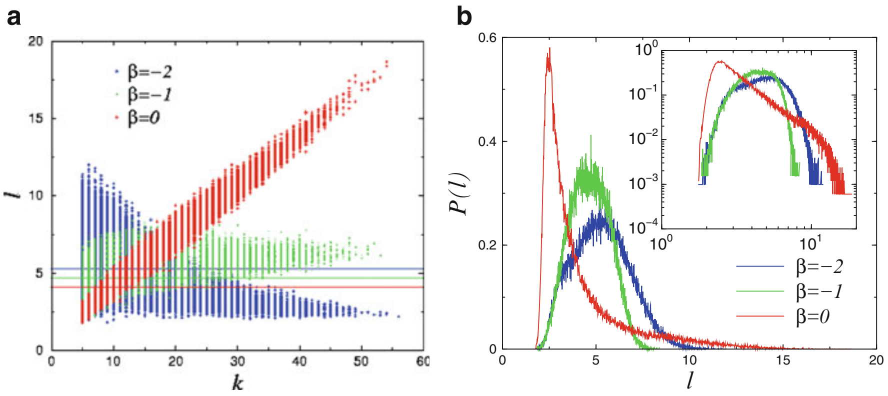


Fig. 3.6 (**a**) Scatter plot for the load (current-flow betweenness) vs. the degree for BA networks with $N = 100$ and $\langle k \rangle \simeq 10$ for three different β values. *Horizontal lines* indicate the average load. (**b**) Load distribution of BA networks with the same parameters. The inset shows the same distributions on log–log scales

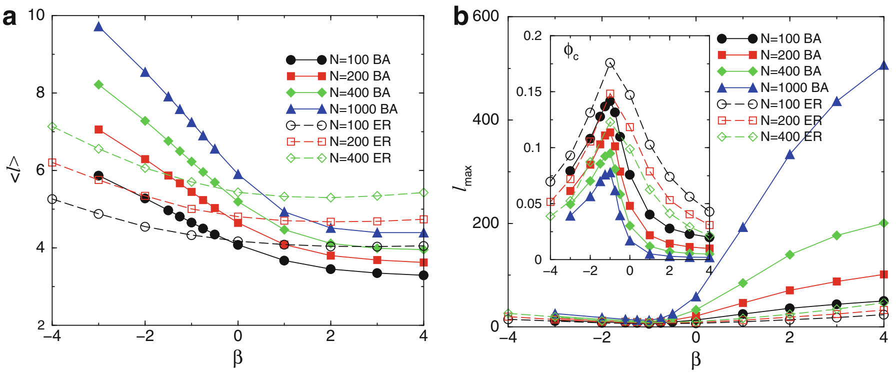


Fig. 3.7 (**a**) Average load and (**b**) maximum load in BA and ER networks with $\langle k \rangle \simeq 10$ for various network sizes as a function of β. The inset in (**b**) shows the network throughput vs. β

at around $\beta \approx -1$ [Fig. 3.7b]. In turn, the network throughput, assuming identical source–target rates and unit processing capabilities at each node [analogously to (3.36)]

$$\phi_c = \frac{1}{l_{\max}} \tag{3.42}$$

exhibits a maximum at around $\beta \approx -1$ [Fig. 3.7b inset]. Thus, with the simple weighting scheme $C_{ij} \propto (k_i k_j)^{\beta}$ one can optimize current flow such that the network throughput is maximum ($\beta^* \approx -1$).

Finally, we note that a homogeneous random network [Erdős–Rényi (ER) random graph [14, 145]] exhibits qualitatively similar characteristic in the throughput and load profile as a function of the weighting parameter β [Fig. 3.7]. Further, as

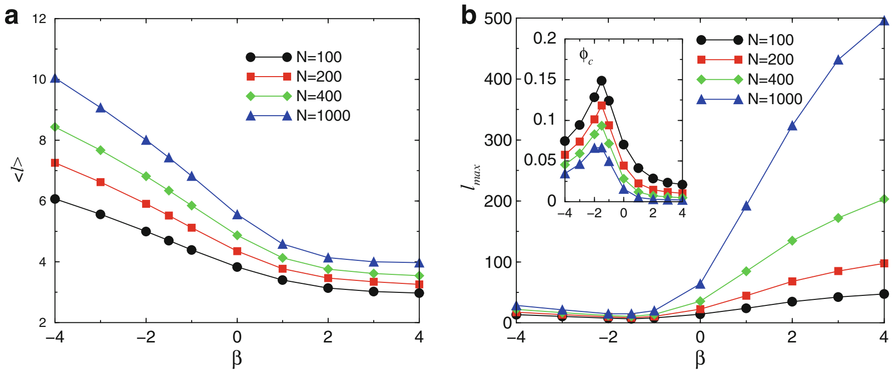


Fig. 3.8 (**a**) Average load and (**b**) maximum load for heterogeneous source/target frequencies with $\rho = 1.00$ in BA networks with $\langle k \rangle \simeq 10$ for various network sizes as a function of β. The inset in (**b**) shows the network throughput vs. β

can be seen from Fig. 3.7b, the network throughput of an ER network outperforms that of a heterogeneous BA network of the same average degree and network size for any β. Interestingly, the average load is lower for BA (ER) networks for $\beta > 0$ ($\beta < 0$) [Fig. 3.7a].

3.5.1 Current Flow Optimization for Heterogeneous Source/Target Frequencies

Analogously to the question addressed in Sec 3.3.1, one can ask what is the optimal weighting of link conductivities to maximize throughput for heterogeneous source/target frequencies. Note that there the task was to maximize global average network conductance with a fixed edge cost. Here, the task is to minimize current-flow betweenness (maximize throughput) subject to identical unit node processing capabilities for a given heterogeneous "boundary condition" (source/target rates). Here, we consider source/target rates proportional to $(k_s k_t)^\rho$, such that the global source/target flow rate per node is ϕ. Then, using (3.39), the appropriately weighted current-flow betweenness becomes

$$l_i = \frac{N}{\sum_{s,t}(k_s k_t)^\rho} \sum_{s,t} (k_s k_t)^\rho I_i^{st} . \tag{3.43}$$

In Fig. 3.8, we show results for $\rho = 1.00$ on BA networks. Similar to homogeneous source/target profiles, the average current-flow betweenness is a monotonically decreasing function of β. The maximum current-flow betweenness $l_{\max} = \max_{i=1,N}\{l_i\}$, however, exhibit a minimum, at around $\beta = -1.50$. In turn, the network throughput $\phi_c = 1/l_{\max}$ shows a maximum at the same point.

The behavior of the $\rho = 0$ [Fig. 3.7] and $\rho = 1.00$ [Fig. 3.8] are qualitatively very similar. The main quantitative difference is that the location of the optimal weighting β somewhat decreases ($\beta^* \approx -1.00$ for $\rho = 0$ and $\beta^* \approx -1.50$ for $\rho = 1.00$). Since the traffic entering and leaving the network places extra burden on the hubs, the negative optimal value of β with a larger magnitude necessitates a relatively stronger hub avoidance.

3.6 Shortest Path Betweenness in Weighted Networks

In the simplest and most commonly considered models of routing, every source node s sends packets to a given destination node t through the path of least total weight connecting s and t. This path is called the *weighted shortest path* or the *optimal path* between the given source–destination pair. The concept of betweenness previously defined in Sect. 3.4 can be adapted to the present context as follows: the *shortest path betweenness* of a node (edge) in a weighted network is defined as the number of shortest paths passing through that node (edge) [134]. The characteristics of a variant of the shortest path betweenness defined here – referred to as betweenness centrality – have been studied extensively on unweighted networks (or equivalently, for $\beta = 0$) [52, 53, 146]. Specifically, for scale-free networks with degree exponent $2 \leq \gamma \leq 3$, the distribution of betweenness centrality is known to be heavy tailed, i.e., $P(B) \sim B^{-\delta}$, where δ has been reported to be universal ($\delta \approx 2.2$) [52] or varying slowly [146].

As pointed out in Sect. 3.4, the throughput of the network (assuming identical unit processing capabilities for each node) is given by $\phi = (N-1)/B_{\max}$ where $B_{\max}$ is the maximal betweenness of the network [10, 66, 67, 70, 71, 73]. Thus, the throughput can be increased by reducing the maximal betweenness of the network. While the question of a lower bound (optimum) on the scaling of the maximal betweenness $B_{\max}$ has been previously studied [71], in the present article we focus on edge weighting schemes that can optimize throughput on the network. We restrict our study to the case where the edge weight connecting to nodes i, j is given by $w_{ij} = (k_i k_j)^{-\beta}$ where k_i, k_j are the degree of nodes i, j respectively. The edge weights considered here can be interpreted as: (1) explicit parameters like latency (time taken to traverse an edge) or (2) virtual weights assigned to edges to facilitate the assignment of paths with certain properties like *hub avoidance*. Here, for our numerical investigations, we employed the configuration model [147] with a structural degree cutoff $\sim N^{1/2}$ to generate uncorrelated scale-free graphs [118, 138], with degree exponent $\gamma = 2.5$ and with minimum degree $m = 2$.

In an unweighted network ($\beta = 0$), the betweenness of a node is known to be correlated with its degree (see Fig. 3.9). This implies that analogous to the case of random walk routing in Sect. 3.4, hubs in a scale-free network carry the highest load, and the distribution of betweenness over the network is highly heterogeneous (intuitively, this is obvious since on an unweighted network the shortest path between two nodes is the one with the smallest number of links;

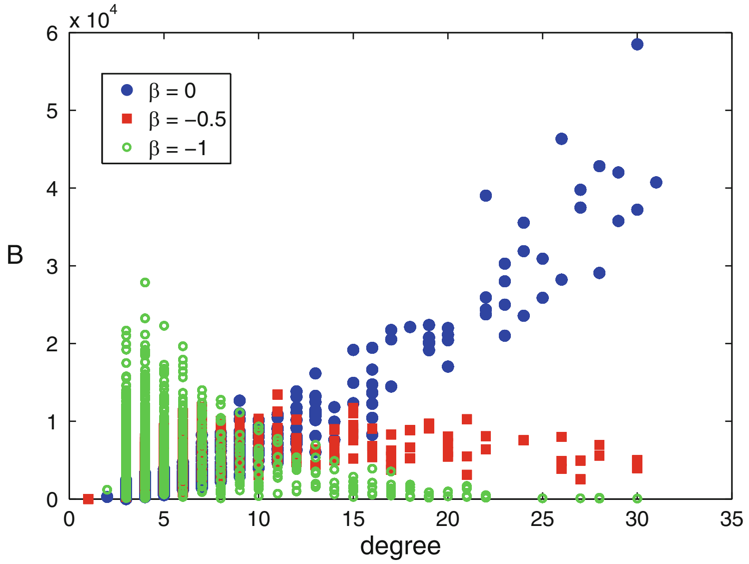


Fig. 3.9 Scatter plot showing the correlation between degree and betweenness from ten realizations of weighted scale-free networks with $N = 1{,}024$ and degree exponent $\gamma = 2.5$ using the configuration model [147]. For an unweighted network $\beta = 0$, betweennness is clearly correlated with the degree. As β is decreased below zero, at $\beta = -0.5$, the betweenness appears to be uncorrelated with the degree, while at $\beta = -1$ betweenness is biased towards lower degree nodes

since hubs by definition are well connected to the rest of the network, there is some hub that connects the source and destination through a very short path). This can be seen from Fig. 3.10, where the straight line fit to a logarithmic plot of the betweenness distribution has a slope of ≈ -2.14. From the point of view of alleviating congestion, and minimizing cascading failures (see Sect. 3.12), the ideal situation is one where the total betweenness in the network is distributed homogeneously, while keeping the value of the maximal betweenness as low as possible. Homogenizing the betweenness landscape can be achieved by introducing a small amount of hub avoidance as shown by the betweenness distribution for $\beta = -0.5$ in Fig. 3.10. The tail of the distribution is no longer fat (more appropriately it is exponential, not shown), and the maximal betweenness is lower than for $\beta = 0$ (Fig. 3.11). Also, betweenness is now no longer correlated with degree (Fig 3.9). This homogenization of the betweenness landscape comes at the expense of increasing the average betweenness on the network (see inset, Fig. 3.10). As β is decreased from -0.5, rather than further homogenizing the betweenness landscape, the hub avoidance causes the shortest paths to get longer, thus increasing the total betweenness in the network. This increase causes both the average and the maximal betweenness to rise. Furthermore, the betweenness is now largely biased towards nodes of lower degree (Fig. 3.9). Consequently, the optimal distribution

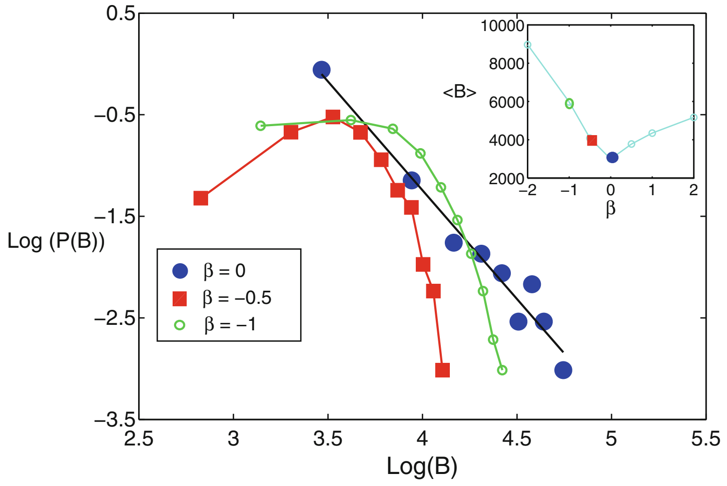


Fig. 3.10 The distribution of betweenness on weighted scale-free networks with degree exponent $\gamma = 2.5$ and network size $N = 1{,}024$. *Blue, red, and green circles* correspond to $\beta = 0, -0.5$, and -1, respectively. The *black line* is a straight line fit with slope -2.14. The inset shows the average betweenness $\langle B \rangle$ as a function of β. Results are obtained from 100 network realizations and networks are constructed using the configuration model [147]

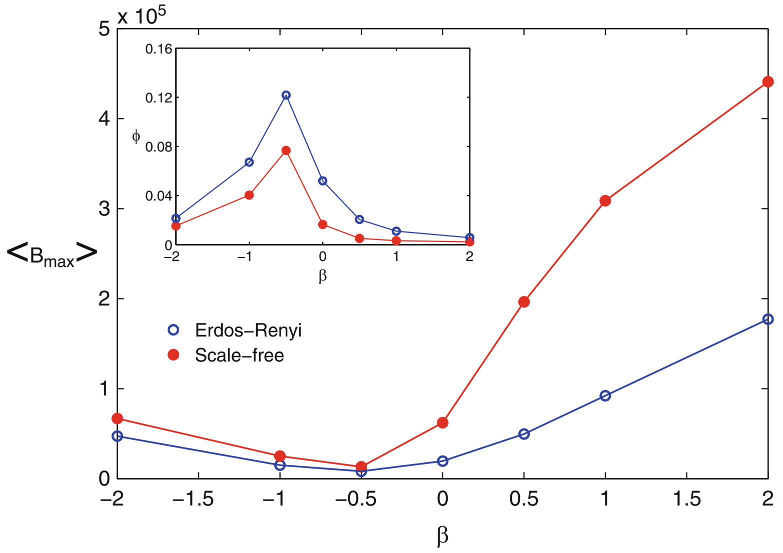


Fig. 3.11 The average value of the maximal betweenness (over 100 realizations) $\langle B_{\max} \rangle$ on weighted scale-free networks and weighted Erdős–Rényi networks of size $N = 1{,}024$. The scale-free networks considered here have degree exponent $\gamma = 2.5$ and are constructed using the configuration model [147]. Optimal values of $\langle B_{\max} \rangle$ are obtained at $\beta = -0.5$ for both classes of networks

of betweenness is obtained at $\beta = -0.5$ where the throughput is highest. Note that the same observation for the homogenization of the betweenness landscape and the minimization of the maximum betweenness was reported recently by Yang et al. [90] for BA scale free networks ($\gamma = 3$). Although there have been some attempts at analytical estimations of the optimal value of β [90], no rigorous arguments are known at present which explain this optimal value. A study of the optimal weight distribution on weighted Erdős–Rényi graphs yields similar results. However, a point worth mentioning is that for similar network size and average degree, the throughput for an Erdős–Rényi network is consistently greater than that of a scale-free network as β is varied (see Fig. 3.11).

3.6.1 Cascading Failures and Cascade Control in Weighted Networks

Infrastructure networks with complex interdependencies are known to be vulnerable to *cascading failures*. A cascading failure is a domino effect which originates when the failure of a given node triggers subsequent failures of one or several other nodes, which in turn trigger their own failures. Examples of cascading failures are abundant in the real world, including the "Northeast Blackout of 2003" (http://en.wikipedia.org/wiki/Northeast_blackout_of_2003) and the current global economic crisis [148].

The first notable study of cascading failures on networks was by Motter and Lai [149], and the model they proposed is the one we pursue here. The model assumes that in the network under consideration each node is transmitting one unit of some quantity (energy, information, etc.) to every other node through the shortest path between them. As a result, there is some "load" or betweenness incurred on each node which is equal to the number of shortest paths passing through that node. It is assumed that each node is attributed a *capacity* which is the maximum load that can be handled by the node. Since cost constraints prohibit indiscriminately increasing a node's capacity, a natural assumption is that the capacity assigned to a node is proportional to the load that it is expected to handle. Thus [149, 150],

$$C_j = (1+\alpha)B_j \tag{3.44}$$

where $\alpha \geq 0$ is a tolerance parameter which quantifies the excess load that a given node can handle. The failure of a node is simulated by the removal of the node and all links connected to it. The functioning of the network after a node failure requires a recomputation of the shortest paths that originally may have passed through the failed node. This redistribution of shortest paths can radically alter the landscape of betweenness on the network. If the redistribution causes certain nodes to have a load greater than their capacity, these nodes also fail. These failures can in turn trigger more failures, thus leading to a cascade. A natural quantity that signifies the

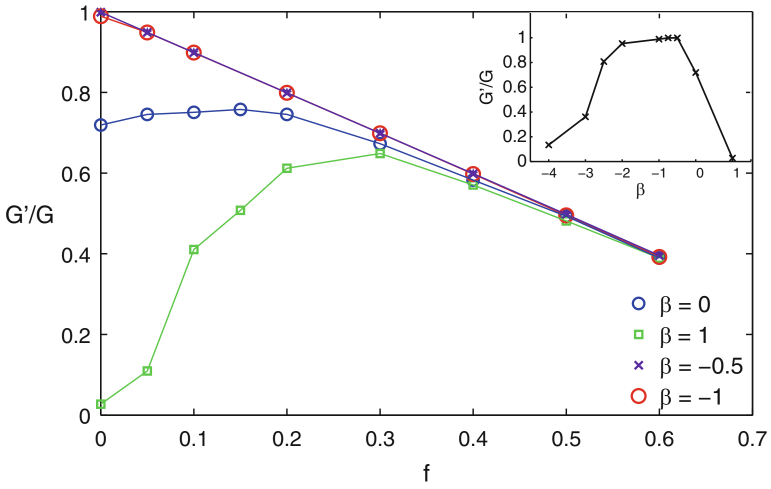


Fig. 3.12 Simulation results showing the relationship between the fraction of intentionally removed nodes f and the fractional size of the surviving giant connected component G'/G on weighted scale-free networks. Results are for ten network realizations of scale-free networks with degree exponent $\lambda = 2.5$ and $N = 1{,}000$, constructed using the configuration model [147]. The parameter α which quantifies the excess capacity (3.44) is set to 0.5 here. The inset shows the relative size of the giant component of the surviving network vs. the weighting parameter β for the $f = 0$ baseline scenario (no intentional node removal)

severity of a cascade is the ratio of the size of the giant connected component G' remaining after the cascade, to the size of the original giant component, G. Motter and Lai [149] showed that for scale free networks that local failures originating at high degree or high betweenness nodes results in cascades with a high degree of severity. In contrast, a random node failure seldom initiates a cascade, and therefore leaves most of the giant connected component intact.

In a subsequent study [151], Motter demonstrated that cascades can be stopped through the intentional removal of nodes after the initial failure has occurred but before the secondary overload failures have begun. One such strategy is to remove a certain fraction f of the nodes with the lowest betweenness. Here, we show the results of this procedure extended to weighted networks. In our simulations, cascades are initiated by the removal of the highest betweenness node on a scale-free network with $N = 1{,}000$ nodes and with $\alpha = 0.5$. Notice that the damage caused on an unweighted network $\beta = 0$ by a cascade in the absence of any defense strategy ($f = 0$) results in the giant component losing about 30% of its nodes (Fig. 3.12).

Intentional removals marginally improve the ratio G'/G until a certain optimal value of f beyond which the damage to the network is primarily a result of the intentional removals itself. Thus, beyond the optimal f for a given β,

$$\frac{G'}{G} \approx 1 - f.$$

When β is decreased below zero the shortest paths avoid the hubs; thus alleviating the load on the high degree nodes. For small negative values, $\beta = -0.5, -1$ since the total load on the network is balanced more homogeneously among all the nodes in the network (see Figs. 3.9 and 3.10) than on the unweighted network, the size of the cascade dramatically reduces even without any intentional removals i.e. $G'/G \approx 0.99$ at $f = 0$ for both $\beta = -0.5, 1$ (Fig. 3.12, inset). Furthermore, intentional removals ($f > 0$) only cause further damage . For $\beta = 1$, shortest paths are biased towards the hubs, thereby broadening the fat tail of the load distribution making it even more heterogeneous than for an unweighted network (not shown). As would be expected, the severity of a cascade in this case is far greater than that in an unweighted network, and consequently the gain arising from intentional removals is also extremely high. At the optimal f, the size of the giant component is greater than half the original network size as opposed to 3% without intentional removals. Thus, in summary for any weighted network there exists an optimal fraction of intentionally removed nodes at which the damage caused by the cascade is the least severe. Furthermore, this optimal removed fraction is very close to zero for a weighted network with $\beta = -1$; thus, implying that for this value of β networks are maximally resilient to cascading failures for the network parameters used here.

In the model addressed here, the loads and therefore the capacities result from the particular assignment of shortest paths on the network. Thus, the loads and capacities are inherently tied to the topology of the network. An alternative model proposed in [142] looked at similar failure triggered cascades but where the loads on each node were drawn from an arbitrary distribution uncorrelated with the topology of the network. Further studies of cascading failures on weighted networks subject to empirically observed forms of the load–capacity relationship [152] can be found in [89, 90]. The closely related problem of attacking a network by iteratively damaging the node with the highest betweenness and recalculating the betweenness after each damage iteration has been studied in [143, 153].

3.7 Summary and Outlook

In this chapter, we considered a simple class of weighted networks in the context of synchronization, flow, and robustness. In particular, we considered weighted edges $C_{ij} \propto A_{ij}(k_i k_j)^{\beta}$, and investigated optimizing the relevant network observables, i.e., minimizing the width of the synchronization landscape, maximizing the throughput in network flow, or maximizing the size of the surviving giant component following cascading failures (triggered by local overloads).

Our models and methods provided some insights into the challenging problem of optimizing the allocation of limited resources [152, 154] in weighted complex networks. Our results for these fundamental models support that even with this simple one-parameter (β) optimization, one can significantly improve global network performance, as opposed to performing an exhaustive and computationally prohibitive search for optimal weight allocations. It is also important to note that in our optimization problems for RWs (Sect. 4) and flow (Sects. 5 and 6), for simplicity, we considered processing or queuing limitations at the nodes. Within an identical framework, however, one should also consider and study edge-limited flows (motivated by finite bandwidth) with weighted links [155, 156]. Our preliminary results indicate that while optimization is possible, it naturally occurs at a different value of the weighting parameter β. This implies that one cannot optimize and balance traffic for both queueing and bandwidth limitation simultaneously, but instead, trade-offs have to be considered with the knowledge of specific systems.

Real-life information, communication, and infrastructure networks are not only weighted and heterogeneous, but are also spatially embedded [65, 157, 158] and can also exhibit degree correlations [15, 18]. The corresponding metrics (Euclidean distance) strongly influences the cost of the edges, and in turn, the optimal distribution of limited resources. We currently explore and investigate these problems on weighted spatially-embedded complex networks.

Acknowledgements This research was supported in part by RPI's Seed Grant, NSF Grant No. DMR-0426488, DTRA Award No. HDTRA1-09-1-0049, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

Appendix 1: Globally Optimal Network with Fixed Edge Cost

In this Appendix we determine the the minimum attainable width in the EW synchronization problem for networks with a fixed edge cost. Further, we identify a network which realizes this globally optimal synchronization efficiency. For the EW synchronization problem we can express the total edge cost with the eigenvalues of the network Laplacian,

$$2C_{\text{tot}} = \sum_{i,j} C_{ij} = \sum_{i} C_i = \sum_{i} \Gamma_{ii} = \text{Tr}(\Gamma) = \sum_{l\neq 0} \lambda_l . \tag{3.45}$$

Thus, the global optimization problem can be cast as

$$\langle w^2 \rangle = \frac{1}{N} \sum_{l=1}^{N-1} \frac{1}{\lambda_l} = \text{minimum}, \tag{3.46}$$

with the constraint

$$\sum_{l=1}^{N-1} \lambda_l = 2C_{\text{tot}} = \text{fixed}. \tag{3.47}$$

This elementary extremum problem, (3.46) and (3.47), immediately yields a solution where all $N-1$ non-zero eigenvalues are equal,

$$\lambda_l = \frac{2C_{\text{tot}}}{N-1}\,, \quad l = 1,2,\ldots,N-1, \tag{3.48}$$

and the corresponding *absolute* minimum of the width is

$$\langle w^2\rangle_{\text{min}} = \frac{(N-1)^2}{2NC_{\text{tot}}}. \tag{3.49}$$

As one can easily see, the above set of identical eigenvalues corresponds to a coupling matrix and network structure where each node is connected to all others with identical strength $C_{ij} = 2C_{\text{tot}}/[N(N-1)]$. That is, for fixed cost, the *fully-connected* (FC) network is optimal, yielding the absolute minimum width.

Appendix 2: The Mean-Field Approximation in Stochastic Synchronization on Networks

Summing up the exact equations of motion (3.1) over all nodes and exploiting the symmetry $C_{ij} = C_{ji}$ yields the stochastic equation for the mean

$$\partial_t \overline{h} = \xi(t), \tag{3.50}$$

where $\xi(t) = \frac{1}{N}\sum_i \eta_i(t)$. From the properties of the individual noise terms in (3.1) it follows that $\langle\xi(t)\rangle = 0$ and $\langle\xi(t)\xi(t')\rangle = \frac{2}{N}\delta(t-t')$. Note that the above stochastic equation is exact for the mean $\overline{h}(t)$. In the mean-field (MF) approximation one replaces the local neighborhood averages by the global mean $\overline{h}$ (3.11) (which is a crude approximation) yielding

$$\partial_t h_i \approx -C_i\left(h_i - \overline{h}\right) + \eta_i(t). \tag{3.51}$$

Since the time evolution of the mean is now explicit (3.50), from (3.51) we can obtain the approximate equations of motion for the fluctuations with respect to the mean, $\Delta_i(t) \equiv h_i(t) - \overline{h}(t)$,

$$\partial_t \Delta_i(t) \approx -C_i\,\Delta_i(t) + \tilde{\eta}_i(t), \tag{3.52}$$

where $\tilde{\eta}_i(t) \equiv \eta_i(t) - \xi(t)$ with $\langle\tilde{\eta}_i(t)\rangle$ and $\langle\tilde{\eta}_i(t)\tilde{\eta}_j(t')\rangle = 2(\delta_{ij} - \frac{1}{N})\delta(t-t')$. From elementary properties of the above linear stochastic differential equations [159] for the equal-time steady-state fluctuations one finds

$$\langle \Delta_i(t)\Delta_j(t)\rangle = \frac{2}{C_i+C_j}\left(\delta_{ij}-\frac{1}{N}\right). \tag{3.53}$$

Thus, the steady-state fluctuations about the mean decouple in the asymptotic large N limit, while $\langle (h_i-\bar{h})^2\rangle = \langle \Delta_i^2\rangle \approx 1/C_i$.

Appendix 3: The Weighted Degree for Uncorrelated SF Graphs

Here, we establish an approximate relationship between the weighted degree C_i and the degree k_i of node i for *uncorrelated* (UC) weighted SF graphs. Note that C_i also becomes the effective coupling to the mean in the mean-field approximation of the EW synchronization problem. Using the specific form of the weights as constructed in (3.8), we write

$$C_i = \sum_j C_{ij} = N\bar{k}\frac{\sum_j A_{ij}(k_ik_j)^\beta}{\sum_{l,n} A_{ln}(k_lk_n)^\beta} = N\bar{k}\frac{k_i^\beta \sum_j A_{ij}k_j^\beta}{\sum_l k_l^\beta \sum_n A_{ln}k_n^\beta}. \tag{3.54}$$

For large minimum (and in turn, average) degree, expressions of the form $\sum_j A_{ij}k_j^\beta$ can be approximated as

$$\sum_j A_{ij}k_j^\beta = \left(\sum_j A_{ij}\right)\frac{\sum_j A_{ij}k_j^\beta}{\sum_j A_{ij}} = k_i\frac{\sum_j A_{ij}k_j^\beta}{\sum_j A_{ij}} \approx k_i \int \mathrm{d}kP(k|k_i)k^\beta, \tag{3.55}$$

where $P(k|k')$ is the probability that an edge from node with degree k' connects to a node with degree k. For *uncorrelated* random graphs, $P(k|k')$ does *not* depend on k', and one has $P(k|k') = kP(k)/\langle k\rangle$ [15, 18], where $P(k)$ is the degree distribution and $\langle k\rangle$ is the ensemble-averaged degree. Thus, (3.54), for UC random networks, can be approximated as

$$C_i \approx N\langle k\rangle \frac{k_i^{\beta+1}\int \mathrm{d}kP(k|k_i)k^\beta}{N\int \mathrm{d}k'k'^{\beta+1}P(k')\int \mathrm{d}kP(k|k')k^\beta} = \langle k\rangle \frac{k_i^{\beta+1}}{\int_m^\infty \mathrm{d}k'k'^{\beta+1}P(k')}. \tag{3.56}$$

Here, we consider SF degree distributions,

$$P(k) = (\gamma-1)m^{\gamma-1}k^{-\gamma}, \tag{3.57}$$

where m is the minimum degree in the network and $2<\gamma\le 3$. The average and the minimum degree are related through $\langle k\rangle = m(\gamma-1)/(\gamma-2)$. No upper cutoff is needed for the convergence of the integral in (3.56), provided that $2+\beta-\gamma<0$, and one finds

$$C_i \approx \frac{\gamma-2-\beta}{\gamma-2}\frac{k_i^{\beta+1}}{m^\beta}. \tag{3.58}$$

Thus, for uncorrelated random SF graphs with large minimum degree, the effective coupling coefficient C_i only depends on the degree k_i of node i, i.e., for a node with degree k

$$C(k) \approx \frac{\gamma-2-\beta}{\gamma-2}\frac{k^{\beta+1}}{m^\beta}. \tag{3.59}$$

Appendix 4: RW Hitting Times and the Network Propagator

Employing Doyle and Snell's result [84] for the expected number of visits (3.27), and expressing the voltage difference of the associated resistor networks in terms of the network propagator (or pseudo inverse of the network Laplacian) (3.19) one has

$$E_i^{s,t} = C_i(V_i - V_t) = C_i(\hat{V}_i - \hat{V}_t) = C_i(G_{is} - G_{it} - G_{ts} + G_{tt}). \tag{3.60}$$

Then the hitting (or first passage) time, which is the expected number of steps in a RW which starts at node s and ends upon first reaching node t, can be written as

$$\tau_{st} = \sum_i E_i^{s,t} = \sum_i C_i(G_{is} - G_{it} - G_{ts} + G_{tt}). \tag{3.61}$$

The expression for the symmetric commute time (expected number of steps for a "round-trip" between nodes s and t) simplifies significantly,

$$\tau_{st} + \tau_{ts} = \sum_i (E_i^{s,t} + E_i^{t,s}) = \sum_i C_i(G_{ss} + G_{tt} - 2G_{ts}) = \left(\sum_i C_i\right) R_{st}, \tag{3.62}$$

where we used the expression for the two-point resistance of the associated resistor network (3.20).

References

1. S.H. Strogatz, Nature **410**, 268 (2001).
2. S.H. Strogatz, *Synch: the Emerging Science of Spontaneous Order* (Hyperion, New York. 2003).
3. S. Boccaletti, J. Kurths, G. Osipov, D. L. Valladares, and C. S. Zhou, Phys. Rep. **366**, 1 (2002).
4. J.A. Acebrón, L.L. Bonilla, C.J. Pérez Vicente, F. Ritort, and R. Spigler, Rev. Mod. Phys. **77**, 137 (2005).
5. A. Arenas *et al.*, Phys. Rep.**469**, 93 (2008).

6. R. Olfati-Saber, J.A. Fax, and R.M. Murray, Proc. IEEE **95**, 215 (2007).
7. R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, Englewood Cliffs, NJ, 1993).
8. B. Tadic, G.J. Rodgers, and S. Thurner, Int. J. Bifurcation and Chaos, **17**, 2363 (2007).
9. L.K. Gallos, C. Song, S. Havlin, and H.A. Makse, Proc. Natl. Acad. Sci. USA **104**, 7746 (2007).
10. G. Korniss, Phys. Rev. E **75**, 051121 (2007).
11. A. Schrijver, "Flows in railway optimization", Nieuw Archief voor Wiskunde **5/9** 126 (2008).
12. D.J. Watts and S.H. Strogatz, Nature **393**, 440 (1998).
13. A.-L. Barabási and R. Albert, Science **286**, 509 (1999).
14. R. Albert and A.-L. Barabási, *Rev. Mod. Phys.* **74**, 47 (2002).
15. S.N. Dorogovtsev and J.F.F. Mendes, *Adv. in Phys.* **51**, 1079 (2002).
16. M.E.J. Newman, *SIAM Review* **45**, 167 (2003).
17. L. Li, D. Alderson, R. Tanaka, J.C. Doyle, W. Willinger, Internet Math. **2**, 431 (2005); arXiv:cond-mat/0501169.
18. R. Pastor-Satorras and Alessandro Vespignani, *Evolution and Structure of the Internet: A Statistical Physics Approach* (Cambridge University Press, 2004).
19. S. Boccaletti, V. Latora Y. Moreno, M. Chavez, D.-U. Hwang, Phys. Rep. **424** 175 (2006).
20. A. Barrat, M. Barthelemy, and Alessandro Vespignani, *Dynamical processes in complex networks* (Cambridge University Press, 2008).
21. G. Korniss, M.A. Novotny, H. Guclu, and Z. Toroczkai, P.A. Rikvold, Science **299**, 677 (2003).
22. R. Olfati-Saber and R.M. Murray, IEEE Trans. Automat. Contr. **49**, 1520 (2004).
23. R. Olfati-Saber, in Proc. American Control Conf. (IEEE, Los Alamitos, CA, 2005). pp. 2371–2378.
24. C.W. Reynolds, Computer Graphics, **21**, 25 (1987).
25. T. Vicsek *et al.*, Phys. Rev. Lett. **75**, 1226 (1995).
26. F. Cucker and S. Smale, IEEE Trans. Automat. Contr. **52**, 852 (2007).
27. J.A. Fax and R.M. Murray, IEEE Trans. Automat. Contr. **49**, 1465 (2004).
28. T.I. Netoff, R. Clewley, S. Arno, T. Keck, and J.A. White, The Journal of Neuroscience **24**, 8075 (2004).
29. G. Grinstein and R. Linsker, Proc. Natl. Acad. Sci. USA **102**, 9948 (2005).
30. E. Izhikevich, SIAM Rev. **43**, 315 (2001).
31. Q. Wang, Z. Duan, M. Perc, and G. Chen, Europhys. Lett. **83**, 50008 (2008).
32. Q. Wang, M. Perc, Z. Duan, and G. Chen, Phys. Rev. E **80**, 026206 (2009).
33. A.T. Winfree, J. Theor. Biol. **16**, 15 (1967).
34. D. Lusseau, B. Wilson, P.S. Hammond, K. Grellier, J.W. Durban, K.M. Parsons, T.R. Barton, P.M. Thompson, Journal of Animal Ecology **75**, 14 (2006).
35. Y. Rabani, A. Sinclair, and R. Wanka, "Local Divergence of Markov Chains and the Analysis of Iterative Load-Balancing Schemes", in Proc, 39th Annual Symposium on Foundations of Computer Science (IEEE Computer Society, Washington, DC, 1998) pp. 694–702.
36. G. Korniss, Z. Toroczkai, M.A. Novotny, and P.A. Rikvold, Phys. Rev. Lett. **84**, 1351 (2000).
37. P.M.A. Sloot, B.J. Overeinder, and A. Schoneveld, Comput. Phys. Commun. **142**, 76 (2001).
38. S. Kirkpatrick, Science **299**, 668 (2003).
39. A. Kolakowska and M. A. Novotny, in *Artificial Intelligence and Computer Science*, edited by S. Shannon (Nova Science Publishers, Inc., New York, 2005), pp. 151–176.
40. A. Schrijver, Mathematical Programming, **91**, 437 (2002).
41. A. N. Tolstoi, in *Transportation Planning*, Vol. I, (TransPress of the National Commissariat of Transportation, Moscow, 1930) pp. 23.55.
42. F.L. Hitchcock, J. of Math. and Phys. **20**, 224 (1941).
43. L.V. Kantorovich and M.K. Gavurin, in *Collection of Problems of Raising the Efficiency of Transport Performance*, Akademiia Nauk SSSR, Moscow-Leningrad, 1949, pp. 110–138.
44. Tj.C. Koopmans, Econometrica **17** (Supplement), 136 (1949).

45. T.E. Harris, F.S. Ross, “Fundamentals of a Method for Evaluating Rail Net Capacities”, Research Memorandum RM-1573, The RAND Corporation, Santa Monica, California, 1955.
46. L.R. Ford and D.R. Fulkerson, “Maximal Flow through a Network”, Research Memorandum RM-1400, The RAND Corporation, Santa Monica, CA, 1954; Canadian J. Math. **8**, 399 (1956).
47. L.R. Ford and D.R. Fulkerson, *Flows in Networks* (Princeton University Press, Princeton, NJ, 1962).
48. D.L. Alderson, Operations Research **56**, 1047 (2008).
49. L. Donetti, F. Neri, and M.A. Munoz, J. Stat. Mech. P08007 (2006).
50. P. J. Macdonald, E. Almaas, A.-L. Barabási, Europhys. Lett. **72**, 308 (2005).
51. A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani, Proc. Natl. Acad. Sci. USA **101**, 3747 (2004).
52. K.-I. Goh, B. Kahng, and D. Kim, Phys. Rev. Lett. **87**, 278701 (2001).
53. K.-I. Goh, J.D. Noh, B. Kahng, and D. Kim, Phys. Rev. E **72**, 017102 (2005).
54. J.D. Noh and H. Rieger, Phys. Rev. Lett. **92**, 118701 (2004).
55. L.K. Gallos, Phys. Rev. E **70**, 046116 (2004).
56. E. Almaas, R.V. Kulkarni, and D. Stroud, Phys. Rev E **68**, 056105 (2003).
57. M. Argollo de Menezes and A.-L. Barabási, Phys. Rev. Lett. **92** 028701 (2004).
58. E. Almaas, B. Kovacs, T. Vicsek, Z.N. Oltvai and A.-L. Barabási, Nature **427**, 839 (2004).
59. Z. Toroczkai and K. Bassler, Nature **428**, 716 (2004).
60. R. Guimerà, S. Mossa, A. Turtschi, and L.A.N. Amaral, Proc. Natl. Acad. Sci. USA **102**, 7794 (2005).
61. K. Park, Y.-C. Lai, L. Zhao, and N. Ye, Phys. Rev. E **71**, 065105(R) (2005).
62. D.J. Ashton, T.C. Jarrett, and N.F. Johnson, Phys. Rev. Lett. **94**, 058701 (2005).
63. D.-S. Lee and H. Rieger, Europhys. Lett. **73**, 471 (2006).
64. D. Brockmann, L. Hufnagel, and T. Geisel, Nature **439**, 462 (2006).
65. V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani, Proc. Natl. Acad. Sci. USA **103**, 2015 (2006).
66. W. Krause, I. Glauche, R. Sollacher, and M. Greiner, Physica A **338**, 633 (2004).
67. R. Guimerà, A. Díaz-Guilera, F. Vega-Redondo, A. Cabrales, and A. Arenas, Phys. Rev. Lett. **89**, 248701 (2002).
68. B. Tadić, S. Thurner, G.J. Rodgers, Phys. Rev. E **69**, 036102 (2004).
69. P.E. Parris and V.M. Kenkre, Phys. Rev E **72**, 056119 (2005).
70. L. Zhao, Y.-C. Lai, K. Park, and Nong Ye, Phys. Rev. E **71**, 026125 (2005).
71. S. Sreenivasan, R. Cohen, E. López, Z. Toroczkai, and H.E. Stanley, Phys. Rev. E. **75**, 036105 (2007).
72. B. Danila, Y. Yu, S. Earl, J.A. Marsh, Z. Toroczkai, and K.E. Bassler, Phys. Rev. E **74**, 046114 (2006).
73. B. Danila, Y. Yu, J.A. Marsh, and K.E. Bassler, Phys. Rev. E **74**, 046106 (2006).
74. B. Danila, Y. Yu, J.A. Marsh, and K.E. Bassler, Chaos **17**, 026102 (2007).
75. T. Antal and P.L. Krapivsky, Phys. Rev. E **74**, 051110 (2006).
76. A. Nagurney and Q. Qiang, Europhys. Lett. **79**, 38005 (2007).
77. A. Nagurney and Q. Qiang, Europhys. Lett. **80**, 68001 (2007).
78. D.J. Aldous, “Cost-volume relationships for flows through a disordered network”, Math. Oper. Res. **33**, 769 (2008).
79. J.S. Andrade, Jr., H.J. Herrmann, R.F.S. Andrade, and L.R. da Silva, Phys. Rev. Lett. **94**, 018702 (2005).
80. E. López, S.V. Buldyrev, S. Havlin, and H.E. Stanley, Phys. Rev. Lett. **94**, 248701 (2005).
81. S. Carmi, Z. Wu, E. López, S. Havlin, and H.E. Stanley, Eur. Phys. J. B **57**, 165 (2007).
82. Z. Wu, L.A. Braunstein, S. Havlin, and H.E. Stanley, Phys. Rev. Lett. **96**, 148702 (2006).
83. M. Barthélemy and A. Flammini, J. Stat. Mech. L07002 (2006).
84. P.G. Doyle and J.L. Snell, *Random Walks and Electric Networks*, Carus Mathematical Monograph Series Vol. 22 (The Mathematical Association of America, Washington, DC, 1984), pp. 83–149; arXiv:math.PR/0001057.

85. L. Lovász, *Random Walks on Graphs: A Survey in Combinatorics*, Paul Erdős is Eighty Vol. 2, edited by D. Miklós, V.T. Sós, and T. Szőnyi (János Bolyai Mathematical Society, Budapest, 1996), pp. 353-398; http://research.microsoft.com/users/lovasz/erdos.ps.
86. S. Redner, *A Guide to First-Passage Processes* (Cambridge University Press, Cambridge, UK, 2001).
87. G. Korniss, M.B. Hastings, K.E. Bassler, M.J. Berryman, B. Kozma, and D. Abbott, Phys. Lett. A **350**, 324 (2006).
88. C. Zhou, A.E. Motter, and J. Kurths, Phys. Rev. Lett. **96**, 034101 (2006).
89. W.-X. Wang and G. Chen, Phys. Rev. E **77**, 026101 (2008).
90. R. Yang, W.-X. Wang, Y.-C. Lai, and G. Chen, Phys. Rev. E **79**, 026112 (2009).
91. A. Baronchelli and R. Pastor-Satorras, Phys. Rev. E **82**, 011111 (2010).
92. R. Huang, "Flow Optimization in Complex Networks", M.S. Thesis, Rensselaer Polytechnic Institute, Troy, NY (2010).
93. Y. Kuramoto, in *Proceedings of the International Symposium on Mathematical Problems in Theoretical Physics*, edited by H. Araki, Lecture Notes in Physics Vol. 39 (Springer, New York, 1975) pp. 420–422.
94. H. Hong, M.Y. Choi, and B.J. Kim, Phys. Rev. E **65**, 026139 (2002).
95. T. Ichinomiya, Phys. Rev. E **70**, 026116 (2004).
96. D.-S. Lee, Phys. Rev. E **72**, 026208 (2005).
97. M. Barahona and L.M. Pecora, Phys. Rev. Lett. **89**, 054101 (2002).
98. T. Nishikawa, A.E. Motter, Y.-C. Lai, and F.C. Hoppensteadt, Phys. Rev. Lett. **91**, 014101 (2003).
99. A.E. Motter, C. Zhou, and J. Kurths, Europhys. Lett. **69**, 334 (2005).
100. A.E. Motter, C. Zhou, and J. Kurths, Phys. Rev. E. **71**, 016116 (2005).
101. C. Zhou and J. Kurths, Chaos **16**, 015104 (2006).
102. L.M. Pecora and T.L.Carroll, Phys. Rev. Lett **80**, 2109 (1998).
103. S.M. Park and B.J. Kim, Phys. Rev E **74**, 026114 (2006).
104. T. Nishikawa and A.E. Motter, Phys. Rev. E **73**, 065106(R) (2006).
105. T. Nishikawa and A.E. Motter, Proc. Natl. Acad. Sci. U.S.A. **107**, 10342 (2010).
106. Z. Toroczkai, G. Korniss, M. A. Novotny, and H. Guclu, in *Computational Complexity and Statistical Physics*, edited by A. Percus, G. Istrate, and C. Moore, Santa Fe Institute Studies in the Sciences of Complexity Series (Oxford University Press, 2005), pp. 249–270; arXiv:cond-mat/0304617.
107. H. Guclu, G. Korniss, Z. Toroczkai, Chaos **17**, 026104 (2007).
108. A. Nagurney, J. Cruz, J. Dong, and D. Zhang, European Journal of Operational Research **26**, 120 (2005).
109. S.F. Edwards and D.R. Wilkinson, Proc. R. Soc. London, Ser A **381**, 17 (1982).
110. B. Kozma and G. Korniss, in *Computer Simulation Studies in Condensed Matter Physics XVI*, edited by D.P. Landau, S.P. Lewis, and H.-B. Schüttler, Springer Proceedings in Physics Vol. 95 (Springer-Verlag, Berlin, 2004), pp. 29–33.
111. B. Kozma, M. B. Hastings, and G. Korniss, Phys. Rev. Lett. **92**, 108701 (2004).
112. B. Kozma, M. B. Hastings, and G. Korniss, Phys. Rev. Lett. **95**, 018701 (2005).
113. B. Kozma, M.B. Hastings, and G. Korniss, in *Noise in Complex Systems and Stochastic Dynamics III*, edited by L.B. Kish, K. Lindenberg, Z. Gingl, Proceedings of SPIE Vol. 5845 (SPIE, Bellingham, WA, 2005) pp.130–138.
114. M. B. Hastings, Eur. Phys. J. B **42**, 297 (2004).
115. D. Hunt, G. Korniss, and B.K. Szymanski, Phys. Rev. Lett. **105**, 068701 (2010).
116. C. E. La Rocca, L. A. Braunstein, and P. A. Macri, Phys. Rev. E **77**, 046120 (2008).
117. C. E. La Rocca, L. A. Braunstein, and P. A. Macri, Phys. Rev. E **80**, 026111 (2009).
118. M. Catanzaro, M. Boguña, and R. Pastor-Satorras, Phys. Rev. E **71**, 027103 (2005).
119. M.E.J. Newman and D.J. Watts, Phys. Lett. A **263**, 341 (1999).
120. R. Monasson, Eur. Phys. J. B **12**, 555 (1999).
121. S. Kirkpatrick, Phys. Rev. Lett. **27**, 1722 (1971).

122. S. Kirkpatrick, Rev. Mod. Phys. **45**, 574 (1973).
123. M.E.J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
124. F. Wu and B.A. Huberman, Eur. Phys. J. B. **38**, 331 (2004).
125. C. Faloutsos, K.S. McCurley, and A. Tomkins, in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM Press, New York, 2004) pp. 118–127.
126. M.E.J. Newman, Social Networks **27**, 39 (2005).
127. U. Brandes and D. Fleischer, *Lecture Notes in Computer Science*, edited by V. Diekert and B. Durand (Springer, NY, 2005) Vol. 3404, pp. 533–544.
128. R. Kaul, Y. Yun, and S.-G. Kim, Comm. ACM **52**, 132 (2009).
129. F.Y. Wu, J. Phys. A **37**, 6653 (2004).
130. A.K. Chandra, P. Raghavan, W.L. Ruzzo, and R. Smolensky, in *Proceedings of the 21st Annnual ACM Symposium on the Theory of Computing* (ACM Press, New York, 1989), pp. 574–586.
131. P. Tetali, J. Theor. Prob. **4** 101 (1991).
132. L. Adamic, R.M. Lukose, A.R. Puniyani, and B.A. Huberman, Phys. Rev. E **64** 046135 (2001).
133. H.P. Thadakamalla, R. Albert, and S.R.T. Kumara, Phys. Rev. E **72** 066128 (2005).
134. L.C. Freeman, Sociometry **40**, 35 (1977).
135. L.C. Freeman, Social Networks **1**, 215 (1979).
136. J.D.C. Little, Operations Res. **9**, 383 (1961).
137. A.O. Allen, *Probability, Statistics, and Queueing Theory with Computer Science Applications*, 2nd ed. (Academic Press, Boston, 1990).
138. M. Boguña, R. Pastor- Satorras, and A. Vespignani, Eur. Phys. J. B **38**, 205 (2004).
139. R. Albert. H. Jeong, and A.-L. Barabási, Nature **406**, 378 (2000).
140. R. Cohen, K. Erez, D. ben-Avraham, S. Havlin, Phys. Rev. Lett. **85**, 4626 (2000).
141. R. Cohen, K. Erez, D. ben-Avraham, S. Havlin, Phys. Rev. Lett. **86**, 3682 (2001).
142. Y. Moreno, R. Pastor-Satorras, A. Vázquez, and A. Vespignani, *Europhys. Lett.* **62**, 292 (2006).
143. L. Dall'Asta, A. Barrat, M. Barthélemy, and A. Vespignani, J. Stat. Mech. P04006 (2006).
144. W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery *Numerical Recipes in C*, 2nd ed. (Cambridge Univ. Press, Cambridge, 1995), Secs. 11.2 and 11.3.
145. P. Erdős and A. Rényi, Publ. Math. Inst. Hung. Acad. Sci. **5**, 17 (1960).
146. M. Barthélemy, Eur. Phys. J, B **38**, 1434 (2004)
147. M. Molloy and B. Reed, Random. Struct. Algorithms **6**, 161 (1995).
148. Jeffrey Sachs, Scientific American **300**, 34 (2009).
149. A.E. Motter and Y.-C. Lai, Phys. Rev. E **66**, 065102(R) (2002).
150. L. Zhao, K. Park, and Y.-C. Lai, Phys. Rev. E **70**, 035101(R) (2004).
151. A.E. Motter, Phys. Rev. Lett. **93**, 098701 (2004).
152. D.-H. Kim and A.E. Motter, New J. Phys. **10**, 053022 (2008).
153. P. Holme, B.J. Kim, C.N. Yoon, and S.K. Han, Phys. Rev. E **65**, 056109 (2002).
154. D.-H. Kim and A.E. Motter, J. Phys. A **41**, 224019 (2008).
155. A. Asztalos, S. Sreenivasan, G. Korniss, and B.K. Szymanski, "Distributed flow optimization and cascading effects in weighted complex networks" (to be submitted, 2011).
156. X. Ling, M.-B. Hu, W.-B. Du, R. Jiang, Y.H. Wu, ans Q.S. Wu, Phys. Lett. A **374**, 4825 (2010).
157. H.P. Thadakamalla, R. Albert, and S.R.T. Kumara, "Search in spatial scale-free networks", New J. Phys. **9** 190 (2007).
158. M. Ángeles Serrano, D. Krioukov, and M. Boguña, "Self-Similarity of Complex Networks and Hidden Metric Spaces", Phys. Rev. Lett. **100**, 078701 (2008)
159. C.W. Gardiner, *Handbook of Stochastic Methods* 2nd ed. (Springer-Verlag, New York, 1985).

Chapter 4
Joint Optimization of Resources and Routes for Minimum Resistance: From Communication Networks to Power Grids

Ali Tizghadam, Alireza Bigdeli, Alberto Leon-Garcia, and Hassan Naser

Abstract In this chapter, we are concerned with robustness design in complex communication networks and power grids. We define robustness as the ability of a network to adapt to environmental variations such as traffic fluctuations, topology modifications, and changes in the source (sink) of external traffic. We present a network theory approach to the joint optimization of resources and routes in a communication network to provide robust network operation. Our main metrics are the well-known point-to-point resistance distance and network criticality (average resistance distance) of a graph. We show that some of the key performance metrics in a communication network, such as average link betweenness sensitivity or average network utilization, can be expressed as a weighted combination of point-to-point resistance distances. A case of particular interest is when the external demand is specified by a traffic matrix. We extend the notion of network criticality to be a traffic-aware metric. Traffic-aware network criticality is then a weighted linear combination of point-to-point resistance distances of the graph. For this reason, in this chapter, we focus on a weighted linear sum of resistance distances (which is a convex function of link weights) as the main metric and we discuss a variety of optimization problems to jointly assign routes and flows in a network. We provide a complete mathematical analysis of the network planning problem (optimal weight assignment), where we assume that a routing algorithm is already in place and governs the distribution of network flows. Then, we extend the analysis to a more general case involving the simultaneous optimization of resources and flows (routes) in a network. Furthermore, we briefly discuss the problems of finding the best set of demands that can be matched to a given network topology (joint optimization of resources, flows, and demands) subject to the condition that the weighted linear sum of all point-to-point resistance distances of the network should remain below a

A. Tizghadam (✉) • A. Bigdeli • A. Leon-Garcia • H. Naser
University of Toronto (and Lakehead University), Toronto, Canada
e-mail: ali.tizghadam@utoronto.ca; alireza.bigdeli@utoronto.ca;
alberto.leongarcia@utoronto.ca; hnaser@lakeheadu.ca

certain threshold. We discuss applications of the proposed optimization methods to the design of virtual networks. Moreover, we show how our techniques can be used in the design of robust communication networks and robust sparse power grids.

4.1 Introduction

Robustness to the environmental changes (such as topology modification and traffic shift) is a key factor in designing communication networks and power grids. Robustness problem can be studied from different points of view. In some of the robust design approaches, the goal is to find an appropriate metric to quantify the effect of environmental changes, and then the robustness is provided by optimizing the proposed metric. In some other robust designs, the goal is to find an optimization problem (usually in the form of min–max or max–min) to guarantee the insensitivity of the network to the unanticipated changes in environmental parameters.

We are interested in combining these two aspects of the robustness problem. We use a well-known graph metric, random-walk betweenness centrality (for nodes and links of a graph), as the main tool to capture the effect of environmental changes. In this research, the environmental parameters of interest are topology, traffic demand, and community of interest (i.e. the set of nodes, where the external traffic is injected or terminated). We extend the notion of betweenness centrality to the traffic-aware betweenness centrality in which the effect of external traffic demand is explicitly considered. We normalize the traffic-aware betweenness of a node by its weight and we refer to it as traffic-aware node criticality. The average of all traffic-aware node criticalities is referred to as traffic-aware network criticality (TANC).

We show that TANC can capture the effect of our environmental parameters. On the other hand it can be shown that TANC is a linear combination of point-to-point resistance distances of the graph; therefore, we choose to have a linear combination of all point-to-point resistance distances of the graph, which is referred to as weighted network criticality (WNC), as our main robustness metric. We study the mathematical properties of WNC and we discuss a range of optimization problems to find a jointly optimal robust allocation of weights and flows to the links of the network.

Our investigation aims to achieve two goals. First, we study the design of robust networks which involves selecting appropriate topologies and assigning suitable weights in order to provide specified degree of robustness in data networks and power grids. Second, we consider the design of algorithms for robust joint traffic engineering (flow assignment) and network planning (capacity allocation) in communication networks.

The rest of the chapter is structured as follows. Section 17.2 summarizes some of the previous works in the context of robust network design and robust routing. The concept of centrality in graphs is reviewed in Sect. 4.3, followed by a discussion of betweenness centrality in Sect. 4.4. In Sect. 4.5, we first provide a brief discussion of the concepts of resistance distance and network criticality, and then we introduce a new graph metric, TANC, and we show that TANC is a linear combination of point-to-point resistance distances. Motivated by this, we choose to have a linear combination of point-to-point resistance distances as the robustness metric, which will be referred to as WNC, and in Sect. 4.6 we provide a detailed discussion on the minimization of WNC. Section 4.7 is dedicated to the investigation a variety of optimization problems based on WNC to achieve different goals, including resource planning; flow assignment; joint optimization of resources, flows, and demands (while WNC is kept below a certain known value); and *k*-robust network design. Section 4.8 discusses the application of the proposed framework in designing virtual networks; robust capacity planning for communication networks; and robust sparse power grid design. Conclusions are provided in Sect. 4.9.

4.2 Related Works

The topological robustness of network topologies is studied in [1]. Graph-theoretic concepts are used to investigate which network topologies are more robust. Authors argue that "node connectivity" is the most useful metric in graph theory to study the topological robustness of a network. They examine the relationship between node connectivity and the degree of symmetry of the network and they suggest that it is important for robust networks to satisfy *node similarity* and *optimal connectivity* conditions. Two nodes are similar if there is an automorphism that can map one to the other. A network is node similar if all of its nodes are similar. A graph is optimally connected if its node connectivity is equal to the link connectivity metric and both are equal to the minimum node degree of the graph. Ref. [1] investigates the relationship between these conditions, and arrives at the result that a network provides maximum resistance to the node destruction if it is both node-similar and optimally connected. The paper then describes a number of ways to design robust networks satisfying these conditions.

Ref. [2] introduces a new measure of symmetry, symmetry ratio of a network. This metric is defined as the ratio of the number of distinct eigenvalues of the network to the diameter. This metric is used to study the robustness of a network topology in the face of targeted attacks.

The concept of vulnerability is closely related to the robustness. In vulnerability analysis, one tries to find critical parts of a network, whose removal can cause a significant adverse affect on the performance of the network. One of the early solid works on vulnerability is reported in [3], where the authors investigate the response of complex networks subject to attacks on links and nodes of the network. [3] defines some attack scenarios and investigates the behavior of some graph metrics including inverse geodesic distance, shortest-path betweenness centrality, and node degree when attack happens. In [4] an axiomatic approach is used to define appropriate vulnerability functions on graphs. The authors argue that a vulnerability function should be invariant under automorphism. In addition, a vulnerability function has to decrease when a new link is added to the graph. Moreover, a vulnerability function must be computable in polynomial time. Based on these axioms, [4] suggests some vulnerability functions, and investigates their characteristics. The main drawback of [3,4] is that the study is valid for unweighted networks, where the effect of topology on vulnerability is important.

Robustness is also important in traffic engineering. Network operators would like their network to support current and future traffic matrices, even when links and routers fail. Not surprisingly, no backbone network can do this today: It is hard to accurately measure the current matrix, and harder still to predict future ones. Even if the matrices are known, how do we know a network will support them, particularly under failures? As a result, today's networks are designed in a somewhat ad-hoc fashion, using rules-of-thumb and crude estimates of current and future traffic. A wealth of literature is available in tackling this problem and to provide a robust traffic management scheme.

An approach to design backbone networks is proposed in [5] that is insensitive to the traffic matrix (i.e., that works equally well for all valid traffic matrices), and that continues to provide guaranteed performance under a user-defined number of link and router failures. The authors use Valiant Load-Balancing method and argue that it is a promising way to design robust backbone networks. The approach was first proposed by Valiant for processor interconnection networks [6], and has received recent interest for scalable routers with performance guarantees [7, 8]. Ref. [5] applies Valiant method to the backbone network design problem and provides appropriate capacity allocation for the links of a logical full mesh topology to support load-balancing for all possible traffic matrices. In Valiant load-balancing method, traffic destined for a sink d is forwarded to intermediate hops with equal splits to all nodes, and then it is forwarded to the destination d. Delay propagation is one of the shortcomings of this method.

One important category of algorithms in the area of robust traffic engineering is oblivious routing. In oblivious routing, routes are computed to optimize the worst-case performance over all traffic demands, therefore, the computed routes are prepared for dynamic changes in traffic demands. In their pioneering work [9], Applegate and Cohen propose an efficient algorithm to compute the worst case oblivious routing for real networks. They also extend oblivious routing to compute failure scenarios [10]. They found that the oblivious ratio is typically around a factor of 2. A penalty as high as 100% may be acceptable when traffic demands are completely unpredictable, but it is a high cost to pay under predictable demands. In other words, oblivious routing takes a pessimistic point of view and may not be appropriate in relatively stable periods or stable networks.

All of these traffic engineering methods try to maximize a variation of throughput or to minimize a function of network utilization. However, these works are usually oblivious about the mutual effect of topology and external traffic. In our previous works [11, 12], we investigated the problem of traffic engineering from a different perspective. We identified three major types of changes that may affect the performance of the network: network topology, community of interest (active source–destination pairs), and traffic demand. In [11], we used a modified deterministic interpretation of "betweenness centrality" [13], a metric from graph theory which characterizes the topological load of a node or link in a network graph. While the results were encouraging, the analytical study of the results was not feasible. In [12], we used a probabilistic interpretation of betweenness to investigate the problem of designing robust traffic engineering methods. In this new direction, we were able to investigate the problem analytically using metrics from graph-theory. We have discovered some useful aspects of the robust network control and traffic engineering problem in networks. In [12], we considered the effect of external traffic in the weight of a link. We defined the link weight as the available bandwidth of the link (difference of the link capacity and the flow passing through the link).

In the present work, we consider a general traffic matrix applied on the network, and we extend our previous results in two directions. First, we propose a new metric for robustness to decouple the effect of external traffic demand from the link weight. Second, we develop optimization problems to jointly optimize the resources (namely capacities) and routes (flows) such that the robustness is maximized. We start our discussion by introducing the necessary concepts from graph theory.

4.3 Centrality Measures

Centrality measures in graph theory quantify the structural importance or prominence of nodes and links. Numerous centrality indices have been introduced and studied in literature, but according to [14], one can categorize these indices into three major classes: reachability measures, vitality measures, and flow measures. In reachability indices, a node is central if it reaches many other nodes, or can be reached by many other nodes. All the centrality measures in this category use some form of distance between two nodes. For instance, degree centrality which is a well-known reachability index counts the number of nodes that can be reached within distance 1.

Vitality measures are the second class of commonly used centrality indices. Given a real-valued function on a graph G, a vitality measure quantifies the difference between the value of the function on G with the presence or absence of a node or a link. For example, in a wireless mobile network, it is desired to keep the network connected all the time. Algebraic connectivity (the second smallest eigenvalue of the graph Laplacian matrix) is an appropriate connectivity metric for real-valued functions on the graph of the mobile network. Algebraic connectivity is non-zero only if the graph is connected. In the case of algebraic connectivity, the vitality of a node or link denotes the change of the algebraic connectivity, if that node or link was removed from the network.

Finally, we have flow centrality measures. Let $\gamma_s(d)$ denote the amount of flow entering at node s destined for node d. Flow indices quantify how much of this flow traverses a specific node or link. In this chapter, our focus is on flow centralities, and in particular we are interested in different variations of betweenness centrality as the most useful flow measure. We refer the reader to [14] for a complete review of centrality measures.

4.4 Betweenness Centrality Measures

Our work in this chapter is based on the concept of betweenness centrality, which can be categorized as a flow centrality measure. We first provide necessary background and required definitions. Then, in Sect. 4.4.1, we derive a useful relationship between a specific type of betweenness (i.e. random-walk betweenness) and packet networks, justifying the usefulness of the betweenness measure in the study of communication networks.

Freeman [13] introduced a very useful metric in graph theory referred to as shortest-path betweenness centrality. For node k the shortest-path betweenness centrality with respect to flows from source node s to destination node d is defined as the proportion of instances of the shortest paths from node s to d that traverse node k. This can be interpreted as the probability that node k is involved in any communication between s and d with the assumption that all communication is along equiprobable shortest paths. The overall shortest-path betweenness centrality of node k is the sum of the centralities over all source–destination pairs. Link betweenness is defined similarly. The concept of betweenness centrality is closely related to the principle of conservation of flow in a communication network. According to this principle for any intermediate node (or link) in a communication path, the incoming flow is equal to the outgoing flow. When we count the proportion of instances that a node is involved in a communication along shortest paths (or more generally a path), we implicitly assume the conservation of flow.

A major drawback of the shortest-path betweenness is that it does not consider the information that can be obtained from other paths of the network. In communications networks, it is frequently desirable to take a path other than the shortest path. To overcome this issue, other betweenness centrality metrics have been proposed. In [15] the authors introduce flow betweenness centrality. Suppose that each link of the network is capable of transferring a unit flow of information. The flow betweenness of a node k is defined as the proportion of flow through node k when maximum flow is transmitted from source s to destination d averaged over all $s-d$ pairs. The maximum flow that can be sent between two nodes is in general more than a unit flow since one can use multiple paths to transmit information. The flow betweenness is in fact a vitality measure because the amount of flow passing along node k can be found in the following way. Let $f_s(d)$ denote the maximum flow that can be transmitted from source node s to destination d. Further, suppose we remove node k (and its incident links) from the graph, and let $f_s^k(d)$ denote the maximum flow that can be sent from s to d in the new graph. Then the flow traversing node k is: $f_s(d) - f_s^k(d)$. In fact, the flow betweenness measures the betweenness centrality of network nodes when maximum possible flows are fed into the network (between every pair of nodes). While the flow betweenness considers paths other than shortest paths, it suffers from some of the limitations in the definition of shortest path betweenness. In maximum flow problems, we still have one (or more) ideal path(s) mandating the communication (just like shortest path); however, in many practical situations flow does not take ideal paths either shortest path, max-flow path, or any other type of ideal path.

Deterministic betweenness is a straightforward extension of shortest path betweenness and is proposed in [11]. Deterministic betweenness of a node (or link) k is the fraction of total paths between a source s and destination node d traversing node (or link) k, averaged over all active traffic sources and sinks, which we refer to as the community of interest. One can easily recognize two main differences between deterministic betweenness and the original shortest-path betweenness. First, in the former all the paths are involved, whereas in the latter only shortest paths are considered. Second, in deterministic betweenness only the active path set (paths within a community of interest) is involved in the definition of betweenness, whereas in the original shortest path betweenness, all possible node pairs are considered. Unfortunately, the enumeration of paths does not lend itself to tractable analytic results.

In order to overcome these tractability problems, Newman [16] proposed random-walk betweenness which is a probabilistic approach to define and analyze the betweenness of a node/link. We begin with a graph model and we define necessary notations, which will be used throughout.

An undirected graph $G(V,E,W)$ consists of a finite node set V which contains n nodes, together with a link set $E \in V \times V$, and each link has an associated non-negative weight w_{ij}. The weight of a node i is defined as $W_i = \sum_j w_{ij}$. We can define a transition probability matrix $P = [p_{ij}]$ of an irrecucible Markov random walk through the undirected graph which satisfies $\sum_j p_{ij} = 1 \; \forall i \in V$. Moreover, we define weighted graph Laplacian as $L = D - W$, where D is a diagonal matrix whose main diagonal entries are: $D(i,i) = W_i$.

We are interested in a special type of random-walks referred to as weight-based random-walk. The weight-based random-walk is defined on a Markov chain with transition probability matrix P according to the following rule:

$$p_{sk}(d) = \frac{w_{sk}}{W_s}(1 - \delta(s-d)) \tag{4.1}$$

where $A(s)$ is the set of adjacent nodes of s and w_{sk} is the weight of link (s, k) (if there is no link between node s and k, then $w_{sk} = 0$), and $\delta(.)$ is the Kronecker delta function (i.e. $\delta(x) = 1$ if $x = 0$ and $\delta(x) = 0$ otherwise). The delta function in (4.1) is due to the fact that the destination node d is an absorbing node, and any random-walk coming to this state, will be absorbed or equivalently $p_{dk}(d) = 0$. Clearly, (4.1) defines a Markovian random-walk.

We are now ready to define random-walk betweenness. Consider the set of trajectories that begin at node s and terminate when the walk first arrives at node d, that is, destination node d is an absorbing node. The random-walk betweenness $b_{sk}(d)$ of node k for the $s-d$ trajectories is defined as the number of times node k is visited in trajectories from s to d averaged over all possible $s-d$ trajectories and the total betweenness of node k is $b_k = \sum_{s,d} b_{sk}(d)$.

Let $B_d = [b_{sk}(d)]$ be the $n \times n$ matrix of betweenness metrics of node k for walks that begin at node s and end at node d. Note that the dth row of the matrix is zero. It is shown in [16] that matrix B_d can be written as

$$B_d = (I - P_d)^{-1}\Theta_d \tag{4.2}$$

$$\Theta_d = [\theta_{sk}(d)] = \begin{cases} 1 \text{ if } s = k \neq d \\ 0 \text{ otherwise} \end{cases}$$

Matrix P_d is the same as P except that its dth row and dth column are zero vectors.

Finally, traffic-aware betweenness [17] is a natural extension to explicitly account for the effect of traffic demands in the definition of betweenness centrality index. Let $\Gamma = [\gamma_s(d)]$ and γ denote the traffic matrix and the total external traffic (i.e. $\gamma = \sum_{s,d} \gamma_s(d)$) respectively. We define traffic-aware betweenness (TAB) of node k as:

$$b'_{sk}(d) = b_{sk}(d) + \frac{\gamma_s(d)}{\gamma} b_{sk}(d)$$

$$b'_{sk}(d) = \left(1 + \frac{\gamma_s(d)}{\gamma}\right) b_{sk}(d) \tag{4.3}$$

$$b'_k = \sum_{s,d} \left(1 + \frac{\gamma_s(d)}{\gamma}\right) b_{sk}(d) \tag{4.4}$$

If traffic matrix is zero, then we have the original topological betweenness as a special case. This definition is quite general and applicable for all types of betweenness. If we consider $b_{sk}(d)$ as the shortest-path betweenness of node k for source–destination pair sd, then we will have traffic-aware shortest-path betweenness, and so on.

4.4.1 Random-Walk Betweenness in Data Networks

Random-walk betweenness is closely related to packet network models. We consider a packet switching network in which external packets arrive to packet switches according to independent arrival processes. Each packet arrival has a specific destination and the packet is forwarded along the network until it reaches the destination. We assume that packet switches are interconnected by transmission lines that can be modeled as single-server queues. Furthermore, we suppose that packet switches use a form of routing where the proportion of packets at queue i forwarded to the next-hop queue j is p_{ij}.

We calculate the total arrival/departure rate of the traffic to/from each node. The total input rate of node k (internal plus external) is denoted by x_k. After receiving service at the ith node, the proportion of customers that proceed to node k is p_{ik}. To find x_k we need to solve the following set of linear equations (see [24]):

$$x_k = \gamma_k + \sum_{i=1}^{n} x_i p_{ik} \quad \forall k \in V \tag{4.5}$$

where γ_k is the external arrival rate to node k. Note that (4.5) is essentially the KCL (Kirchhoff's Current Law). If we denote $\overrightarrow{x} = [x_1, x_2, ..., x_n]$ and $\overrightarrow{\gamma} = [\gamma_1, \gamma_2, ..., \gamma_n]$, then (4.5) becomes:

$$\overrightarrow{x} = \overrightarrow{\gamma} + \overrightarrow{x} P \tag{4.6}$$

Suppose we focus on traffic destined to node d, then node d is an absorbing node, and we suppose that the arrival rate at node d is zero (since these arrivals do not affect other nodes) and (4.6) can be written as:

$$\overrightarrow{x_d} = (\overrightarrow{\gamma_d} + \overrightarrow{x_d} P_d) \times \Theta_d \tag{4.7}$$

where $\overrightarrow{x_d}$ and $\overrightarrow{\gamma_d}$ are the same as $\overrightarrow{x}$ and $\overrightarrow{\gamma}$ except for the dth element which is 0. Matrix P_d is also the same as P except that its dth row and dth column are zero vectors. Equation (4.7) can be solved for $\overrightarrow{x_d}$

$$\overrightarrow{x_d} = \overrightarrow{\gamma_d} \times \Theta_d \times (I - P_d \times \Theta_d)^{-1} \tag{4.8}$$

To find the relationship of betweenness B_d and the input arrival rate x_k we notice that $p_{dk}(d) = 0$ which means that $P_d = \Theta_d \times P_d$. Thus,

$$\begin{aligned} P_d \times \Theta_d &= \Theta_d \times P_d \times \Theta_d \\ \Theta_d - P_d \times \Theta_d &= \Theta_d - \Theta_d \times P_d \times \Theta_d \\ (I - P_d) \times \Theta_d &= \Theta_d \times (I - P_d \times \Theta_d) \\ \Theta_d \times (I - P_d \times \Theta_d)^{-1} &= (I - P_d)^{-1} \times \Theta_d \end{aligned}$$

Using (4.2) we will have

$$\Theta_d \times (I - P_d \times \Theta_d)^{-1} = B_d \tag{4.9}$$

We substitute (4.9) in (4.8) and obtain the relationship between the node traffic and node betweenness,

$$\overrightarrow{x_d} = \overrightarrow{\gamma_d} \times B_d \tag{4.10}$$

If we denote the kth element of $\overrightarrow{x_d}$ and $\overrightarrow{\gamma_d}$ by $x_k(d)$ and $\gamma_k(d)$ respectively, we have

$$x_k(d) = \sum_{s} \gamma_s(d) b_{sk}(d) \tag{4.11}$$

Now we can find the total load at node k by adding the effect of all destinations in (4.11),

$$x_k = \sum_d x_k(d) = \sum_{s,d} \gamma_s(d) b_{sk}(d) \tag{4.12}$$

It is constructive to establish the relationship of node betweenness and node traffic in a more intuitive way. Consider the traffic generated by packets that arrive at s and are destined for d. Each packet in this flow generates $b_{sk}(d)$ arrivals on average at node k. Let $\gamma_s(d)$ be the number of external packets per second that arrive at node s with destination d. Over a large number of such trials, say N, the average number of times node k is visited will be approximately $N \times b_{sk}(d)$. Suppose that it takes T seconds to have N arrivals at node s, then the average number of visits per second to node k is

$$\frac{N \times b_{sk}(d)}{T} = \gamma_s(d) \times b_{sk}(d),$$

since the average arrival rate at s for d is approximately N/T.

We only consider external arrivals with destinations other that the originating node so $\gamma_d(d) = 0$. The total traffic $x_{sk}(d)$ generated by the $s-d$ flow at node k is then $\gamma_s(d) \times b_{sk}(d)$, where s is not equal to d. Recalling that $b_{sd}(d) = 1$, we obtain:

$$x_{sk}(d) = \begin{cases} \gamma_s(d) b_{sk}(d) & \text{if } s \neq d \;\&\; d \neq k \\ \gamma_s(d) & \text{if } s \neq d \;\&\; k = d \\ 0 & \text{if } s = d \end{cases}$$

The total traffic into node k is obtained by summing over all s and d, with s not equal to d

$$\begin{aligned} y_k &= \sum_{s,d} x_{sk}(d) \\ &= \sum_{s \neq k} \gamma_s(k) + \sum_{s \neq d} \sum_{d \neq k} \gamma_s(d) b_{sk}(d) \end{aligned} \tag{4.13}$$

The first sum on the right hand side of (4.13) is the total network packet arrival rate destined for k, that is, the total flow absorbed at node k. The second term is the total traffic that flows across queue k, that is, the flow through k that originates at nodes other than d and that are not destined for k. This second term, the transit flow through queue k, accounts for the effect of the network topology, so we let x_k denote this flow:

$$x_k = \sum_{d \neq k} \gamma_k(d) b_{kk}(d) + \sum_{s \neq k} \sum_{d \neq k} \gamma_s(d) b_{sk}(d) \tag{4.14}$$

The first sum in (4.14) is the arrivals at k destined for d, including revisits. The second sum is the total transit traffic through k that did not originate locally. x_k can be viewed as a measure of betweenness of queue k that takes the different arrival rates into account.

Suppose that different queues have different total external arrival rates but the fraction of external traffic destined for d does not depend on s, i.e.,

$$\gamma_s(d) = \gamma_s a_d$$

where

$$a_d \geq 0, \ \sum_d a_d = 1$$

The total traffic through queue k is then

$$\begin{aligned} x_k &= \sum_{d \neq k} \gamma_k a_d b_{kk}(d) + \sum_{s \neq k} \gamma_s \left[\sum_{d \neq k} a_d b_{sk}(d) \right] \\ &= \gamma_k \left[\sum_{d \neq k} a_d b_{kk}(d) \right] + \sum_{s \neq k} \gamma_s \left[\sum_{d \neq k} a_d b_{sk}(d) \right] \end{aligned} \tag{4.15}$$

The terms inside the square brackets in (4.15) can be viewed as betweenness measures that have been weighted by the differential preferences for destinations according to a_d. These weighted betweenness measures are in turn scaled according to the arrival rates at different queues.

In the case where arriving packets are equally likely to be destined to any destination (other than the arriving node), we have $a_d = \frac{1}{n-1}$, so

$$\begin{aligned} x_k &= \frac{\gamma_k}{n-1} \left[\sum_{d \neq k} b_{kk}(d) \right] + \sum_{s \neq k} \frac{\gamma_s}{n-1} \left[\sum_{d \neq k} b_{sk}(d) \right] \\ &= \gamma_k \bar{b}_{kk} + \sum_{s \neq k} \gamma_s \bar{b}_{sk} \end{aligned} \tag{4.16}$$

Finally suppose that the arrival rate at every node is equal, that is, $\gamma_s = \gamma/n$, where γ is the total external packet arrival rate to the network, then

$$\begin{aligned} x_k &= \frac{\gamma}{n(n-1)} \left(\sum_{d \neq k} b_{kk}(d) \right) + \frac{\gamma}{n(n-1)} \sum_{s \neq k} \sum_{d \neq k} b_{sk}(d) \\ &= \frac{\gamma}{n(n-1)} b_k \end{aligned} \tag{4.17}$$

where $b_k = \sum_s \sum_{d \neq k} b_{sk}(d)$ is the random walk betweenness for node k. In summary, we have derived the following theorem.

Theorem 4.1. *Consider a network with n nodes, and assume that the average traffic rate on all of the nodes is $\gamma_n = \gamma/n$ where γ is the total external input traffic rate of the network. Let x_k be the total arrival rate of a node k and b_k be the total betweenness of this node, then*

$$x_k = \frac{\gamma_n}{n-1} b_k = \frac{\gamma}{n(n-1)} \times b_k$$

Equation (4.12) and Theorem 4.1 clearly show that the traffic of a node in a communication network can be captured with the notion of random-walk betweenness centrality.

4.5 Network Criticality and Resistance Distance

We now briefly explain the notion of *network criticality*, which is discussed in [12] to quantify the robustness of a network. We start by defining node/link criticality.

Definition 4.1. Node criticality is defined as the random-walk betweenness of a node normalized by its weight value. In other words, node criticality is the node random-walk betweenness divided by the node weight. Likewise, link criticality is defined as the betweenness of the link over its weight.

Let η_k be the criticality of node k and η_{ij} be the criticality of link $l = (i, j)$. It is shown in [12] that η_i and η_{ij} can be obtained by the following expressions:

$$\eta_k = \frac{b_k}{W_k} = \frac{1}{2}\tau \tag{4.18}$$

$$\eta_{ij} = \frac{b_{ij}}{w_{ij}} = \tau \tag{4.19}$$

τ in the above equations is defined as follows:

$$\tau = \sum_s \sum_d \tau_{sd} \tag{4.20}$$

$$\tau_{sd} = l^+_{ss} + l^+_{dd} - 2l^+_{sd} \ \ or \ \ \tau_{sd} = u^t_{sd} L^+ u_{sd} \tag{4.21}$$

where $L^+ = [l^+_{ij}]$ is the Moore–Penrose inverse of graph Laplacian matrix L [18], n is the number of nodes, and $u_{ij} = [0 \ldots 1 \ldots -1 \ldots 0]^t$ (1 and -1 are in ith and jth positions, respectively).

Equations (4.18)–(4.21) show that node criticality (η_k) and link criticality (η_{ij}) are independent of the node/link position and only depend on τ, which is a global quantity of the network.

Definition 4.2. We refer to τ_{sd} as *point-to-point network criticality* and τ as *network criticality*.

For future references, we also write the explicit relationship between point-to-point network criticality and random-walk betweenness centrality

$$\frac{b_{sk}(d)}{W_k} = l^+_{dd} - l^+_{sd} - l^+_{dk} + l^+_{sk} \tag{4.22}$$

$$\tau_{sd} = \frac{b_{sk}(d) + b_{dk}(s)}{W_k} \tag{4.23}$$

The proof of (4.22) and (4.23) can be found in [12].

One can see that τ is a global quantity on the network graph. Equations (4.18) and (4.19) show that node (link) betweenness consists of a local parameter (weight) and a global metric (network criticality). τ can capture the effect of topology through the betweenness values. A higher node/link betweenness results in a higher risk (criticality) in using the node/link. Furthermore, one can define node/link capacity as the weight of a node/link, then increasing the weight of a node/link will decrease the risk of using the node/link. Therefore, network criticality can quantify the risk of using a node/link in a network which in turn indicates the degree of robustness of the network. For comparison purposes, we may use the average network criticality, which is defined as $\hat{\tau} = \frac{1}{n(n-1)}\tau$.

Network criticality can be interpreted as the total resistance of a corresponding electrical network. Consider an electrical circuit with the same graph as our original network graph, and with link resistances equal to the reciprocal of link weights. It can be shown that the network criticality is equal to the total resistance distance (effective resistance) [22] seen between different pairs of nodes in the electrical circuit. A high network criticality is an indication of high resistance in the equivalent electrical circuit, therefore, in two networks with the same number of nodes, the one with lower network criticality is better connected, hence better positioned to accommodate network flows.

Network criticality can also quantify the sensitivity of a network to the environmental changes. It has been shown that network criticality equals the average of link betweenness sensitivities, where link betweenness sensitivity is defined as the partial derivative of link betweenness with respect to the corresponding link weight [19],

$$\tau = \frac{1}{m-1} \sum_{(i,j)\in E} \frac{\partial b_{ij}}{\partial w_{ij}} \tag{4.24}$$

where m denotes the number of links of the network. Equation (4.24) states that the minimization of network criticality results in the minimization of the average sensitivity of link betweennesses with respect to the changes in the corresponding link weights (which in turn captures sensitivity to environmental changes). Therefore, a control algorithm for minimum network criticality balances the betweennesses of the links in such a way as to keep the average sensitivity below a desired level. From another point of view, the lower the network criticality, the better distributed is the traffic between all the links of a network, and the better balanced is the load of the traffic among all active links. This implies better fairness in routing the traffic in the nodes of the network. More detailed information on properties and interpretations of the network criticality can be found in [20, 21].

Another advantage of having low network criticality is the robustness enhancement of the network. Suppose that a node is failing or becoming inaccessible so that it cannot route the traffic passing through it. If we adapt the routing to minimize the criticality, the result is to adjust the betweenness in such a way that traffic is re-routed to other nodes instead of the impaired one and that the resulting flows provide higher robustness against additional unpredictable deleterious situations.

It has been shown that τ_{sd} is a convex function of link weights and τ is a strictly convex function of link weights [23].

Definition 4.3. In analogy to (4.18), we define traffic-aware node criticality (TANOC) as the traffic-aware node betweenness normalized by node weight (sum of the link weights incident to the node),

$$\begin{aligned}\tau'_{sk}(d) &= 2\frac{b'_{sk}(d)}{W_k}, \text{ where } W_k = \sum_j w_{kj} \\ \tau'_k &= 2\frac{b'_k}{W_k}\end{aligned} \tag{4.25}$$

Furthermore, we define the TANC as the average traffic-aware node criticality:

$$\tau' = \frac{1}{n}\sum_k \tau'_k \tag{4.26}$$

4.5.1 Network Utilization and Network Criticality

In this section, we derive a general expression for the average network utilization (and individual node utilization). Theorem 4.1 establishes a connection between the load of a node and its betweenness when the average input rate to all the nodes is uniform. In general, for a traffic matrix $\Gamma = [\gamma_i(j)]$ the utilization of a node can be expressed as a linear combination of point-to-point network criticalities subject to considering weight-based random-walks as defined in (4.1). To see this consider, (4.11):

$$\begin{aligned}x_k &= \sum_{s,d} \gamma_s(d) b_{sk}(d) \\ &= \frac{1}{2}\sum_{s,d}(\gamma_s(d)b_{sk}(d) + \gamma_d(s)b_{dk}(s))\end{aligned}$$

$$= \frac{1}{2}\sum_{s,d}(\gamma_s(d)b_{sk}(d) + \gamma_d(s)(W_k\tau_{sd} - b_{sk}(d)))$$

$$= \frac{W_k}{2}\sum_{s,d}\gamma_d(s)\tau_{sd} + \frac{1}{2}\sum_{s,d}(\gamma_s(d) - \gamma_d(s))b_{sk}(d) \tag{4.27}$$

where we have used (4.23) to obtain (4.27). Now we write $b_{sk}(d)$ in terms of different elements of matrix $\Omega = [\tau_{sd}]$. We have

$$\tau_{sd} = l^+_{ss} + l^+_{dd} - 2l^+_{sd}$$
$$\tau_{dk} = l^+_{dd} + l^+_{kk} - 2l^+_{dk}$$
$$\tau_{sk} = l^+_{ss} + l^+_{kk} - 2l^+_{sk}$$

Considering (4.22), we have

$$\tau_{sd} + \tau_{dk} - \tau_{sk} = 2(l^+_{dd} - l^+_{sd} - l^+_{dk} + l^+_{sk})$$
$$= 2\frac{b_{sk}(d)}{W_k}$$
$$b_{sk}(d) = \frac{W_k}{2}(\tau_{sd} + \tau_{dk} - \tau_{sk}) \tag{4.28}$$

Using (4.28) in (4.27), we have

$$x_k = \frac{W_k}{2}\sum_{s,d}\gamma_d(s)\tau_{sd} + \frac{W_k}{4}\sum_{s,d}(\gamma_s(d) - \gamma_d(s))(\tau_{sd} + \tau_{dk} - \tau_{sk})$$

$$\frac{x_k}{W_k} = \frac{1}{4}\sum_{s,d}(\gamma_s(d) + \gamma_d(s))\tau_{sd} + \frac{1}{4}\sum_{s,d}(\gamma_s(d) - \gamma_d(s))(\tau_{dk} - \tau_{sk}) \tag{4.29}$$

Node utilization is defined as the load of a node normalized by its capacity (or in a more general sense by its weight). We denote the utilization of node k by $V_k = \frac{x_k}{W_k}$ and the average network utility by $\bar{V} = \frac{\sum_k V_k}{n}$. For the average network utilization $\bar{V}$ we have

$$\bar{V} = \frac{1}{4}\sum_{s,d}(\gamma_s(d) + \gamma_d(s))\tau_{sd} + \frac{1}{4n}\sum_{s,d}(\gamma_s(d) - \gamma_d(s))(\tau_{d*} - \tau_{s*}) \tag{4.30}$$

where $\tau_{i*} = \sum_k \tau_{ik}$. Equation (4.30) can be simplified as

$$\bar{V} = \sum_{s,d}\beta_{sd}\tau_{sd} \tag{4.31}$$

where

$$\beta_{sd} = \frac{\gamma_s(d)+\gamma_d(s)}{4} + \frac{\gamma_{*s}-\gamma_{s*}}{2n}.$$

We can easily express the average network utilization $\bar{V}$ in terms of network criticality and TANC. Since $x_k = \sum_{sd} \gamma_s(d) b_{sk}(d)$, from (4.4) we conclude that

$$b'_k = b_k + \frac{1}{\gamma} x_k$$

$$V_k = \frac{x_k}{W_k} = \gamma \frac{b'_k - b_k}{W_k}$$

$$V_k = \frac{\gamma}{2}(\tau'_k - \tau)$$

Finally,

$$\bar{V} = \frac{1}{n}\sum_{k=1}^{n} V_k = \frac{\gamma}{2}(\tau' - \tau) \tag{4.32}$$

Proceeding as we did for $\bar{V}$, one can see that:

$$\tau' = \sum_{s,d}\left(1 + \frac{\gamma_s(d)+\gamma_d(s)}{2\gamma} + \frac{\gamma_{*s}-\gamma_{s*}}{n\gamma}\right)\tau_{sd} \tag{4.33}$$

It will be easily verified that the coefficients of τ_{sd} in (4.33) (i.e. $1 + \frac{\gamma_s(d)+\gamma_d(s)}{2\gamma} + \frac{\gamma_{*s}-\gamma_{s*}}{n\gamma}$) are always non-negative; therefore, TANC is a convex function of link weights since τ_{sd} is always convex. Consequently, form (4.32) one can see that the average network utilization is in most general form the difference of two convex functions (or equivalently the sum of a convex and a concave function). Minimizing the difference of two convex functions can be converted to a convex maximization problem, which can be numerically solved with methods like branch-and-bound [25].

We can find a subset of traffic matrices, for which the average network utilization $\bar{V}$ is pure convex. In fact the average network utilization is a convex function of link weights if and only if in (4.31) we have $\forall s,d \beta_{sd} + \beta_{ds} \geq 0$ (note that $\tau_{sd} = \tau_{ds}$), or:

$$\gamma_s(d) + \gamma_d(s) \geq \frac{1}{n}(\gamma_{s*} - \gamma_{*s} + \gamma_{d*} - \gamma_{*d}) \;\; \forall\, s,d \in V \tag{4.34}$$

Inequality (4.34) defines a subset of all possible traffic matrices for which the average network utilization is convex. We examined condition set (4.34) for real traffic matrix traces recorded form Abilene network [26]. These traces can be fond in TOTEM project [27] website. The condition set (4.34) was satisfied for the majority of traffic matrices that we examined; therefore, we could assume that for these real traffic traces, the average network utilization is a convex function of weights. In the following, we assume that the traffic matrices are within this subset.

This motivates the rest of this chapter. In order to minimize the average network utilization (or to minimize the maximum of node utilization), we have to effectively solve a convex optimization problem, which is investigated in next section.

4.6 Minimizing WNC

We first consider a general weighted version of network criticality (WNC) defined as follows:

$$\tau_\alpha = \sum_{i,j} \alpha_{ij} \tau_{ij}, \;\; \forall i,j \in N \;\; \alpha_{ij} + \alpha_{ji} \geq 0 \tag{4.35}$$

Obviously, the average network utilization and individual node utilization are special cases of WNC by appropriate selection of coefficients. To study the minimization of WNC, we rewrite WNC in a matrix form as follows:

$$\begin{aligned} \tau_\alpha &= \sum_{i,j} \alpha_{ij} \tau_{ij} \\ &= \sum_{ij} \alpha_{ij} u_{ij}^t L^+ u_{ij} \\ &= \sum_{ij} \alpha_{ij} \mathrm{Tr}(u_{ij} u_{ij}^t L^+) \\ &= \mathrm{Tr}(U_\alpha L^+) \end{aligned} \tag{4.36}$$

where $U_\alpha = \sum_{ij} \alpha_{ij} U_{ij}$, $U_{ij} = u_{ij} u_{ij}^t$, and $\mathrm{Tr}(A)$ denotes the trace of matrix A.

It is easy to see that the sum of the rows in U_α is zero, and for $\alpha_{ij} \geq 0$ it is a symmetric and positive semidefinite matrix. One example of U_α for $n = 3$ (number of nodes) is given in the following:

$$U_\alpha = \begin{pmatrix} \alpha'_{12} + \alpha'_{13} & -\alpha'_{12} & -\alpha'_{13} \\ -\alpha'_{12} & \alpha'_{12} + \alpha'_{23} & -\alpha'_{23} \\ -\alpha'_{13} & -\alpha'_{23} & \alpha'_{13} + \alpha'_{23} \end{pmatrix}$$

where $\alpha'_{ij} = \alpha_{ij} + \alpha_{ji}$.

In order to continue, we need the following preposition.

Proposition 4.2 *For any non-singular square $(n \times n)$ matrix X and any $n \times n$ matrices A and B, we have*

$$\frac{\mathrm{d}}{\mathrm{d}X} \mathrm{Tr}(AX) = A$$

$$\frac{\mathrm{d}}{\mathrm{d}X} \mathrm{Tr}(AX^{-1}B) = -X^{-1}BAX^{-1}$$

where in general the derivative $\frac{\mathrm{d}}{\mathrm{d}X}f(X)$ *of a scalar-valued differentiable function* $f(X)$ *of a matrix argument* $X \in R^{p\times q}$ *is the* $q \times p$ *matrix whose* (i,j)*th entry is* $\frac{\partial f(X)}{\partial X(j,i)}$ *[28].*

Proof. See [28].

We now consider the minimization of WNC. First, we show that the minimization is viable. To this end we need the following lemma.

Lemma 4.1. *The partial derivative of* τ_α *with respect to link weight* w_{ij} *is always non-positive and can be obtained from the following equation.*

$$\frac{\partial \tau_\alpha}{\partial w_{ij}} = -\left\|F_\alpha L^+ u_{ij}\right\|^2$$

where F_α *is a matrix such that* $U_\alpha = F_\alpha^t F_\alpha$*. This decomposition is always possible because* U_α *is a positive semidefinite matrix.*

Proof. Let $\Gamma = L + \frac{J}{n}$ where J is a square $n\times n$ matrix with all entries equal to 1, then $L^+ = \Gamma^{-1} - \frac{J}{n}$ [28]. Note that $U_\alpha J = 0$, consequently $\mathrm{Tr}(U_\alpha L^+) = \mathrm{Tr}(U_\alpha \Gamma^{-1})$. We use Proposition 4.2 to derive the result

$$\begin{aligned}
\frac{\partial \tau_\alpha}{\partial w_{ij}} &= \frac{\partial \mathrm{Tr}(U_\alpha \Gamma^{-1})}{\partial w_{ij}} \\
&= -\mathrm{Tr}(U_\alpha \Gamma^{-1}\frac{\partial \Gamma}{\partial w_{ij}}\Gamma^{-1}) \\
&= -\mathrm{Tr}(U_\alpha \Gamma^{-1} u_{ij}u_{ij}^t \Gamma^{-1}) \\
&= -\mathrm{Tr}(F_\alpha^t F_\alpha \Gamma^{-1} u_{ij}u_{ij}^t \Gamma^{-1}) \\
&= -\mathrm{Tr}(F_\alpha L^+ u_{ij}u_{ij}^t L^+ F_\alpha^t) \\
&= -\mathrm{Tr}((F_\alpha L^+ u_{ij})(F_\alpha L^+ u_{ij})^t) \\
&= -\mathrm{Tr}((F_\alpha L^+ u_{ij})^t(F_\alpha L^+ u_{ij})) \\
&= -\left\|F_\alpha L^+ u_{ij}\right\|^2
\end{aligned}$$

Since WNC is a convex function and its derivative with respect to the weights is always non-positive (according to Lemma 4.1), the minimization of τ_α subject to some convex constraint set is possible.

In formulating the optimization problem, we add a maximum budget constraint to the problem. We assume that there is a cost z_{ij} to deploy each unit of weight on link (i,j). We also assume that there is a maximum budget of C to spend across all network links. This constraint means that $\sum_{(i,j)\in E} w_{ij}z_{ij} \le C$. Now we can write our optimization problem as follows:

$$\begin{aligned} &\mathit{Minimize} \quad \tau_\alpha \\ \mathit{Subject\ to} \quad &\textstyle\sum_{(i,j)\in E} w_{ij} z_{ij} \le C, \quad C \mathit{\ is\ fixed} \\ &w_{ij} \ge 0 \;\; \forall (i,j) \in E \end{aligned} \tag{4.37}$$

Again, we let $\Gamma = L + \frac{J}{n}$. Considering the fact that $L = \sum_{i,j} w_{ij} u_{ij} u_{ij}^t$ (definition of Laplacian), we can write the optimization problem (4.37) as:

$$\begin{aligned} &\text{Minimize} \quad \mathrm{Tr}(U_\alpha L^+) \\ \text{Subject to} \quad &\Gamma = \textstyle\sum_{(i,j)\in E} w_{ij} u_{ij} u_{ij}^t + \frac{J}{n} \\ &L^+ = \Gamma^{-1} - \tfrac{J}{n} \\ &\textstyle\sum_{(i,j)\in E} w_{ij} z_{ij} = C, \quad C \text{ is fixed} \\ &w_{ij} \ge 0 \;\; \forall (i,j) \in E \end{aligned} \tag{4.38}$$

In order to find the condition of optimality, we need the following lemma.

Lemma 4.2. *For any weight matrix W of the links of a graph:* $\mathrm{Vec}(W)^t \nabla \tau + \tau = 0$, *where* $\mathrm{Vec}(W)$ *is a vector obtained by concatenating all the rows of matrix W to get a vector of w_{ij}'s.*

Proof. In [23], it has been shown that if we scale all the link weights with t, the effective resistance τ_{ij} will scale with $1/t$. Since τ_α is a linear combination of point-to-point effective resistances, τ_α will also scale with $1/t$:

$$\tau_\alpha(t\mathrm{Vec}(W)) = \frac{1}{t} \tau_\alpha(\mathrm{Vec}(W)) \tag{4.39}$$

By taking the derivative of τ with respect to t, we have

$$\mathrm{Vec}(W)^t \nabla \tau_\alpha = \frac{-1}{t^2} \tau_\alpha(W) \tag{4.40}$$

It is enough to consider (4.40) at $t = 1$ to get $\mathrm{Vec}(W)^t \nabla \tau_\alpha + \tau_\alpha = 0$.

Now we are ready to state the condition of optimality.

Lemma 4.3. *The condition of optimality for optimization problem (4.37) can be written as*

$$\min_{(i,j)\in E} \frac{C}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}} + \tau_\alpha \ge 0$$

Moreover,

$$w_{ij}^* \left(C \frac{\partial \tau_\alpha}{\partial w_{ij}} + z_{ij} \tau_\alpha \right) = 0 \;\; \forall (i,j) \in E \tag{4.41}$$

where w_{ij}^* denotes the optimal weight for link (i,j).

Proof. In general, one can apply the condition of optimality [31,32] on optimization problem (4.37) to derive the necessary condition for a weight vector to be optimal. Let W^* be the optimal weight matrix and let W_t be another weight matrix satisfying the constraints of optimization problem (4.37), then according to the condition of optimality,

$$\nabla \tau_\alpha.(\mathrm{Vec}(W_t) - \mathrm{Vec}(W^*)) \geq 0$$

Now, we choose W_t as follows:

$$W_t = [w_{uv}] = \begin{cases} \frac{C}{2z_{ij}} & \text{if } u = i \ \& \ v = j \\ \frac{C}{2z_{ij}} & \text{if } u = j \ \& \ v = i \\ 0 & \text{otherwise} \end{cases}$$

Clearly, W_t satisfies the constraints of optimization problem (4.37); therefore, using the condition of optimality and considering Lemma 4.2 we have

$$\begin{aligned} \nabla \tau_\alpha.(\mathrm{Vec}(W_t) - \mathrm{Vec}(W^*)) &\geq 0 \\ \nabla \tau_\alpha.\mathrm{Vec}(W_t) - \nabla \tau_\alpha.\mathrm{Vec}(W^*) &\geq 0 \\ \frac{C}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}} + \tau_\alpha &\geq 0 \quad \forall (i,j) \in E \\ \min_{(i,j)\in E} \frac{C}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}} + \tau_\alpha &\geq 0 \end{aligned} \tag{4.42}$$

Now, to prove the second part of the theorem we write the constraint of the optimization problem as an inner product of costs and weights,

$$(\mathrm{Vec}(Z).\mathrm{Vec}(W^*))\tau_\alpha = \left(\sum_{(i,j)\in E} w^*_{ij} z_{ij} \right) \tau_\alpha = C\tau_\alpha \tag{4.43}$$

Combining Lemma 4.2 and (4.43) one can see that

$$\begin{aligned} C\nabla \tau_\alpha.\mathrm{Vec}(W^*) + \mathrm{Vec}(Z).\mathrm{Vec}(W^*)\tau_\alpha &= 0 \\ \mathrm{Vec}(W^*).(C\nabla \tau_\alpha + \tau_\alpha \mathrm{Vec}(Z)) &= 0 \\ w^*_{ij} \left(C \frac{\partial \tau_\alpha}{\partial w_{ij}} + \tau_\alpha z_{ij} \right) &= 0 \end{aligned}$$

This completes the proof of Lemma 4.3.

Lemma 4.4. *The dual of the optimization problem (4.38) is as follows:*

$$\begin{array}{ll} \text{maximize} & \frac{1}{C}\mathrm{Tr}^2(U_\alpha X) \\ \text{subject to} & \frac{1}{\sqrt{z_{ij}}} \left\| F_\alpha X u_{ij} \right\| \leq 1 \ \ \forall (i,j) \in E \end{array}$$

$$X\overrightarrow{1} = 0$$

$$X \succeq 0$$

where $X \succeq 0$ *means that* X *is a positive semidefinite matrix. More precisely* $X = \frac{1}{\sqrt{\lambda}}L^{+}$ *where* L^{+} *is the Moore-Penrose inverse of Laplacian matrix, and* $\lambda = \max_{(i,j)\in E} \frac{1}{z_{ij}} \left\|F_\alpha L^{+} u_{ij}\right\|^2$.

Proof. The Lagrangian of optimization problem is:

$$\begin{aligned} L(\Gamma, W, T, \lambda, \rho) &= \mathrm{Tr}(U_\alpha \Gamma^{-1}) + \mathrm{Tr}(T\Gamma) - C\lambda \\ &\quad + \sum_{(i,j)\in E} w_{ij}(-u_{ij}^t T u_{ij} + \lambda z_{ij} - \rho_{ij}) - \frac{1}{n}\overrightarrow{1}^t T \overrightarrow{1} \end{aligned}$$

To find the dual formulation, it is enough to take the infimum of the Lagrangian over Γ, W.

$$\begin{aligned} d(T,\lambda,\rho) &= \inf_{\Gamma,W} L(\Gamma, W, T, \lambda, \rho) \\ &= \inf_{\Gamma} \mathrm{Tr}(U_\alpha \Gamma^{-1} + T\Gamma) \\ &\quad + \inf_W \left(- \sum_{(i,j)\in E} w_{ij}(-u_{ij}^t T u_{ij} + \lambda z_{ij} - \rho_{ij}) - \frac{1}{n}\overrightarrow{1}^t T \overrightarrow{1} - C\lambda \right) \end{aligned} \tag{4.44}$$

The second term in (4.44) is minimized if its derivative with respect to all link weights is zero. The minimum of the first term is $-\infty$ unless matrix T is positive semidefinite. Therefore,

$$d(T,\lambda,\rho) = \begin{cases} \inf_\Gamma \mathrm{Tr}(U_\alpha \Gamma^{-1} + T\Gamma) - \frac{1}{n}\overrightarrow{1}^t T \overrightarrow{1} - C\lambda \\ \text{if} - u_{ij}^t T u_{ij} + \lambda z_{ij} - \rho_{ij} = 0, \;\; \rho_{ij} \geq 0 \;\; \forall (i,j) \in E \\ \text{and} \;\; T \succeq 0 \\ -\infty \quad \text{otherwise} \end{cases} \tag{4.45}$$

where $T = [t_{ij}]$, and $T \succeq 0$ means that matrix T is positive semidefinite.

Using Proposition 4.2 one can find $\inf_\Gamma (U_\alpha \mathrm{Tr}\Gamma^{-1} + T\Gamma)$ as follows:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\Gamma}\mathrm{Tr}(U_\alpha \Gamma^{-1} + T\Gamma) &= \frac{\mathrm{d}}{\mathrm{d}\Gamma}\mathrm{Tr}(U_\alpha \Gamma^{-1}) + \frac{\mathrm{d}}{\mathrm{d}\Gamma}\mathrm{Tr}(T\Gamma) = 0 \\ T &= \Gamma^{-1} U_\alpha \Gamma^{-1} \end{aligned} \tag{4.46}$$

Considering the fact that U_α and Γ^{-1} are symmetric matrices, after some calculations we have

$$\inf_\Gamma \mathrm{Tr}(U_\alpha \Gamma^{-1} + T\Gamma) = 2\mathrm{Tr}(U_\alpha \Gamma^{-1}) \tag{4.47}$$

We also note that $T\overrightarrow{1} = 0$ (because $\Gamma^{-1}\overrightarrow{1} = (L^{+} + \frac{J}{n})\overrightarrow{1} = \overrightarrow{1}$ and $U_\alpha \overrightarrow{1} = 0$). Now we consider a change of variable as $X = \frac{1}{\sqrt{\lambda}}L^{+}$, clearly $X\overrightarrow{1} = 0$. It is easier to write the optimization problem based on new variable X. Applying this change of variable, we have

$$2\mathrm{Tr}(U_\alpha \Gamma^{-1}) = 2\sqrt{\lambda}\,\mathrm{Tr}(U_\alpha X) \tag{4.48}$$

On the other hand,

$$\begin{aligned} T &= \Gamma^{-1}U_\alpha \Gamma^{-1} \\ &= \lambda X U_\alpha X \end{aligned} \tag{4.49}$$

Finally, we observe from constraint part of (4.45) that

$$\begin{aligned} \frac{1}{z_{ij}} u_{ij}^t T u_{ij} &\le \lambda \;\; \forall (i,j) \in E \\ \frac{1}{z_{ij}} u_{ij}^t \lambda X U_\alpha X u_{ij} &\le \lambda \;\; \forall (i,j) \in E \\ \frac{1}{\sqrt{z_{ij}}} \left\| F_\alpha X u_{ij} \right\| &\le 1 \end{aligned} \tag{4.50}$$

where F_α is an $m \times n$ matrix (m and n are the number of links and nodes of the graph respectively) such that $U_\alpha = F_\alpha^t F_\alpha$. This matrix decomposition always exists, since U_α is a positive semidefinite matrix.

Now it is enough to simplify the dual objective function of (4.45) (i.e. $d(X,\lambda) = 2\sqrt{\lambda}\,\mathrm{Tr}(U_\alpha X) - C\lambda$) using (4.48). In order to maximize the dual function with respect to the dual variable λ, one should have $\frac{\mathrm{d}}{\mathrm{d}\lambda} d(X,\lambda) = 0$. By applying this, and after some calculations, the dual objective will be equal to $\frac{1}{C}\mathrm{Tr}^2(U_\alpha X)$, which is now only a function of dual variable X.

Therefore, considering (4.50), one can write the dual optimization problem as

$$\begin{array}{ll} \text{maximize} & \frac{1}{C}\mathrm{Tr}^2(U_\alpha X) \\ \text{subject to} & \frac{1}{\sqrt{z_{ij}}} \left\| F_\alpha X u_{ij} \right\| \le 1 \;\; \forall (i,j) \in E \\ & X\overrightarrow{1} = 0 \\ & X \succeq 0 \end{array}$$

Finally, we observe that

$$\lambda = \max_{(i,j)\in E} \frac{1}{z_{ij}} u_{ij}^t T u_{ij}$$

$$= \max_{(i,j)\in E} \frac{1}{z_{ij}} u_{ij}^t L^+ U_\alpha L^+ u_{ij}$$

$$= \max_{(i,j)\in E} \frac{1}{z_{ij}} \left\| F_\alpha L^+ u_{ij} \right\|^2 \tag{4.51}$$

This completes the proof of Lemma 4.4.

We are ready to give an upper bound for the optimality gap in the optimization problem. The following theorem summarizes the result.

Theorem 4.3. *Consider the following optimization problem:*

$$\begin{aligned} &\text{Minimize} \quad \tau_\alpha \\ &\text{Subject to} \quad \textstyle\sum_{(i,j)\in E} z_{ij} w_{ij} = C \quad , C \text{ is fixed} \\ &\qquad w_{ij} \geq 0 \; \forall (i,j) \in E \end{aligned}$$

For any sub-optimal solution of the convex optimization problem, the deviation from optimal solution (optimality gap) has the upper bound of

$$\frac{\tau_\alpha}{C \min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}}} \left(C \min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}} + \tau_\alpha \right)$$

Proof. We denote the duality gap (the difference between the objective function of the dual and primal optimization problem) by d_{gap}. Using Lemma (4.4), we have

$$\begin{aligned} d_{\text{gap}} &= \operatorname{Tr}(U_\alpha L^+) - \frac{1}{C} \operatorname{Tr}^2(U_\alpha X) \\ &= \operatorname{Tr}(U_\alpha L^+) - \frac{1}{C} \frac{1}{\max_{(i,j)\in E} \frac{1}{z_{ij}} \left\| F_\alpha L^+ u_{ij} \right\|^2} \operatorname{Tr}^2(U_\alpha L^+) \\ &= \operatorname{Tr}(U_\alpha L^+) \left(1 + \frac{1}{C} \frac{\operatorname{Tr}(U_\alpha L^+)}{\min_{(i,j)\in E} -\frac{1}{z_{ij}} \left\| F_\alpha L^+ u_{ij} \right\|^2} \right) \end{aligned} \tag{4.52}$$

Now it is enough to simplify (4.52) using Lemma 4.1,

$$\begin{aligned} d_{\text{gap}} &= \tau_\alpha \left(1 + \frac{1}{C} \frac{\tau_\alpha}{\min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}}} \right) \\ &= \frac{\tau_\alpha}{C \min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}}} \left(C \min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}} + \tau_\alpha \right) \end{aligned} \tag{4.53}$$

According to the duality theorem, this completes the proof of Theorem 4.3.

Theorem 4.3 is in fact an extension for the results of [23] in which the total resistance distance is considered as the main metric of interest. Those results can be derived as special cases of Lemma 4.4 and Theorem 4.3.

4.6.1 Network Planning Using an Interior Point Algorithm

Optimization problem (4.37) can be solved using a wide range of methods developed for convex optimization problems. We use a modified version of interior-point method which is developed in [23] based on the duality gap obtained in Theorem 4.3. In this method we use logarithmic barrier for the non-negativity constraints $w_{ij} \geq 0 \; \forall (i,j) \in E$:

$$\Phi = -\sum_{(i,j)\in E} \log w_{ij}$$

In the interior-point method we minimize $t\tau_\alpha + \Phi$, using Newton's method (t is a parameter), subject to $\sum_{(i,j)\in E} z_{ij} w_{ij} = C$. The sub-optimality in this case would be at most $\frac{m}{t}$, where m is the number of links. On the other hand, Theorem 4.3 provides an upper bound for sub-optimality:

$$\hat{t} = \frac{\tau_\alpha}{C \min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}}} \left(C \min_{(i,j)\in E} \frac{1}{z_{ij}} \frac{\partial \tau_\alpha}{\partial w_{ij}} + \tau_\alpha \right)$$

We can use this bound to update parameter t in each step of our interior-point method, by taking $t = \frac{m}{\hat{t}}$. In each step, for a relative precision ε, the Newton's method finds change of $\Delta \overrightarrow{w}$ for the vector of all weights (denoted by $\overrightarrow{w}$) by solving

$$(t\nabla^2 \tau_\alpha + \nabla^2 \Phi)\Delta \overrightarrow{w} = -t\nabla \tau_\alpha + \nabla \Phi \tag{4.54}$$

The next task is to find the Newton step length s by backtracking line search [32], and then updating the weight vector by $\overrightarrow{w} = \overrightarrow{w} + s\Delta \overrightarrow{w}$. The algorithm exits when we have $\tau_\alpha - \tau_\alpha^{opt} \leq \hat{t} \leq \varepsilon \tau_\alpha$. We can choose ε small enough to have a desired precision.

We note that, in order to use this recursive method, we need to have the gradient vector $\nabla \tau_\alpha$ and Hessian matrix $\nabla^2 \tau_\alpha$. Lemma 4.1 provides the entries of the gradient vector and using the lemma, it is easy to see that the entries of Hessian matrix can be found from the following equation:

$$\frac{\partial^2 \tau_\alpha}{\partial w_{pq} \partial w_{ij}} = 2u_{ij}^t L^+ u_{pq} u_{pq}^t L^+ U_\alpha L^+ u_{ij} \; \forall (i,j),(p,q) \in E$$

In matrix form this can be shown as

$$\nabla^2 \tau_\alpha = (B^t L^+ B) \; o \; (B^t L^+ U_\alpha L^+ B) \tag{4.55}$$

Algorithm

```
(∗ Summary of the interior point algorithm ∗)
1.   for $(i,j) \in E$
2.        do $w_{ij} \leftarrow \frac{C}{mz_{ij}}$ (initialization)
3.   repeat
4.        $\hat{t} \leftarrow \frac{\tau_\alpha}{C\min_{(i,j)\in E}\frac{1}{z_{ij}}\frac{\partial \tau_\alpha}{\partial w_{ij}}}(C\min_{(i,j)\in E}\frac{1}{z_{ij}}\frac{\partial \tau_\alpha}{\partial w_{ij}} + \tau_\alpha)$
5.        $t \leftarrow \frac{m}{\hat{t}}$
6.        Calculate $\Delta \overrightarrow{w}$ from (4.54) and (4.55).
7.        Update the value of Newton step $s$ using backtracking line search.
8.        Update weight: $\overrightarrow{w} = \overrightarrow{w} + s\Delta\overrightarrow{w}$
9.   until $\hat{t} \leq \varepsilon\tau_\alpha$
10.  return $\overrightarrow{w}$
```

where B is the incidence matrix of the graph, and o denotes Hadamard (componentwise) product. Assuming that we know the value of ε, the interior point algorithm can be summarized in the following chart:

4.7 Applications

This section considers applications of WNC. First, we develop a semidefinite programming formulation for general weight planning to minimize τ_α. We then develop optimization problems to jointly optimize the resources (weights) and routes (link flows) in a communication network. We will also discuss a joint optimization of demands, flows, and resources in order to maximize a concave utility function of demands, while keeping network criticality below a certain threshold. Finally, we discuss robust optimization of network weights in order to protect the network against k link failures.

4.7.1 Network Planning Using Semidefinite Programming

Optimization problem (4.38) (and problem (4.37)) provides an approach for robust network design via optimal allocation of network link weights to minimize the WNC. Optimization problem (4.38) can be converted to a semidefinite program (SDP) as stated in the following.

$$\begin{aligned} &\text{Minimize} \quad \mathrm{Tr}(Y) \\ &\text{Subject to} \quad \textstyle\sum_{(i,j)\in E} w_{ij}z_{ij} \leq C, \quad C \text{ is fixed} \end{aligned} \tag{4.56}$$

$$w_{ij} \geq 0 \;\; \forall (i,j) \in E$$

$$\begin{pmatrix} L + \frac{J}{n} & U_\alpha^{\frac{1}{2}} \\ U_\alpha^{\frac{1}{2}} & Y \end{pmatrix} \succeq 0$$

where $\succeq$ means positive semidefinite.

Capacity allocation is an important case in which we define the weight of a link to represent its capacity. In this case, optimization of network criticality results in capacity planning. A quite general case of capacity planning problem is when a routing mechanism is already designed for the network and it is known that each link of the network is supposed to carry a known amount of traffic demands. Suppose we know that our routing scheme is the shortest path and traffic matrix $[\gamma_i(j)]$ is given for the network. Assume that we have found the values of flows for each link to meet the given traffic matrix via shortest-path routing and the result is flow λ_{ij} for link (i,j). Then by applying the change of variable $w_{ij} = c_{ij} + \lambda_{ij}$ to the optimization problem (4.56), we will have the following convex optimization problem for the optimal capacity allocation.

$$\begin{aligned} &\text{Minimize} \quad \mathrm{Tr}(Y) \\ \text{Subject to} \quad &\textstyle\sum_{(i,j)\in E} c_{ij} z_{ij} = C',\; C' \text{ is fixed} \\ &c_{ij} \geq 0 \;\; \forall (i,j) \in E \\ &\begin{pmatrix} L + \frac{J}{n} & U_\alpha^{\frac{1}{2}} \\ U_\alpha^{\frac{1}{2}} & Y \end{pmatrix} \succeq 0 \end{aligned} \tag{4.57}$$

where $C' = C - \sum_{(i,j)\in E} z_{ij}\lambda_{ij}$. This has the same form of optimization problem (4.56), and both are SDP representations of optimization problem (4.37) (with $w_{ij} \to c_{ij}$ and $C \to C'$); therefore, all the results developed for optimization problem (4.37) are applicable for the capacity assignment problem.

Solving this SDP problem is much faster and can be done with a variety of existing packages (e.g. see [29, 30]) to solve SDP problems.

4.7.2 *Joint Optimization of Routes and Resources*

Optimization problem (4.37) can be extended to a more general case where the weights of the network links (resources) and the link flows (routes) are unknown. In this section, we focus on capacity as the resource and assume that the link weight equals the capacity (or available capacity depending on the context) of the link. In order to account for flows, it is enough to add the equations for the conservation of flow at each node (and for each entry of the traffic matrix) to the constraints of the problem. For a specific node k and entry $\gamma_s(d)$ of the traffic matrix, the conservation of flow can be stated as:

$$\sum_{i \in A(k)} f_{ik}^{(sd)} - \sum_{j \in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d)$$

where $A(k)$ denotes the set of neighbor nodes of node k, $f_{ik}^{(sd)}$ denotes the flow of link (i,k) for traffic entry between source s and destination d, and $\delta(x)$ is Kronecker delta function. Furthermore, the flow of each link should not exceed the capacity of the link; therefore, we will need the following constraints for each link of the network:

$$\begin{aligned} f_{ij} &= \sum_{sd} f_{ij}^{(sd)} \;\; \forall (i,j) \in E \\ f_{ij} &\leq w_{ij} \;\; \forall (i,j) \in E \\ f_{ij} &\geq 0 \;\; \forall (i,j) \in E \end{aligned}$$

Now we can write the optimization for robust joint flow assignment and resource allocation as follows:

$$\begin{gathered} \text{Minimize} \quad \tau_\alpha \\ \text{Subject to} \quad \sum_{(i,j)\in E} w_{ij} z_{ij} \leq C, \quad C \text{ is fixed} \\ w_{ij} \geq 0 \;\; \forall (i,j) \in E \\ \sum_{i\in A(k)} f_{ik}^{(sd)} - \sum_{j\in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d) \;\; \forall k \in N, \;\; \forall \gamma_s(d) \\ f_{ij} = \sum_{sd} f_{ij}^{(sd)} \;\; \forall (i,j) \in E \\ f_{ij} \leq w_{ij} \;\; \forall (i,j) \in E \\ f_{ij} \geq 0 \;\; \forall (i,j) \in E \end{gathered} \tag{4.58}$$

One efficient method to solve optimization problem (4.58) is the *dual decomposition* [32, 33] which can separate the network flow problem from resource allocation. We form a dual problem for (4.58) by introducing Lagrangian multiplier matrix Λ for constraint set $f_{ij} \leq w_{ij} \;\; \forall (i,j) \in E$. The resulting partial Lagrangian is $L = \tau_\alpha - \mathrm{Tr}(\Lambda(F - W))$, where $F = [f_{ij}]$ and $W = [w_{ij}]$ are the matrices of link flows and link weights, respectively. The dual objective function is then

$$\begin{aligned} d(\Lambda) = \inf_W \left(\tau_\alpha + \mathrm{Tr}(\Lambda W)\right) & \Bigg\| \sum_{(i,j)\in E} w_{ij} z_{ij} = C, \;\; w_{ij} \geq 0 \;\; \forall (i,j) \in E \\ + \inf_F (-\mathrm{Tr}(\Lambda F)) & \left\| \begin{gathered} \sum_{i\in A(k)} f_{ik}^{(sd)} - \sum_{j\in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d) \\ \forall k \in N, \\ f_{ij} = \sum_{sd} f_{ij}^{(sd)} \;\; \forall (i,j) \in E \end{gathered} \right. \end{aligned}$$

where $a \| b$ means a subject to condition set b. Note that two infimum functions in the dual objective are working on separate variables (the first infimum on weights, and the second one on flows). The dual function can be seen as the sum of the following two functions:

$$d_W(\Lambda) = \inf_W (\tau_\alpha + \mathrm{Tr}(\Lambda W)) \left\| \begin{array}{ll} \sum_{(i,j)\in E} w_{ij} z_{ij} = C, & C \text{ is fixed} \\ w_{ij} \geq 0 & \forall (i,j) \in E \end{array} \right. \tag{4.59}$$

$$d_F(\Lambda) = -\sup_F \mathrm{Tr}(\Lambda F) \left\| \begin{array}{c} \sum_{i\in A(k)} f_{ik}^{(sd)} - \sum_{j\in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d) \\ \forall k \in N, \\ f_{ij} = \sum_{sd} f_{ij}^{(sd)} \;\; \forall (i,j) \in E \end{array} \right. \tag{4.60}$$

The dual problem associated with the primal optimization problem (4.58) is

$$\begin{gathered} \text{Maximize} \quad d(\Lambda) = d_W(\Lambda) + d_F(\Lambda) \\ \text{Subject to} \quad \Lambda \geq 0 \end{gathered} \tag{4.61}$$

where $\geq$ is a component-wise operator (i.e. $\Lambda = [\lambda_{ij}] \geq 0$ means $\lambda_{ij} \geq 0 \;\; \forall i,j$). Optimization problem (4.61) is convex because the dual function is always convex [32]. We assume that the Slater's condition [32] is satisfied in optimization problem (4.58) in order to guarantee that the strong duality holds, i.e. the solution of optimization problem (4.58) and its dual (4.61) are equal. Note that in general this is not true when the primal objective function is not strictly convex. By assuming Slater's condition we can solve optimization problem (4.61) instead of the primal one.

To find the solution of dual optimization problem (4.61) we study dual functions $d_W(\Lambda)$ and $d_F(\Lambda)$ separately, and then we add them together. Considering (4.59), $d_W(\Lambda)$ is the solution of the following optimization problem:

$$\begin{gathered} \text{Minimize} \quad \tau_\alpha + Tr(\Lambda W) \\ \text{Subject to} \quad \sum_{(i,j)\in E} w_{ij} z_{ij} \leq C, \quad C \text{ is fixed} \\ w_{ij} \geq 0 \;\; \forall (i,j) \in E \end{gathered} \tag{4.62}$$

Optimization problem (4.62) can be viewed as the network planning subproblem which will assign the optimal values of weights. We refer to this as the *resource (weight) allocation subproblem* associated with problems (4.58) and (4.61). Similarly, (4.60) implies that $d_F(\Lambda)$ can be found by solving optimization problem (4.63) as follows:

$$\begin{gathered} \text{Maximize} \quad \mathrm{Tr}(\Lambda F) \\ \text{Subject to } f_{ij} = \sum_{sd} f_{ij}^{(sd)} \;\; \forall (i,j) \in E \end{gathered} \tag{4.63}$$

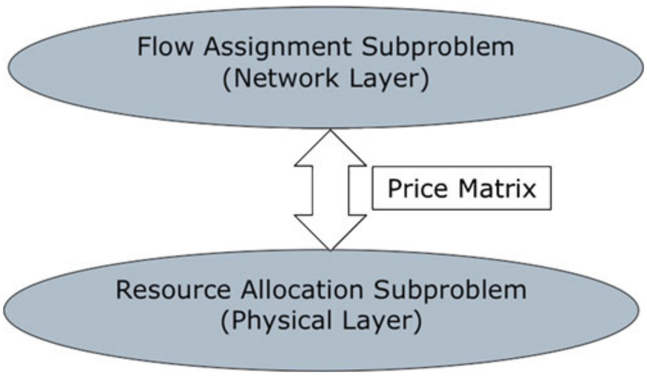


Fig. 4.1 Layered view of resource allocation and flow assignment subproblems

$$f_{ij} \geq 0 \ \ \forall (i,j) \in E$$

$$\Sigma_{i \in A(k)} f_{ik}^{(sd)} - \Sigma_{j \in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d) \ \ \forall k \in N, \ \ \forall \gamma_s(d)$$

Problem (4.63) determines the optimum routing given the optimal values of weights and we refer to it as *flow assignment subproblem* associated with problems (4.58) and (4.61).

Entry λ_{ij} of matrix Λ can be interpreted as the price of allocating unit weight to link (i, j) in the weight matrix. Therefore, the resource allocation subproblem (4.62) tries to minimize WNC incremented by total price of deploying a complete weight matrix, and the flow assignment subproblem (4.63) will try to maximally utilize the allocated weights to run the network flows.

A detailed discussion of numerical methods to solve optimization problem (4.61) is beyond the scope of our work in this chapter; however, we indicate how we can iteratively solve optimization problem (4.61) in the following manner. We start with an initial value for Λ, and we find the optimal weight and flow matrices using resource allocation and flow assignment subproblems. Then we find an update for price matrix Λ through subgradient method. Using the updated value of Λ we reoptimize the weight and flow matrices by solving optimization problems (4.62) and (4.63). This process continues until we arrive at a stable price matrix. This procedure allows us to view the resource allocation subproblem as an optimization process working on physical layer, while the flow assignment subproblem operates independently on a network layer and that provides optimal routing scheme for the problem. The connection between these two layers is through price matrix Λ as illustrated in Fig. 4.1.

Solution of optimization problem (4.58) is a symmetric set of link weights representing total capacity of links and associated symmetric link flows, but in general capacities and flows are asymmetric. We can modify the optimization problem to provide an asymmetric capacity and flow assignment (i.e. we change the undirected graph model to a directed one). We interpret link weights as the *available capacities* and reformulate the optimization problem accordingly. Let c_{ij} and w_{ij} denote the capacity and weight (available capacity) of link (i,j), respectively. Optimization problem (4.58) can be converted to the following problem:

$$\begin{gathered}
\text{Minimize} \quad \tau_\alpha \\
\text{Subject to} \quad \textstyle\sum_{(i,j)\in E} c_{ij}z_{ij} \le C, \quad C \text{ is fixed} \\
\textstyle\sum_{i\in A(k)} f_{ik}^{(sd)} - \sum_{j\in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d) \\
\forall k \in N, \ \forall \gamma_s(d) \\
f_{ij} = \textstyle\sum_{sd} f_{ij}^{(sd)} \ \ \forall (i,j) \in E \\
f_{ij} = c_{ij} - w_{ij} \ \ \forall (i,j) \in E \\
f_{ij} \ge 0 \ \ \forall (i,j) \in E \\
w_{ij} \ge 0 \ \ \forall (i,j) \in E
\end{gathered} \tag{4.64}$$

Note that optimization problem (4.64) provides optimal solutions of weights (available capacities), capacities, and flows for all the links. The weights will be symmetric; however, the link capacities (c_{ij}'s) need not be symmetric. This means that the optimization problem allocates capacities and flows in such a way that the final residual bandwidth or available capacities of link (i,j) and link (j,i) are equal (i.e. $w_{ij} = w_{ji}$), while the total capacity of link (i,j) is not necessarily equal to that of link (j,i) (i.e. $c_{ij} \neq c_{ji}$). The main difference between the solutions of optimization problems (4.58) and (4.64) is in the way they minimize τ_α. In the former, the link capacity is designed such that τ_α is minimized before applying any flow to the network, while the latter determines flows and capacities in such a way that τ_α for the residual network is minimized.

4.7.3 Joint Optimization of Demands, Flows, and Resources

We consider one more extension for the optimization problem which tries to find the joint optimal assignment of external demands (traffic matrix) and link flows when the link weights (link weights are set to the link capacities in this example) are known. In this problem, we optimize a concave utility function of traffic matrix $\Gamma = [\gamma_s(d)]$ (denoted by $\Psi(\Gamma)$) subject to the condition that the average network utilization $\bar{V}$ (see (4.32)) is less than a given maximum value ($\bar{V} \le a$ where a is fixed and assumed to be known). A well-known example of concave utility functions is the logarithmic function. In this work we are interested in a subset of all possible traffic matrices for which the constraint for network utilization $\bar{V}$ is convex; therefore, we add the set of inequalities (4.34) as constraints to the optimization problem. Clearly, the flow conservation and capacity constrains are also necessary. Summarizing all above, we can write the following optimization problem to find the optimal joint assignment of traffic demands and link flows.

$$\begin{aligned}
&\text{Maximize} \quad \Psi(\Gamma)\\
&\text{Subject to} \quad \textstyle\sum_{(i,j)\in E} w_{ij} z_{ij} \le C, \quad C \text{ is fixed}\\
&f_{ij} = \textstyle\sum_{sd} f_{ij}^{(sd)} \ \ \forall (i,j) \in E\\
&f_{ij} \le w_{ij} \ \ \forall (i,j) \in E\\
&f_{ij} \ge 0 \ \ \forall (i,j) \in E\\
&w_{ij} \ge 0 \ \ \forall (i,j) \in E\\
&\textstyle\sum_{i\in A(k)} f_{ik}^{(sd)} - \sum_{j\in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d) \ \ \forall k \in N, \ \ \forall \gamma_{sd}\\
&\textstyle\sum_{sd} \left(\frac{\gamma_s(d)+\gamma_d(s)}{4} + \frac{\gamma_{*s}-\gamma_{s*}}{2n} \right) \tau_{sd} \le a, \quad a \text{ is fixed}\\
&\gamma_s(d) + \gamma_d(s) \ge \tfrac{1}{n}(\gamma_{s*} - \gamma_{*s} + \gamma_{d*} - \gamma_{*d})\\
&\gamma_i(j) \ge 0 \ \ \forall i,j \in N
\end{aligned} \tag{4.65}$$

where a is a known maximum acceptable value for average network utilization.

Finally, we can jointly optimize resources, traffic demands, and routes. In this example, we let the link weight be the available capacity of the link. We assume that the goal is to maximize a concave function of traffic demands subject to a known worst case upper bound for network criticality. Moreover, we assume a maximum budget for the total weights (given a cost for deploying a unit of weight). Thus, considering the flow conservation and capacity constraints we will have the following convex optimization problem for joint optimization of weights, acceptable demands, and routes (flows):

$$\begin{aligned}
&\text{Maximize} \quad \Psi(\Gamma)\\
&\text{Subject to} \quad \textstyle\sum_{ij} z_{ij} c_{ij} \le C\\
&\tau \le b, \ \ b \text{ is fixed}\\
&\textstyle\sum_{i\in A(k)} f_{ik}^{(sd)} - \sum_{j\in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d) \ \ \forall k \in N, \ \ \forall \gamma_s(d)\\
&f_{ij} = \textstyle\sum_{sd} f_{ij}^{(sd)} \ \ \forall (i,j) \in E\\
&f_{ij} = c_{ij} - w_{ij} \ \ \forall (i,j) \in E\\
&f_{ij} \ge 0 \ \ \forall (i,j) \in E\\
&w_{ij} \ge 0 \ \ \forall (i,j) \in E\\
&\gamma_i(j) \ge 0 \ \ \forall i,j \in N
\end{aligned} \tag{4.66}$$

The solution of optimization problem (4.66) plans network weights, gives the optimal set of demands $\Gamma = [\gamma_i(j)]$ (maximizing concave utility function Ψ) which can be accommodated by the network, and provides link flows subject to the condition that the network criticality does not exceed a known value b.

It is possible to find a layered approach for optimization problem (4.66). The steps are similar to what we did for joint optimization of flows and resources (see (4.61)–(4.63)). Here, we construct a dual problem for (4.66) by introducing Lagrangian multiplier matrix Λ for constraint set $f_{ij} = c_{ij} - w_{ij} \quad \forall (i,j) \in E$. The resulting partial Lagrangian is $L = \Psi(\Gamma) - \mathrm{Tr}(\Lambda(F - C + W))$ where $F = [f_{ij}]$, $C = [c_{ij}]$ and $W = [w_{ij}]$ denote the matrices of link flows, link capacities, and link weights, respectively. The dual objective function is then $d(\Lambda) = d_{F,\Gamma}(\Lambda) + d_W(\Lambda) + d_C(\Lambda)$, where

$$d_W(\Lambda) = \sup_W \; -\mathrm{Tr}(\Lambda W) \left\| \begin{array}{l} w_{ij} \geq 0 \;\forall (i,j) \in E \\ \tau \leq b \end{array} \right.$$

$$d_C(\Lambda) = \sup_C \; \mathrm{Tr}(\Lambda C) \left\| \begin{array}{ll} \sum_{(i,j)\in E} c_{ij} z_{ij} = C, & C \text{ is fixed} \\ c_{ij} \geq 0 & \forall (i,j) \in E \end{array} \right.$$

$$d_{F,\Gamma}(\Lambda) = \sup_{F,\Gamma} \; (\Psi(\Gamma) - \mathrm{Tr}(\Lambda F)) \left\| \begin{array}{l} \sum_{i\in A(k)} f_{ik}^{(sd)} - \sum_{j\in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) \\ \qquad -\gamma_s(d)\delta(k-d) \;\; \forall k \in N, \\ f_{ij} = \sum_{sd} f_{ij}^{(sd)} \;\; \forall (i,j) \in E \\ f_{ij} \geq 0 \;\; \forall (i,j) \in E \\ \gamma_i(j) \geq 0 \;\; \forall i,j \in N \end{array} \right.$$

The dual problem associated with the primal optimization problem (4.66) is

$$\begin{aligned} \text{Minimize} \quad & d(\Lambda) = d_W(\Lambda) + d_C(\Lambda) + d_{F,\Gamma}(\Lambda) \\ \text{Subject to} \quad & \Lambda \geq 0 \end{aligned} \tag{4.67}$$

Since, in general, the primal objective function is not strictly concave, we assume that Slater's condition is satisfied in the optimization problem (4.66). This guarantees that the strong duality holds for this problem and that the solution of dual optimization problem (4.67) is equal to the solution of primal problem (4.66).

As we discussed before, in order to solve dual problem (4.67) we can evaluate $d_W(\Lambda)$, $d_C(\Lambda)$, and $d_{F,\Gamma}(\Lambda)$. We note that $d_{F,\Gamma}(\Lambda)$ is the solution of the following optimization problem:

$$\begin{gathered} \textit{Maximize} \quad \Psi(\Gamma) - \mathrm{Tr}(\Lambda F) \\ \textit{Subject to} \quad f_{ij} \geq 0 \;\; \forall (i,j) \in E \\ \gamma_i(j) \geq 0 \;\; \forall i,j \in N \\ \sum_{i\in A(k)} f_{ik}^{(sd)} - \sum_{j\in A(k)} f_{kj}^{(sd)} = \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d) \;\; \forall k \in N, \;\; \forall \gamma_s(d) \end{gathered} \tag{4.68}$$

Optimization problem (4.68) can be further divided into two optimization problems separating the effect of demands and flows by introducing another set of Lagrange multipliers. Similarly, $d_W(\Lambda)$ is the solution of optimization problem, (4.69).

$$\begin{aligned} &\text{Minimize} \quad Tr(\Lambda W) \\ &\text{Subject to} \quad \tau \leq b, \quad b \text{ is fixed} \\ &w_{ij} \geq 0 \ \ \forall (i,j) \in E \end{aligned} \tag{4.69}$$

The solution of optimization problem (4.70) provides us with $d_c(\Lambda)$,

$$\begin{aligned} &\text{Maximize} \quad \text{Tr}(\Lambda C) \\ &\text{Subject to} \quad \textstyle\sum_{(i,j)\in E} c_{ij} z_{ij} = C, \qquad C \text{ is fixed} \\ &c_{ij} \geq 0 \ \ \forall (i,j) \in E \end{aligned} \tag{4.70}$$

If we interpret Λ as the price matrix, then optimization problem (4.68) tries to maximize a utility function discounted by total price of assigning link flows. Optimization problem (4.69) finds the minimum required weights (available capacities) in order to guarantee that the network criticality of the residual network is not more than a pre-specified value b. Optimization problem (4.70) finds the best capacity assignment under price model mandated by matrix Λ.

In order to find the optimum solution of dual optimization problem (4.67), we start from an initial guess for matrix Λ. Solution of optimization problems (4.69) and (4.70) provide optimum values of weights and capacities at this stage, then the difference between capacity and weight of each link is equal to the flow of the link. Using optimization problem (4.68) we can then find best possible demand set. Then a new approximation for Lagrangian matrix Λ can be obtained using subgradient method and the iteration continues until we arrive at the stable matrix Λ.

4.7.4 Robust Network Design: Protecting Against Multiple Link Failures

The solution of optimization problem (4.37) provides a robust network design method via optimal weight assignment; however, it does not necessarily protect the network against multiple link failures. Link (node) failures can be the result of unplanned random failures or due to targeted attacks. In this section we extend optimization problem (4.37) to account for multiple link failures,

Let $D \in \{0,1\}^m$ be a binary matrix representing the location of link failures (m is the total number of links in the network), i.e. $d_{ij} = 0$ for failed link (i,j) and $d_{ij} = 1$ for operational ones. We replace the weight matrix W with DoW (o is the Hadamard operator) and redo the optimization of WNC. Now if we want that the network to be robust to up to k link failures (we refer to such a network as *k-robust* network), we need to minimize the following objective function:

$$\max_{\sum_{i,j} d_{ij}=m-k} \mathrm{Tr}(U_\alpha L^+(DoW))$$

Note that the above function is convex because it is a point-wise maximum of a set of convex functions. By minimizing this function we find a k-robust topology (along with its optimal link weights). Therefore, a general optimization problem to provide a k-robust network can be written as

$$\begin{aligned} \text{Minimize} \quad & \max\nolimits_{\sum_{i,j} d_{ij}=m-k} \mathrm{Tr}(U_\alpha L^+(DoW)) \\ \text{Subject to} \quad & \textstyle\sum_{(i,j)\in E} w_{ij}z_{ij} \le C, \quad C \text{ is fixed} \\ & w_{ij} \ge 0 \ \ \forall (i,j) \in E \end{aligned} \tag{4.71}$$

Interpreting weight as capacity, (4.71) can be extended to provide simultaneous solution for k-robust flow assignment and weight allocation, just by adding the flow conservation equations and link capacity constraints

$$\begin{aligned} \text{Minimize} \quad & \max\nolimits_{\sum_{i,j} d_{ij}=m-k} \mathrm{Tr}(U_\alpha L^+(DoW)) \\ \text{Subject to} \quad & \textstyle\sum_{(i,j)\in E} c_{ij}z_{ij} \le C, \quad C \text{ is fixed} \qquad (4.72) \\ \textstyle\sum_{i\in A(k)} f_{ik}^{(sd)} - \sum_{j\in A(k)} f_{kj}^{(sd)} &= \gamma_s(d)\delta(k-s) - \gamma_s(d)\delta(k-d) \\ & \forall k \in N, \ \ \forall \gamma_s(d) \\ f_{ij} &= \textstyle\sum_{sd} f_{ij}^{(sd)} \ \ \forall (i,j) \in E \\ f_{ij} &= c_{ij} - w_{ij} \ \ \forall (i,j) \in E \\ f_{ij} &\ge 0 \ \ \forall (i,j) \in E \\ w_{ij} &\ge 0 \ \ \forall (i,j) \in E \end{aligned} \tag{4.73}$$

4.7.5 Case of Directed Networks

Most of the discussion in this section was based on the assumption that our network is modeled with an undirected graph. We considered the case of having asymmetric capacities and link flows by interpreting the link weight as available capacity; however, even in this case the residual network has symmetric link weights. The main reason for this assumption is that the concept of resistance distance is only available on reversible Markov chains. However, there is another nice interpretation for network criticality which provides guidlines to extend the notion of network criticality to directed graphs.

Suppose that there are costs associated with traversing links along a path and consider the effect of network criticality on average cost incurred by a message during its walk from source s to destination d. It is shown in [20] that the average incurred cost is the product of network criticality and the total cost of all the link weights. Therefore, if we set a fixed maximum budget for the cost of assigning weights to links, then the average travel cost is minimized when network criticality is minimized.

While the analogy between resistance distance and random-walks does not hold in directed graphs, we can still find the hitting times and commute times for a directed graph, and the interpretation of average travel cost (or equivalently average commute time) still holds. In fact, we have shown that the average travel time in a directed graph can be found using the exact same analytical machinery [34], i.e. the trace of generalized inverse of the combinatorial Laplacian matrix of a directed graph (L) which is defined as $L = \Phi(I - P)$, where Φ is a diagonal matrix with main diagonal entry i equal to the ith entry of stationary probability vector corresponding to transition probability matrix P. We propose to use the average travel time as the objective of our optimization problem in the case of directed graphs. In this case the optimization problem is not necessarily convex (with regards to the link weights).

4.8 Case Study

In this section, we consider some simple applications of the optimization problems discussed in Sect. 4.7. In particular, we are interested in virtual network design to assign robust resources and flows to different customers of the network. We will also apply our optimization problems to find the optimal joint resource/flow assignment for a real (Abilene) with existing traffic matrix traces. Furthermore, we will apply the proposed optimization problems in robust capacity allocation for communication networks (Rocketfuel topologies) and power grids.

4.8.1 Virtual Network Assignment

In our first case study, we investigate the problem of assigning virtual networks to different customers of a communication network in order to meet their contracted service levels. In this study, we are interested in end-to-end bandwidth requirements as the main service.

For illustrative purposes we consider a simple topology which is shown in Fig. 4.2. This network is referred to as trap topology in networking literature. The trap topology is well-known in the context of survivable routing. Suppose there is a demand from node 1 for node 6. The min-hop path from node 1 to 6 is the straight

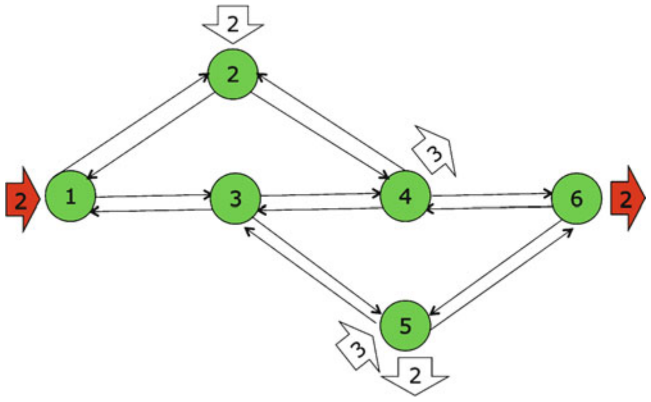


Fig. 4.2 Trap network with given traffic matrix

line $1 \rightarrow 3 \rightarrow 4 \rightarrow 6$. It appears that this path is the best choice to run the demand, but in survivable routing we need to assign backup paths to each primary route. In trap network, there is no link-disjoint backup path for $1 \rightarrow 3 \rightarrow 4 \rightarrow 6$. Therefore it would be desirable to choose path $1 \rightarrow 2 \rightarrow 4 \rightarrow 6$ ($1 \rightarrow 3 \rightarrow 5 \rightarrow 6$) as the primary route for demands from 1 to 6. Then the link-disjoint backup path will be $1 \rightarrow 3 \rightarrow 5 \rightarrow 6$ ($1 \rightarrow 2 \rightarrow 4 \rightarrow 6$).

We assume each customer in the network is defined by a set of demands identified by their source, destination and required bandwidth, i.e. each demand σ_i is defined by a triple (s_i, d_i, b_i), where s_i, d_i, b_i denote source, destination and required bandwidth of demand σ_i. In this study, we assume two customers (CS1 and CS2) exist on trap network with the following demands:

$$\begin{aligned}
\sigma_1 &= (1,6,2) \\
\sigma_2 &= (2,5,2) \\
\sigma_3 &= (5,4,3) \\
CS1 &= \{\sigma_1, \sigma_2\} \\
CS2 &= \{\sigma_3\}
\end{aligned}$$

From the above description, the requirements of customers can be summarized in separate virtual networks assigned to each customer as shown in Fig. 4.3.

Our goal is to determine the optimal robust allocation of capacities and flows for each customer so as to meet the requirements of all the customers. We will use optimization problem (4.64) to find optimal capacity allocation and flow assignment simultaneously. Optimization problem (4.64) permits us to find asymmetric capacity assignment for the links; however, due to the nature of network criticality, the residual capacity of links after flow assignment is symmetric.

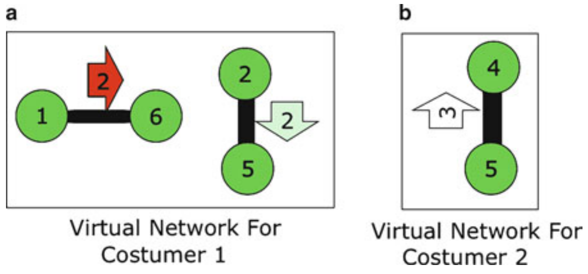

a
b
1
2
6
2
2
5
4
3
5
Virtual Network For Costumer 1
Virtual Network For Costumer 2

Fig. 4.3 Desired virtual networks for CS1 and CS2

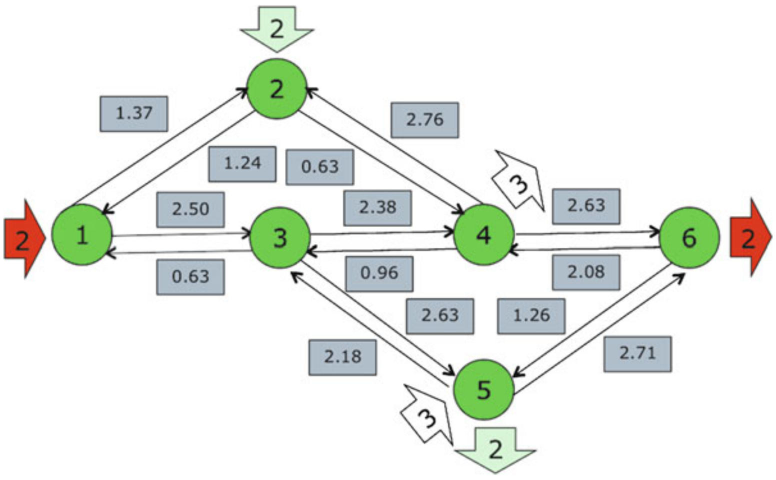


Fig. 4.4 Optimal capacity assignment for trap network

We suppose that the cost of deploying all the links (z_{ij}'s) are equal (and assumed to be 1) and the total budget for capacity is 26 (it means that $\sum_{ij} z_{ij}c_{ij} = \sum_{ij} c_{ij} = 26$). Solution of optimization problem (4.64) will result in the capacity allocation of Fig. 4.4.

Now we need to find the exact graph embedding for virtual networks of CS1 and CS2. For CS1, based on the solution of optimization problem (4.64), the optimal flow assignment for the physical substrate (trap network) is shown in Fig. 4.5. From Fig. 4.5 we see that all the nodes of the substrate network involve in providing connection (service) for CS1, however, four links ((3,1), (4,2), (5,3), (6,4)) do not contribute in building the virtual network. As a matter of fact the virtual network can be viewed as a subset of link/node resources dedicated to a customer.

The optimal flow assignment for CS2 is shown in Fig. 4.6. It can be seen that for CS2 nodes 1 and 2 do not contribute in providing service and among all the links only four links involve in guaranteeing service for CS2.

Adding the flows of two customers, we can have the total flow of each link in trap network as depicted in Fig. 4.7.

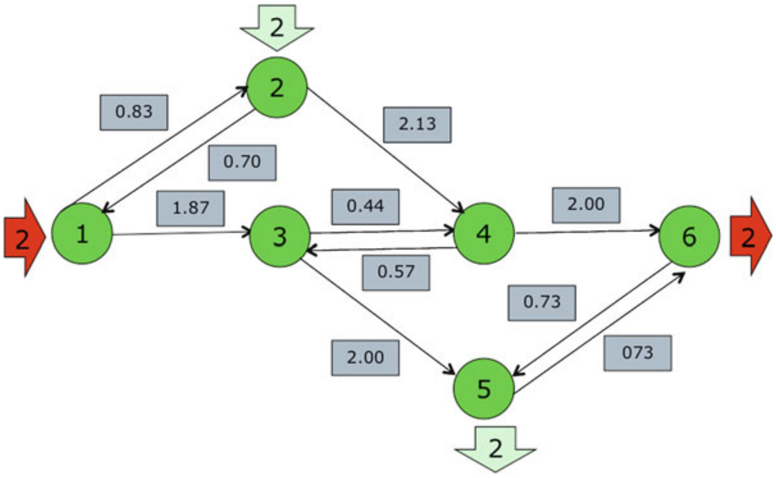
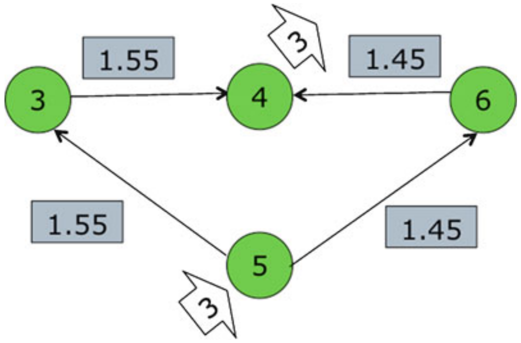


Fig. 4.5 Optimal resource assignment for virtual Network 1 (Customer 1) on trap physical topology

Fig. 4.6 Optimal resource assignment for virtual Network 2 (Customer 2) on trap physical topology



4.8.2 *Optimal Joint Capacity Allocation/Routing for Abilene*

As the next case study, we consider Abilene network [26]. Real traffic matrix traces for Abilene are publicly available. For example, these traffic matrices can be obtained from the website of TOTEM project [27], which is an open source platform including a number of well-known traffic engineering methods. We used one of the available traffic matrices for Abilene, and solved joint optimization problem (4.64) to find optimal link capacities and link flows simultaneously. We measured the utilization of all the link of the Abilene and compared it with the link utilization of two other traffic engineering methods, i.e. SPF (shortest path first) and OW, where OW is a weight optimization tool for Intra-domain internet routing protocols such as OSPF and IS-IS. It determines the weight of the links in order to utilize the network more efficiently by using tabu search meta-heuristic method [36]. In SPF and OW

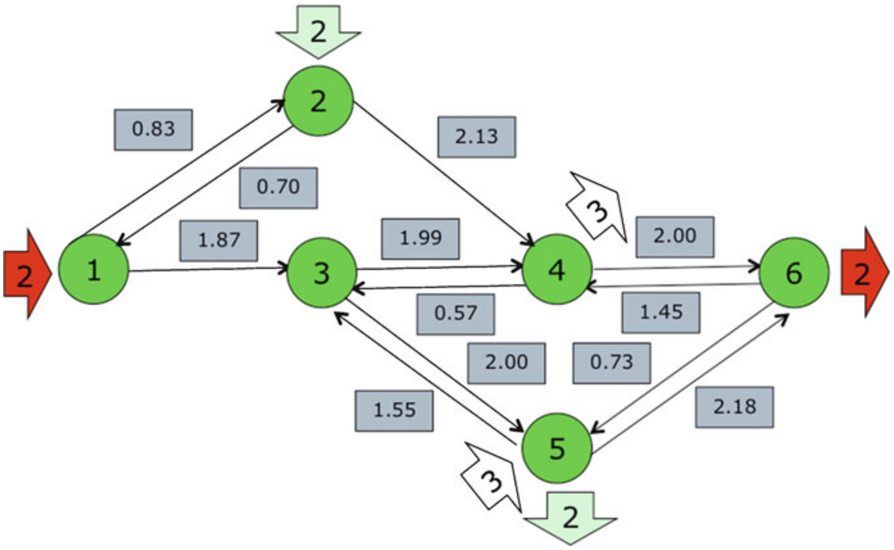


Fig. 4.7 Total link flow assignment on trap network

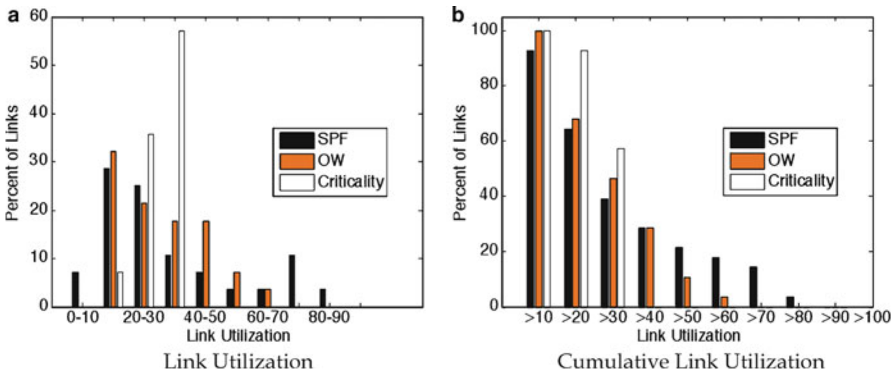


Fig. 4.8 Link utilization for three different traffic engineering methods: SPF, OW, and network criticality minimization

methods, we used the existing capacity allocation for Abilene network and obtained the flow assignment per link. We used TOTEM package [27] to solve SPF and OW problems.

Figure 4.8(a) shows the distribution of link utilization in each one of the three methods, i.e. it shows what percentage of links have a specific utilization. Figure 4.8(b) is the cumulative link utilization representation, which says what percentage of links have utilization higher than a specific value. Figure 4.8(b) clearly shows that in our method (criticality), there is no link with utilization

Table 4.1 RocketFuel dataset ISPs

ISP	Routers	Links	Reduced cities	Reduced links	Weight per link	Total weight
1,755	87	322	18	33	0.7822	51.628
3,967	79	294	21	36	0.6743	48.551
1,239	315	1,944	30	69	0.7231	99.784

more than 50%, while in the other two methods (SPF, and OW) we have links with high utilization, which makes the network vulnerable to the future demands (if any).

4.8.3 Robust Capacity Assignment

We study the application of optimization problem (4.71) in robust capacity allocation for RocketFuel topologies [35] as the most trustable existing dataset for Internet service provider (ISP) networks. In this experiment, we assumed that $\alpha_{ij} = \frac{1}{n(n-1)}\ \forall(i,j) \in E$, which means that the objective function of optimization problem (4.71) in this experiment equals average network criticality, i.e. $\hat{\tau}$.

In order to use the dataset, we followed the method described in [9] and collapsed the RocketFuel ISP topologies into PoP to PoP connectivity networks. In other words, we consolidated all the nodes within a city into a single node, and aggregated all the links from one city to another one in a single link, where the capacity of the link equals the sum of the capacities of all the original links connecting different sub-nodes between two cities. There are six ISP topologies in RocketFuel dataset, whose topological information are given in [9]. The topologies in RocketFuel dataset do not include the capacities of the links, but we can use OSPF weight information which is provided in RocketFuel dataset to associate compatible capacities using Cisco recommendation as described in [9]. Cisco recommends that the link capacities are proportional to the reciprocal of the weights.

We do our experiments on three network topologies from RocketFuel dataset. Table 4.1 shows the specification of the RocketFuel networks we used in our experiments. We would like to study possible gains we may achieve by replacing present capacity allocation for RocketFuel topologies with the optimal weight (capacity) set obtained as the solution of the proposed k-robust method. In the following experiments, we consider four different weight sets. First of all, since the networks are real, we already have an initial weight (IW) set. In IW, the weights are proportional to the capacity of the real network. In the second weight set, the total weight (total capacity) of the network is uniformly distributed among all the link weights; therefore, we have equal weight allocation (EW). The third weight set which is denoted by OT is the solution of optimization problem (4.56). Again for this optimization problem we assume $\alpha_{ij} = \frac{1}{n(n-1)}\ \forall(i,j) \in E$. The solution of OT is robust in the sense that the average network criticality is minimized; however, it is

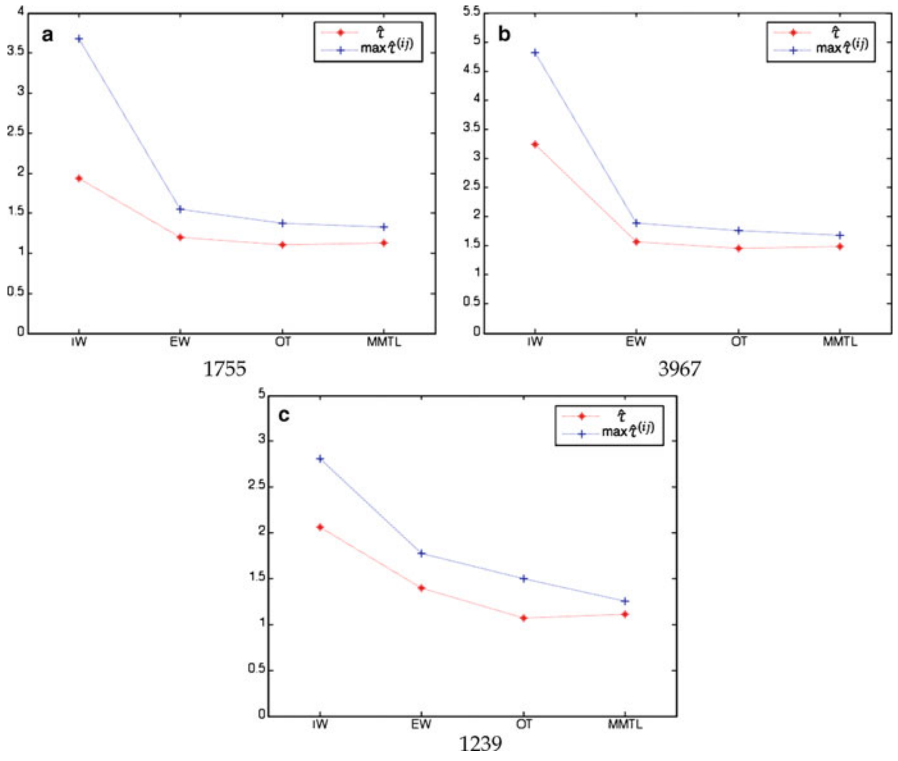
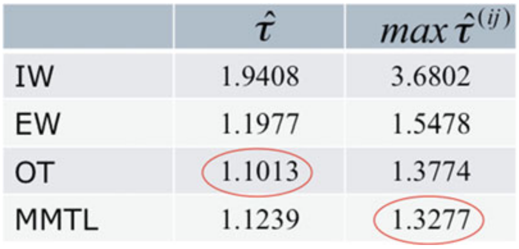


Fig. 4.9 Comparison of IW and EW with optimized weight sets for RocketFuel topologies 1,755, 3,967, and 1,239

not optimized for vulnerability (i.e. failures). Finally, the fourth weight set is the result of optimizing the network for link failure. We consider the case of 1-robust topology weight design using optimization problem (4.71), and the optimal weight set is denoted by MMTL. In these experiments all the link costs (z_{ij}'s) are assumed to be 1.

Figure 4.9(a)–(c) show the value of the average network criticality (objective function of optimization problem (4.56)), and the value of $\max_{\sum_{i,j} d_{ij}=n-k} \mathrm{Tr}(U_\alpha L^+ (DoW))$ (objective function of optimization problem (4.71)) for RocketFuel. In the figures, these values are denoted by $\hat{\tau}$ and $\max \hat{\tau}^{(ij)}$ (the superscript (ij) shows that $\hat{\tau}^{(ij)}$ is the value of network criticality when link (i,j) is removed), respectively. It can be easily seen that there is a huge gap between IW and optimized weight assignment for RocketFuel networks. This verifies that our optimizations can significantly improve the vulnerability of the network. For example, according to Fig. 4.10, the optimal vulnerability parameters of network 1,755 show 42% and 61.4% improvement, respectively, comparing with initial weight (IW) case.



	$\hat{\tau}$	$max\,\hat{\tau}^{(ij)}$
IW	1.9408	3.6802
EW	1.1977	1.5478
OT	1.1013	1.3774
MMTL	1.1239	1.3277

Fig. 4.10 Parameters of different optimized weight sets for 1,755 Network

4.8.4 Design of Robust Power Grids

The concept of TANC has a nice application in smart power grids. Nowadays the idea of using renewable energy sources has gained considerable attention. Many places with renewable energy (such as places with high wind) are not within the reach of existing power grid network and it is required to extend the existing power grid to the places with renewable energy. Thus, we need to know how to design a robust power network as an extension to the existing one. In addition to the robustness, a power grid should be sparse enough, to avoid unnecessary power lines. This problem is recently investigated and a method of sparsification is also developed [37]. Here, we follow the method of [37] to formulate the optimization problem for our power network. We will see that the optimization problem has exactly the form of our *k*-robust problem. We will solve the problem and then and we use an sparsification method based on the concept of resistance distance to prune the network.

It can be shown that in a DC-model approximation of a power grid, the average power dissipation of the grid is proportional to $\mathrm{Tr}(AL^{+})$, where $A = < \overrightarrow{d}\,\overrightarrow{d}^{t} >$ and $\overrightarrow{d}$ is the vector of link electrical currents ($< . >$ denotes time average) [37]. Clearly, the power dissipation has the general form of WNC (or equivalently TANC if we interpret the power as network flow, and weights as line conductances); therefore, minimization of power dissipation in power grids results in minimization of WNC. We address the optimization of a power grid network with multiple random independent loads supplied by a generator. For consumer nodes, we specify mean load $\bar{a}_i < 0$ and the variance σ_i^2. At transmission (relay) nodes, the average and variance of the load are zero. At the generator we must have $a_0 = -\sum_{i\neq 0} a_i$. Therefore, matrix $A = < \overrightarrow{d}\,\overrightarrow{d}^{t} >$ can be written as

$$\begin{pmatrix} (\sum_{i\neq 0}\bar{a}_i)^2 + \sum_{i\neq 0}\sigma_i^2 & -\overrightarrow{1}^{t}(\overrightarrow{d}\,\overrightarrow{d}^{t} + \Sigma) \\ -(\overrightarrow{d}\,\overrightarrow{d}^{t} + \Sigma)\,\overrightarrow{1} & \overrightarrow{d}\,\overrightarrow{d}^{t} + \Sigma \end{pmatrix}$$

We let $\bar{a}_i = -1$ and $\sigma_i^2 = \frac{1}{4}$ for consumer nodes in our tests in this section. We consider an $n - by - n$ grid (n is an odd number) and we let the generator node be the middle node of the grid and consumer nodes on the border nodes (Fig. 4.11(a)), or a middle node in one of the border lines of the grid and consumers on the parallel border (Fig. 4.11(b)).

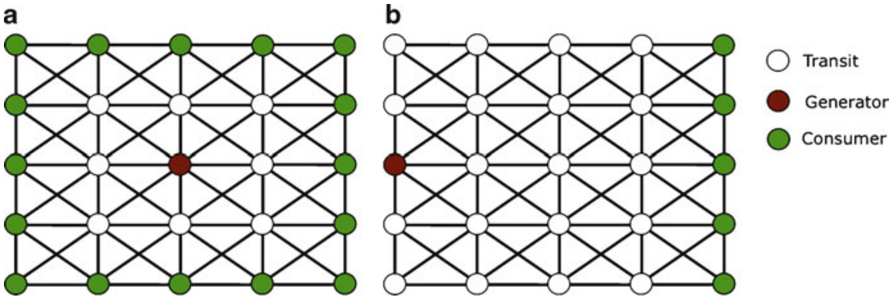


Fig. 4.11 Power grids with one generator node

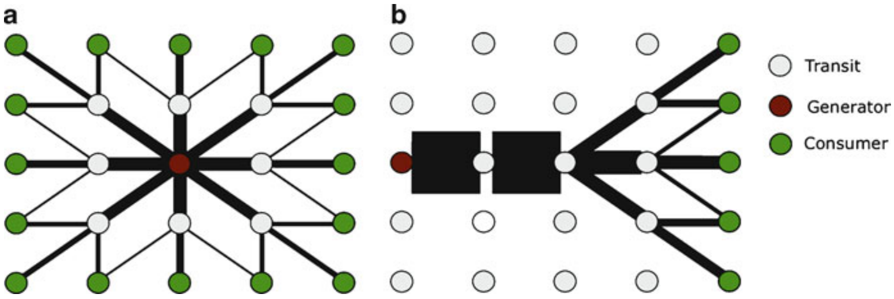


Fig. 4.12 Optimal grid topologies – thickness of the lines represent the conductances

First, we optimize the grid for power dissipation (i.e. we minimize WNC $\mathrm{Tr}(AL^+)$. The optimization problem is essentially the same as problem (4.56) with appropriate values of α_{ij}, so that $U_\alpha = A$. We solved problem (4.56) for the given values of $U_\alpha = A$. The optimal networks are shown in Fig. 4.12(a), (b), where the thickness of the lines represent the link weights or line conductances (thicker line has higher conductance). We discuss Fig. 4.12(b), since it is more vulnerable and needs attention. Figure 4.12(b) shows that by optimizing WNC, we prune the original grid; however, this network does not provide protection against possible link/node failures. We can use optimization problem (4.71) to find a k-robust grid power topology. Figure 4.13a shows an example of a 1-robust topology.

We provide one more extension that is particularly useful for the case of power grids in which the network should be sparse enough while preserving robustness. We would like to sparsify the robust topology of Fig. 4.13a. Fortunately, there is an elegant study on the context of sparsification using resistance distance. In [38], the problem of finding an sparse version of a large network is addressed with the goal of keeping the total resistance distance of the original graph and its sparse version as close as possible. The authors have proposed an algorithm to find such sparse networks. The algorithm works as follows. Suppose H is the sparse version of graph G. Choose a random line (i, j) of the network G with probability p_{ij} proportional to $w_{ij}\tau_{ij}$, where τ_{ij} is the point-to-point network criticality or the

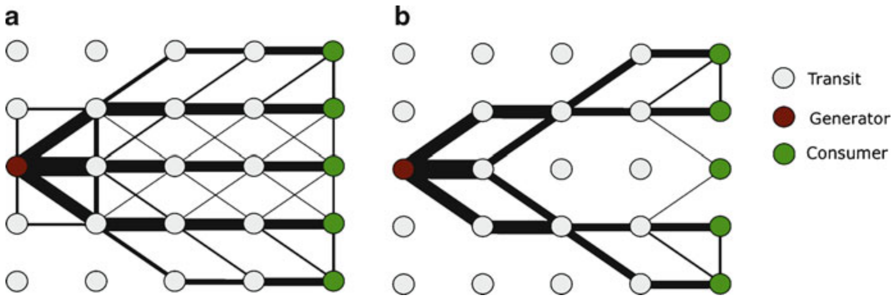


Fig. 4.13 (**a**) Optimal robust grid topology (link failure), (**b**) optimal robust sparse grid topology

resistance distance seen between nodes i and j. Add (i, j) to H with weight $\frac{w_{ij}}{qp_{ij}}$, where q is the number of independent samples (we should sum up weights if a line is chosen more than once). We used this algorithm to simplify the optimal robust network of Fig. 4.13a (note that we have chosen a large grid so that the conditions for applying the sparsifying algorithm are met), and the result is shown in Fig. 4.13b.

The network criticality of the sparse topology in Fig. 4.13b is close to that of the original topology (Fig. 4.13a) and it is still 1-robust, but the number of active links (power lines) in the topology of Fig. 4.13b is much less than the original one.

4.9 Conclusion

In this chapter, we developed a number of optimization problems for simultaneous optimization of resources and flows for communication networks and power grids. Our goal in the optimization problems was to minimize a weighted linear combination of resistance distances (point-to-point network criticalities) which is a convex function of link weights.

In another development, we discussed the problem of finding the best matched traffic matrix to a given network topology, along with optimal routing strategy (flow assignment) associated with optimal demand. We extended this idea and proposed an optimization problem to jointly optimize demands, routes and resources. Moreover, we discussed the application of network criticality in planning k-robust networks, where the topology is potentially protected against up to k link failures.

We used the proposed optimization problems to design virtual networks for different customers of a communication network. We also applied the k-robust strategy to design robust communication networks, and robust sparse power networks.

There are different avenues for further development of the proposed ideas in this chapter. We can extend the optimization problems to the case where other QoS constraints, such as delay partitioning constraints, are also requested. Our discussion in this work was mostly on undirected networks (also we explained how we can have asymmetric capacities within the proposed framework). One more extension is to develop similar optimization problems for the general case of a directed graph.

References

1. A. H. Dekker, B. D. Colbert. Network Robustness and Graph Topology. *Australasian Computer Science Conference*, vol. 26, page 359–368, January 2004.
2. A. H. Dekker, B. D. Colbert. The Symmetry Ratio of a Network. *Australasian symposium on Theory of computing, ACM International Conference Proceeding Series*, vol. 41, page 13–20, Newcastle, Australia, 2005.
3. P. Holme, B. Kimand, C. Yoon, S. K. Han. Attack vulnerability of complex networks. *Physical Review*, E65, 056109, 2002.
4. R. Criado, J. Flores, B. Hernandez-Bermejo, J. Pello, M. Romance. Effective Measurement of Network Vulnerability Under Random and Intentional Attacks. *Journal of Mathematical Modelling and Algorithms*, vol. 4, pp. 307–316, 2005.
5. R. Zhang-Shen, N. McKeown. Designing a Predictable Internet Backbone with Valiant Load-Balancing. *Thirteenth International Workshop on Quality of Service (IWQoS)*, Passau, Germany, June 2005.
6. L. Valiant, G. Brebner. Universal Schemes for Parallel Communication. *Thirteenth Annual Symposium on Theory of Computing*, May 1981.
7. C. S. Chang, D. S. Lee, Y. S. Jou. Load Balanced Birkhoff-von Neumann Switches, Part I: One-Stage Buffering. *IEEE HPSR '01*, Dallas, May 2001.
8. I. Keslassy, S. T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, N. McKeown. Scaling Internet Routers Using Optics. *ACM SIGCOMM*, Karlsruhe, Germany, 2003.
9. D. Applegate, E. Cohen. Making routing robust to changing traffic demands: algorithms and evaluation. *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. 6, pp. 1193–1206, December 2006.
10. D. Applegate, L. Breslau, E. Cohen. Coping with Network Failures: Routing Strategies for Optimal Demand Oblivious Restoration. *ACM SIGMETRICS*, New York, USA, 2004.
11. A. Tizghadam and A. Leon-Garcia. A Robust Routing Plan to Optimize Throughput in Core Networks. *Managing Traffic Performance in Converged Networks, Lorne Mason, Tadeusz Drwiega and James Yan (Eds.), Springer*, pages 117–128, 2007.
12. A. Tizghadam and A. Leon-Garcia. A Graph Theoretical Approach to Traffic Engineering and Network Control Problem. In 21^{th} *International Teletraffic Congress (ITC21)*, Paris, France, September 2009.
13. L. C. Freeman. Centrality in Networks: I. Conceptual Clarification. *Social Networks*, (1): 215–239, 1978/79.
14. D. Koschtzki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, O. Zlotowski. Centrality Indices. *Network Analysis: Methodological Foundations, U. Brandes, T. Erlebach (Editors), Lecture Notes in Computer Science, Springer-Verlag*, vol. 3418, 2005.
15. L.C. Freeman, S.P. Borgatti, and D.R. White. Centrality in valued graphs: a measure of betweenness based on network flow. *Social Networks*, 13:141–154, 1991.
16. M. Newman. A Measure of Betweenness Centrality Based on Random Walks. *Social Networks*, 27:39–54, 2005.
17. A. Tizghadam, A. Leon-Garcia. Robust Network Planning in Nonuniform Traffic Scenarios. *Computer Communications*, vol. 34, no. 12, pp. 1436–1449, 2011.
18. C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and its Applications*. John Weily and Sons Inc., 1971.

19. A. Tizghadam and A. Leon-Garcia. Autonomic Traffic Engineering for Network Robustness. *IEEE Journal of Selected Areas in Communications (J-SAC)*, 28(1):39–50, January 2010.
20. A. Tizghadam and A Leon-Garcia. Survival Value of Communication Networks. In *INFOCOM Workshop on Network Science for Communications (NetSciCom)*, pages 1–6, Rio de Janeiro, Brazil, April 2009.
21. A. Tizghadam and A Leon-Garcia. Betweenness Centrality and Resistance Distance in Communication Networks. In *IEEE Network*, vol. 24, no. 6, pages 10–16, November-December 2010.
22. D. J. Klein and M. Randic. Resistance Distance. *Journal of Mathematical Chemistry*, 12(1):81–95, December 1993.
23. A. Ghosh, S. Boyd, and A. Saberi. Minimizing Effective Resistance of a Graph. *SIAM Review, problems and techniques section*, 50(1):37–66, February 2008.
24. L. Kleinrock. Queueing Systems, volume II. *John Wiley & Sons*, 1975.
25. A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, vol. 28, no. 3, pages 497–520, July 1960.
26. A. Preston, S. Cotter, C. Todorov. US Optical Networking Developments in Support of US Research and Education: FiberCo, a Case Study and a Bit on Optical International Exchange Points. CESNET, Prague Czech Republic, March 2006.
27. G. Leduca, H. Abrahamssone, S. Balona, S. Besslerb, M. D'Arienzoh, O. Delcourta, J. Domingo-Pascuald, S. Cerav-Erbasg, I. Gojmeracb, X. Masipd, A. Pescaph, B. Quoitinf, S. P. Romanoh, E. Salvadoric, F. Skivea, H. T. Tranb, S. Uhligf, and H. Umitg. An Open Source Traffic Engineering Toolbox. *Computer Communications*, 29(5):593–610, March 2006.
28. Dennis S. Bernstein. Matrix Mathematics. *Princeton University Press*, 2th edition, 2009.
29. M. Grant and S. Boyd. CVX: Matlab Software for Disciplined Convex Programming (Web Page and Software). $http://stanford.edu/\ boyd/cvx$. September 2008.
30. M. Grant and S. Boyd. Graph Implementations for Nonsmooth Convex Programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, $http://stanford.edu/\ boyd/graph_dcp.html$. *Lecture Notes in Control and Information Sciences, Springer*, 2008.
31. D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. Convex Analysis and Optimization. *Athena Scientific*, April 2003.
32. S. Boyd and L. Vandenberghe. Convex Optimization. *Cambridge University Press*, 2004.
33. L. Xiao and M. Johansson, and S. Boyd. Simultaneous Routing and Resource Allocation via Dual Decomposition. *IEEE Transactions on Communications*, vol. 52, no. 7, pages 1136–1144, July 2004.
34. A. Tizghadam and A. Leon-Garcia. On Random Walks in Direction-Aware Network Problems. *ACM SIGMETRICS Performance Evaluation Review (PER)*, vol. 38, no. 2, pages 9–11, September 2010.
35. N. Spring, R. Mahajan, D. Wetherall, T. Anderson. Measuring ISP Topologies with Rocketfuel. *IEEE/ACM Transactions on Networking (TON)*, vol. 12, no. 1, pp. 2–16, February 2004.
36. B. Fortz, M. Thorup. Increasing Internet Capacity Using Local Search. *, Computational Optimization and Applications*, vol. 29, pp. 13–48, 2004.
37. J. K. Johnson, M. Chertkov. A Majorization-Minimization Approach to Design of Power Transmission Networks. *arXiv:1004.2285*, September 2010.
38. D. A. Spielman, N. Srivastava. Graph Sparsication by Effective Resistances. *Proceedings of the 40th annual ACM Symposium on Theory of Computing STOC'08*, pages 563–568, 2008.

Chapter 5
Clique Relaxation Models in Social Network Analysis

Jeffrey Pattillo, Nataly Youssef, and Sergiy Butenko

Abstract Clique relaxation models that were originally introduced in the literature on social network analysis are not only gaining increasing popularity in a wide spectrum of complex network applications, but also keep garnering attention of mathematicians, computer scientists, and operations researchers as a promising avenue for fruitful theoretical investigations. This chapter describes the origins of clique relaxation concepts and provides a brief overview of mathematical programming formulations for the corresponding optimization problems, algorithms proposed to solve these problems, and selected real-life applications of the models of interest.

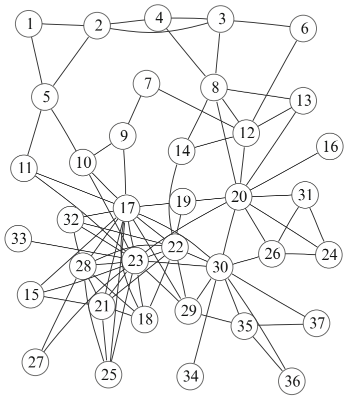
5.1 Introduction

Social networks represent certain types of social interaction, such as acquaintance, friendship, or collaboration between people or groups of people that are referred to as *actors*. In social networks, vertices (nodes, dots) usually stand for actors, and edges (arcs, links, lines) represent the pairwise relations or interactions between the actors. As an illustration of a social network, an example of a terrorist network is given in Fig. 5.1, which describes the connections between terrorists associated with the September 11, 2001 attack on the World Trade Center. The links were identified by Krebs [49] after the terrorist attack; however, the network is reconstructed

J. Pattillo (✉)
Department of Mathematics, Texas A&M University, College Station, TX 77843, USA
e-mail: jptlo@math.tamu.edu

N. Youssef • S. Butenko
Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843-3131, USA
e-mail: nyoussef@tamu.edu; butenko@tamu.edu

1	Wail Alshehri
2	Satam Suqami
3	Nabil al-Marabh
4	Raed Hijazi
5	Waleed Alshehri
6	Ahmed Alghamdi
7	Mohand Alshehri
8	Saeed Alghamdi
9	Fayez Ahmed
10	Mustafa Ahmed Al-Hisawi
11	Abdul Aziz Al-Omari
12	Hamza Alghamdi
13	Ahmed Alnami
14	Ahmed Al Haznawi
15	Mamoun Darkazanli
16	Mohamed Abdi
17	Marwan Al-Shehhi
18	Zakariya Essabar
19	Salem Alhazmi



20	Nawaf Alhazmi
21	Said Bahaji
22	Ziad Jarrah
23	Mohamed Atta
24	Abdussattar Shaikh
25	Mounir El Motassadeq
26	Khalid Al-Mihdhar
27	Zacarias Moussaoui
28	Ramzi Bin al-Shibh
29	Lofti Raissi
30	Hani Hanjour
31	Osama Awadallah
32	Agus Budiman
33	Ahmed Khalid Ibrahim Samir Al-Ani
34	Majed Moqed
35	Rayed Mohammed Abdullah
36	Faisal Al Salmi
37	Bandar Alhazmi

Fig. 5.1 The network representation of the data describing connections between terrorists associated with September 11 events

based on the information that was publicly available *before* September 11. Analysis of such networks, even though performed post-factum, may still be useful for understanding the structure of terrorist organizations and detecting similar structures to prevent potential terrorist attacks in the future.

One of the central concepts in *social network analysis* is the notion of a *cohesive subgroup*, which is a "tightly knit" subgroup of actors in a social network. During the last several decades, social network analysis methods in general, and social cohesion concepts in particular, have been employed in various branches of sociology in the contexts of crime detection/prevention and terrorist network analysis [12, 27, 31, 56, 68, 69], studies on the sociology of political systems and historical revolutionary movements [41, 65], epidemiology of sexually transmitted diseases [67], organizational management [32], and the sociology of worker solidarity [42].

Study of social cohesion can be traced back to one of the founding fathers of sociology, Emile Durkheim (1858–1917), who wrote [35]:

> The totality of beliefs and sentiments common to the average members of a society forms a determinate system with a life of its own. It can be termed the collective or common consciousness.

Durkheim found the notion of cohesiveness difficult to define rigorously,

> ...social solidarity is a wholly moral phenomenon which by itself is not amenable to exact observation and especially not to measurement.

However, a rigorous mathematical definition became possible with the introduction of sociometric analysis and graph theoretic methods into sociology that have led to the association of the notion of social cohesion with certain graph theoretic concepts. In particular, Luce and Perry [53] used *complete subgraphs* to model *social cliques*, which are defined as "an exclusive circle of people with a common purpose" [58]. It should be noted that the term "clique" was used in Hawthorne and Warner studies in the 1930s (see, e.g., [66, 70]), however, it referred to "socially perceived subgroups" and was not strictly defined [70].

While the notion of a clique embodies a "perfect" cohesive group, in which every two entities are connected to each other, this definition is overly conservative in many practical scenarios. Indeed,

1. One may not need to require every possible link to exist between elements of a cohesive subgroup.
2. The social network of interest may be built based on empirical data, which are prone to errors, so, even if a completely connected cohesive subgroup is sought for, it may be impossible to detect due to erroneous data.

To overcome this impracticality of the clique model, other graph-theoretic formalizations of the cohesive subgroup concept have been proposed in the literature. Not surprisingly, all these alternative definitions can be viewed as clique generalizations, each of which relaxes one of the elementary clique properties, such as familiarity, reachability, or robustness [4, 52, 59, 72]. Hence, we use the term "clique relaxations" in reference to such models. Since their introduction in the social network literature in the 1970s, the clique relaxation models received very little attention from researchers in graph theory, computer science, and operations research during the decades that followed. At the same time, methodology-wise, the social network literature dealing with these concepts was largely limited to giving their formal definition within a given context and providing illustrative examples on graphs with several vertices [83].

Recent progress in Internet and telecommunication technologies makes it possible to collect tremendous amounts of social interaction data dynamically and at virtually any level of detail, providing unprecedented opportunities to researchers interested in areas related to social network analysis. The availability of massive amounts of data that needs to be analyzed spurred the interest in developing effective algorithms capable of handling problems of practical scale. The need for advanced computational tools urged the recent progress in developing theoretical foundations for the models of interest and effective algorithms utilizing the theoretical achievements. The objective of this chapter is to provide a brief survey of results concerning optimization problems seeking to find maximum size clique relaxation structures in networks. More specifically, in this chapter we are primarily interested in mathematical programming formulations of the corresponding optimization problems and algorithms proposed for solving these problems. While in social network applications one may also be interested in detecting cohesive subgroups of sizes smaller than maximum, computing the largest cohesive subgroup is of special interest, since its size provides a global measure of cohesiveness of the corresponding network. We will also discuss selected applications of clique relaxation models outside of social network analysis and highlight some of the promising directions for future research in this active area.

The remainder of this chapter is organized as follows. Section 5.2 provides formal definitions of the clique relaxation concepts introduced in the context of social network analysis and discusses some of their basic properties. Mathematical programming formulations of the corresponding optimization problems, that aim to find clique relaxation structures of largest possible size in the given network, are outlined in Sect. 5.3. Section 5.4 surveys existing algorithms for solving the optimization problems of interest either exactly or approximately and discusses successful examples of application of some of these algorithms in practice. Extensions of the discussed methodologies to applications outside of the sociological realm are the focus of Sect. 5.5. Finally, Sect. 5.6 concludes the chapter.

5.2 Definitions and Properties of Clique Relaxation Models

Throughout this chapter, we consider a simple undirected graph $G = (V,E)$, where $V = \{1,...,n\}$ and $E \subseteq V \times V$, respectively, denote the sets of vertices and edges in G, with $|V| = n$ and $|E| = m$. G is said to be *complete* if for every $u,v \in V$ such that $u \neq v$, $(u,v) \in E$; in other words, $G = (V,E)$ is *complete* if $E = V \times V$ with $|E| = \binom{n}{2}$. Given a subset of vertices $S \subseteq V$, $G[S] = (S, E \cap (S \times S))$ denotes the *subgraph induced by S*, obtained by deleting all vertices in $V \setminus S$ and their corresponding incident edges from G.

A clique C is a subset of V such that the subgraph $G[C]$ induced by C on G is complete. A clique is called *maximal* if it is not contained in a larger clique, and it is called *maximum* if there is no larger clique in G. The size of the maximum clique in G is referred to as the clique number and is denoted by $\omega(G)$.

The notion of clique defined above embodies ideal properties of *reachability*, *familiarity*, *density*, and *robustness* discussed below in this chapter. Such structural properties are indeed very useful in modeling group cohesiveness within social networks. The idealized properties of cliques are, however, overly restrictive in practical applications. This observation has called for the development of clique relaxations, aiming at relaxing particular properties of cliques. We next introduce the definitions of the major graph properties that are needed in order to characterize the clique relaxations of interest. A path between two vertices $u,v \in V$ of length k is a sequence of distinct vertices $u = v_0, v_1, ..., v_k = v$, such that $(v_i, v_{i+1}) \in E, 0 \leq i \leq k-1$. Two paths are called *independent* if they only intersect at their ends. The length of the shortest path between two vertices $u,v \in V$ in G is represented by $d_G(u,v)$, referred to as the *distance* between u and v. The largest of the pairwise distances between vertices define the *diameter* of the graph, i.e., $\text{diam}(G) = \max_{u,v \in V} d_G(u,v)$. G is said to be *connected* if every two vertices in G are linked by a path in G. For any vertex $v \in V$, the *open neighborhood* $N(v) = \{u \in V | (u,v) \in E)\}$ defines the set of vertices adjacent to v in G, whereas $N[v] = \{v\} \cup N(v)$ denotes the *closed neighborhood* of v. The *degree* of a vertex $v \in V$, given by $\deg_G(v) = |N(v)|$, represents the number of edges emanating from v. Considering a subgraph $G[S]$ induced by $S \subseteq V$, the degree of a vertex $s \in S$ is given by $\deg_{G[S]}(s) = |N(s) \cap S|$. The minimum degree in a graph is denoted by $\delta(G)$. A subset of vertices D is called a *dominating set* if every vertex in the graph is either in D or has at least one neighbor

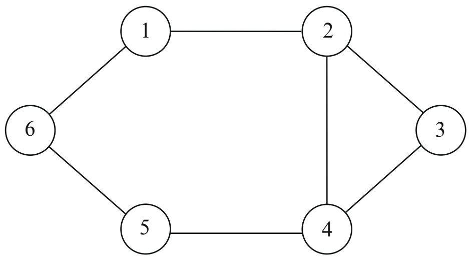


Fig. 5.2 A graph illustrating the difference between 2-cliques and 2-clubs, originally used in [4]

in D. The size of a smallest dominating set, also known as the domination number of G, is denoted by $\gamma(G)$. The *edge density* of G is the ratio of the number of edges to the total number of possible edges, i.e., $m/\binom{n}{2}$.

The concepts of k-cliques and k-clubs were among the first clique relaxations to be studied in the literature and were first introduced in [72] and [71]. They both relax the property of *reachability* desirable for a cohesive subgroup, that is, each member of a cohesive subgroup should be able to easily reach any other member of the subgroup. In graph-theoretic terms, this property can be expressed using the notions of distance and diameter. While pairwise distances between members of the clique is equal to one, k-cliques ensure that any two vertices within the subgraph are at most distance k from each other in the original graph. On the other hand, k-clubs require the diameter of the induced subgraph to be at most k. Next, we formally define these concepts.

Definition 5.1. A k-clique S is a subset of V such that, for all vertices $u, v \in S$, $d_G(u,v) \leq k$. The size of the largest k-clique is called the k-clique number and is denoted by $\tilde{\omega}_k(G)$.

Definition 5.2. A k-club S is a subset of V such that the induced subgraph $G[C]$ has a diameter of at most k. The size of the largest k-club is called the k-club number and is denoted by $\overline{\omega}_k(G)$.

Note that any k-club is a k-clique but the converse is not true. For example, the graph in Fig. 5.2 contains a 2-clique $C = \{1,2,3,4,5\}$, which is not a 2-club, since the distance between vertices 1 and 5 in the subgraph induced by C is 3. Based on this observation, Alba [4] concluded that the concept of k-clique lacks the “tightness” essential to applications in social networks, which motivated him to introduce the concept of a “sociometric clique”, later renamed to “k-clan” by Mokken [59]. According to their definition, a k-clique C is called an *k-clan* if the diameter of the induced subgraph $G(C)$ is no more than k. A considerable drawback in the k-clan definition is that for some graphs a k-clan may not exist [10]. Indeed, Fig. 5.3 shows a graph with two 2-cliques $\{1,2,3,4,5,6,7\}$ and $\{1,2,3,5,6,7,8\}$, neither of which is a 2-clan.

Familiarity is another important property one wants to have in a cohesive subgroup; ideally, every member of the group should be familiar with every other member of the group, which is the case for cliques. Known familiarity-based models either relax the minimum number of neighbors or the maximum number of non-neighbors within the group. The k-core concept, first introduced in

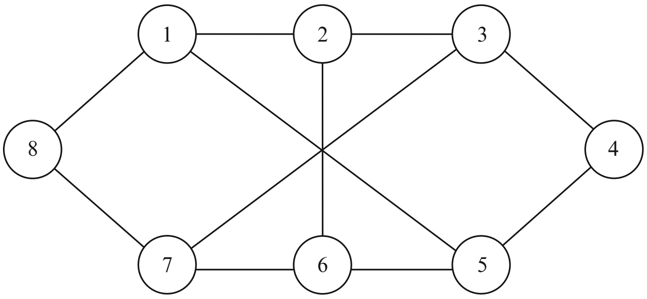


Fig. 5.3 A graph with no 2-clans proposed in [10]

[71], imposes a lower bound on the minimum degree within the subgraph, ensuring that each vertex in the group is connected to at least k other vertices in the group. The k-plex model introduced in [72], on the other hand, requires any subset of k vertices to dominate to entire group, hence restricting the number of non-neighbors per vertex to be at most $k-1$. The formal definitions of these familiarity-based relaxations are given next.

Definition 5.3. A k-core S is a subset of V such that, for all vertices $u,v \in S$, $\deg_{G[S]}(v) = |N(v) \cap S| \geq k$. The size of the k-core in the graph G is denoted by $\omega'_k(G)$.

Definition 5.4. A k-plex S is the maximum subset of V such that any set of k vertices in S dominates the group. Alternatively, a k-plex S is a subset of V such that, for all vertices $u,v \in S$, $\deg_{G[S]}(v) = |N(v) \cap S| \geq |S| - k$. The size of the largest k-plex is denoted by $\omega_k(G)$.

High *density* of connections is yet another defining characteristic of a cohesive subgroup that is perfectly reflected in the notion of clique, which can be alternatively defined as a subset of vertices inducing a subgraph with edge density of 1. This rather strict requirement on edge density could also be relaxed. Namely, instead of opting for cliques with density one, γ-quasi-cliques ensure a subgraph with a density *at least* γ, where $0 \leq \gamma \leq 1$.

Definition 5.5. A γ-quasi-clique or, simply, a γ-clique is a subset S of V such that the induced subgraph $G[S]$ has the edge density $|E \cap (S \times S)| / \binom{|S|}{2}$ of at least γ. The size of the largest γ-quasi-clique is denoted by $\omega_\gamma(G)$.

Finally, *robustness* of a group may be measured in terms of the number of vertices that need to be deleted to completely disconnect the group. Since any vertex within a clique is directly linked to any other vertex of the clique, disconnecting a clique would require removing all its vertices. A model known as the k-vertex connected subgraph relaxes the connectivity property, whereby removal of at least k vertices destroys connectivity. By Menger's theorem [33], a k-vertex connected group can be also defined as a subgraph such that any two of its vertices can be joined by k independent paths.

Definition 5.6. A k-vertex connected subgraph $G[S]$ is defined by a subset S of V such that the induced subgraph $G[S \setminus R]$ is connected for all subsets $R \subset S$, with $|R| < k$. The size of the largest k-connected subgraph is denoted by $\omega_{kc}(G)$.

The corresponding concept in social networks literature is the so-called *structural cohesion*, which is defined as the minimum number of members whose removal from a group would disconnect the group [60].

While, by definition, the above mentioned clique relaxations emphasize a single clique structural aspect, other models have been created ensuring different combinations of properties. Examples include the (λ,γ)-quasi-clique relaxing both density and degree requirements and the r-robust k-club additionally requiring r-connectivity in a k-club. For further readings, consult [23, 81]. Surveys [18] and [63] provide further information and references on the maximum clique and related problems.

5.3 Mathematical Programming Formulations

All the optimization problems of interest to this chapter can be formulated as (mixed) integer programs (MIP). Below in this section we provide some of the known formulations for all problems except for the maximum k-core and maximum k-vertex connected subgraph. No k-core formulation is given due to the fact that, as we will discuss in Sect. 5.4, the maximum k-core problem can be solved to optimality by a simple greedy algorithm that runs in polynomial time, whereas all other problems considered in this chapter are NP-hard. Thus, even though the maximum k-core problem can be easily formulated as an integer program, this is not the way this problem is typically approached. On the other hand, we are not aware of any publications proposing mathematical programming formulations or effective algorithms for the maximum k-vertex connected subgraph problem. Developing such formulations and algorithms is an interesting and important direction for future research, especially since the corresponding notion of structural cohesion is enjoying rapidly increasing popularity in the literature on social network analysis.

5.3.1 Maximum k-Clique

The maximum k-clique problem consists of finding the k-clique in the graph with largest cardinality $\omega_k(G)$. This problem can be formulated as a binary integer linear problem:

$$\tilde{\omega}_k(G) = \max \sum_{i \in V} x_i \tag{5.1}$$

$$\text{subject to} \quad x_i + x_j \le 1,\ \forall i,j \in V\ d_G(i,j) > k, \tag{5.2}$$

$$x_i \in \{0,1\},\ i \in V. \tag{5.3}$$

Each binary variable corresponds to a vertex in the graph and assumes unity if the corresponding vertex is included in the k-clique. Constraint (5.2) ensures that two vertices whose pairwise distance in the original graph G exceeds k are not both considered in the k-clique.

It should be noted that the maximum k-clique problem in graph G can be equivalently formulated as the maximum clique problem in graph G^k representing the k-th power of graph G, which is defined as follows. G^k has the same set of vertices as G, with edges connecting all pairs of vertices that are distance at most k from each other in G. Thus, the numerous mathematical programming formulations for the maximum clique problem available in the literature [18] can be adopted to obtain analogous formulations for the maximum k-clique problem.

5.3.2 Maximum k-Club

The maximum k-club problem looks for the largest k-club in the graph. This problem can be formulated as a binary integer linear problem:

$$\overline{\omega}_k(G) = \max \sum_{i \in V} x_i \tag{5.4}$$

$$\text{subject to} \quad x_i + x_j \leq 1 + \sum_{l:P^l_{i,j} \in \mathbb{P}_{i,j}} y^l_{ij},\ \forall (i,j) \notin E, \tag{5.5}$$

$$x_p \geq y^l_{ij},\ \forall p \in V(P^l_{ij}), P^l_{ij} \in \mathbb{P}_{ij}, (i,j) \notin E, \tag{5.6}$$

$$x_i \in \{0,1\},\ i \in V, \tag{5.7}$$

$$y^l_{ij} \in \{0,1\},\ \forall P^l_{ij} \in \mathbb{P}_{ij}, (i,j) \notin E. \tag{5.8}$$

$\mathbb{P}_{ij}$ represents an index set of all paths of length at most k, whereas P^l_{ij} denotes the path between i and j indexed at position $l \in \mathbb{P}_{ij}$. $V(P^l_{ij})$ denotes the set of vertices included inn path P^l_{ij}. Constraint (5.5) allows both vertices i and j into the solution if no edge but a path of length at most k exists between them, and only if y^l_{ij} is equal to 1 for at least one path between i and j. Hence, this formulation makes sure to include in the k-club all vertices along any one path of length at most k between any two vertices in the k-club.

More recently, Veremyev and Boginski [81] have developed a more compact formulation for the maximum k-club problem with $O(kn^2)$ constraints. We first present the special case with $k = 2$ and then generalize to any $k \geq 2$. Let $A = [a_{ij}]^n_{i,j=1}$, where $a_{ij} = 1$ if there exists an edge between vertices i and j, and 0 otherwise, denote the adjacency matrix of G. Vertices in a 2-club are either connected directly or through at most one other vertex l, which can be expressed using the following non-linear constraint:

$$a_{ij} + \sum_{l=1}^{n} a_{il} a_{lj} x_l \geq x_i x_j. \tag{5.9}$$

Simplifying the aforementioned constraint results in the following formulation:

$$\overline{\omega}_k(G) = \max \sum_{i \in 1}^{n} x_i \tag{5.10}$$

$$\text{subject to} \quad a_{ij} + \sum_{l=1}^{n} a_{il} a_{lj} x_l \geq x_i + x_j - 1, \forall i = 1,...,n; j = i+1,...,n, \tag{5.11}$$

$$x_i \in \{0,1\},\ i = 1,...,n. \tag{5.12}$$

The above formulation can be generalized for any $k \geq 2$, by allowing any two vertices in the k-club to be either directly linked or connected through at most $k-1$ vertices within the subgraph:

$$\overline{\omega}_k(G) = \max \sum_{i \in 1}^{n} x_i \tag{5.13}$$

$$\text{subject to} \quad a_{ij} + \sum_{l=1}^{n} a_{il} a_{lj} x_l + \sum_{l=1}^{n} \sum_{m=1}^{n} a_{il} a_{lm} a_{mj} x_l x_m + \cdots + \tag{5.14}$$

$$\sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_{k-2}=1}^{n} \sum_{i_{k-1}=1}^{n} a_{ii_1} a_{i_1 i_2} \ldots a_{i_{k-2} i_{k-1}} a_{i_{k-1} j} x_{i_1} \ldots x_{i_{k-1}} \tag{5.15}$$

$$\geq x_i + x_j - 1, \forall i = 1,...,n; j = i+1,...,n, \tag{5.16}$$

$$x_i \in \{0,1\},\ i = 1,...,n. \tag{5.17}$$

5.3.3 Maximum k-Plex

The maximum k-plex problem finds the k-plex in the graph with maximum cardinality. This problem can be formulated as a binary integer linear problem, where $\deg_{\overline{G}}(i) = |V \backslash N[i]|$ denotes the degree of vertex i in the complement graph $\overline{G} = (V, \overline{E})$ [9]:

$$\omega_k(G) = \max \sum_{i \in V} x_i \tag{5.18}$$

$$\text{subject to} \quad \sum_{j \in V \backslash N[i]} x_j \leq (k-1)x_i + \deg_{\overline{G}}(i)(1 - x_i),\ \forall i \in V, \tag{5.19}$$

$$x_i \in \{0,1\},\ i \in V. \tag{5.20}$$

Binary variables indicate whether or not a vertex is included in the maximum k-plex. Constraint (5.19) expresses the fact that if vertex i is included in the k-plex, then it has at most $k-1$ non-neighbors, otherwise the constraint becomes redundant.

5.3.4 Maximum Quasi-clique

The maximum γ-quasi-clique problem can be formulated as the following mixed-integer problem with linear objective and a single quadratic constraint [64]:

$$\omega_\gamma(G) = \max \sum_{i=1}^{n} x_i \tag{5.21}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij} x_i x_j \geq \gamma \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i x_j, \tag{5.22}$$

$$x_i \in \{0,1\},\ i = 1,\dots,n. \tag{5.23}$$

Two linearizations of this formulation are proposed in [64]. The first, standard, linearization is based on defining a new variable $x_{ij} = x_i x_j$ and using an equivalent (for binary variables) representation of this quadratic equation in terms of three linear inequalities. This results in the following mixed integer linear programming formulation with $n(n-1)/2$ variables and $\frac{3}{2}n(n-1)+1$ constraints:

$$\omega_\gamma(G) = \max \sum_{i=1}^{n} x_i, \tag{5.24}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \sum_{j=i+1}^{n} (\gamma - a_{ij}) x_{ij} \leq 0, \tag{5.25}$$

$$x_{ij} \leq x_i,\ \ x_{ij} \leq x_j,\ \ x_{ij} \geq x_i + x_j - 1,\ \ j > i = 1,\dots,n \tag{5.26}$$

$$x_{ij} \geq 0,\ x_i \in \{0,1\},\ j > i = 1,\dots,n. \tag{5.27}$$

The second linearization is based on rewriting the single constraint of formulation (5.21)–(5.23) in the form

$$\sum_{i=1}^{n} x_i \left(\gamma x_i + \sum_{j=1}^{n} (a_{ij} - \gamma) x_j \right) \geq 0. \tag{5.28}$$

and introducing a new variable y_i for $i = 1,\dots,n$ as follows:

$$y_i = x_i \left(\gamma x_i + \sum_{j=1}^{n} (a_{ij} - \gamma) x_j \right). \tag{5.29}$$

Again, replacing this quadratic equality constraint with several linear inequality constraints, we obtain the following MIP with $2n$ variables, n of which are binary and n continuous, and $4n+1$ constraints:

$$\omega_\gamma(G) = \max \sum_{i=1}^{n} x_i \tag{5.30}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} y_i \geq 0, \tag{5.31}$$

$$y_i \leq u_i x_i, \;\; y_i \geq l_i x_i, \;\; i = 1, \ldots, n, \tag{5.32}$$

$$y_i \geq \gamma x_i + \sum_{j=1}^{n} (a_{ij} - \gamma) x_j - u_i(1 - x_i), \;\; i = 1, \ldots, n, \tag{5.33}$$

$$y_i \leq \gamma x_i + \sum_{j=1}^{n} (a_{ij} - \gamma) x_j - l_i(1 - x_i), \;\; i = 1, \ldots, n, \tag{5.34}$$

$$x_i \in \{0,1\}; \;\; y_i \in \mathbb{R}, \; i = 1, \ldots, n. \tag{5.35}$$

5.4 Algorithms for Detection of Clique Relaxation Structures

5.4.1 k-Clique

Balasundram et al. [10] have shown that the maximum k-clique problem is NP-hard for any positive integer k. A natural way of attacking this problem is by reducing it to the maximum clique problem in the k-th power of the graph, G^k. The maximum clique problem has been well studied in the literature [18], and all the methods available for this problem can be easily adopted for solving the maximum k-clique problem. Therefore, below we mention some of the numerous algorithms proposed for solving the clique problem. It should be noted that G^k may have a much higher edge density than G, and most of the known algorithms for the maximum clique problem tend to perform better on sparse graphs. Thus, the maximum k-clique problem appears to be more challenging than the maximum clique problem.

Some of the most notable exact combinatorial algorithms for the maximum clique problem have been proposed by Bron and Kerbosch [22], Balas and Yu [8], Applegate and Johnson [5], Carraghan and Pardalos [25], Babel [6], Balas and Xue [7], Wood [84], Sewell [73], Östergård [62], and Tomita and Kameda [78]. Due to its simplicity and effectiveness, especially on sparse graphs, implementation of Carraghan–Pardalos algorithm developed by Applegate and Johnson [5] was used as a benchmark exact algorithm for the maximum clique problem in the Second DIMACS Implementation Challenge [44]. Östergård [62] proposed an enhanced version of the Carraghan–Pardalos algorithm, which uses approximate coloring for the upper bound. The resulting algorithm implemented in software called *cliquer* is considered one of the fastest exact approaches for solving the maximum clique problem. More recently, Tomita and Kameda [78] claim that for randomly generated test instances and DIMACS benchmark instances [34] the algorithm they propose outperforms all other competitive algorithms published literature at the time of this writing, with Östergård's algorithm being the close second.

For a broad overview of heuristics used for the maximum clique problem, see [63]. Greedy heuristics have been developed based on either a sequential addition of vertices until the clique is built or a removal of vertices from a larger graph. They typically use local information, such as vertex degree, to order the vertices in the greedy scheme [45, 48, 79]. Simulated annealing [1, 43] and tabu search [11, 40, 74, 75] have proven very successful, as have hybrid implementations combining simulated annealing and greedy heuristics [26], [46]. These hybrid methods outperform most greedy heuristics on random graphs with up to 1,000 vertices both in time and quality of solution.

Several algorithms for enumerating all maximal cliques in a graph are also available in the literature [22, 29, 47, 51, 55]. For a comprehensive survey of the maximum clique problem formulations, exact and heuristic algorithms as of 1999, the reader is referred to [18].

5.4.2 k-Club

The maximum k-club problem defined on an undirected graph has been proven NP-hard as a result of a polynomial reduction from the clique problem [20]. Moreover, it has been shown in [10] that the problem remains NP-hard even when restricted to graphs of diameter $k+1$. This has motivated several attempts at constructing efficient heuristics. For instance, Bourjolly et al. [19] have proposed three greedy heuristics, DROP, k-CLIQUE & DROP and CONSTELLATION. The main idea behind DROP consists of sequentially deleting nodes from the original graph until obtaining a k-club. At each iteration, the heuristic computes the shortest path lengths between all pairs of nodes. A single node is then ruled out of the graph which has the largest number of vertices with distance from it is larger than k. This process is repeated until all pairwise shortest path lengths in the remaining graph are k or less, hence, defining a k-club. With an overall time complexity of $O(|V|^3|E|)$, DROP yields near-optimal solutions for high-density graphs. A better performance for lower density graphs for $k = 3$ or 4 is obtained by applying DROP to the largest k-clique in the original graph. The obtained algorithm is referred to as k-CLIQUE & DROP and has exponential complexity due to the initial step of finding the largest k-clique in G. CONSTELLATION, on the other hand, stems from the main idea that a star graph forms a 2-club. The heuristic first finds the largest star subgraph in G centered around the vertex with maximum degree in the graph, resulting in a $t+1$-club, where $t = 1$. Then, at iteration t, it identifies an external vertex with the largest number of neighbors not yet included in the previously obtained t-club. Appending these vertices to the subgraph results in a $t+1$-club. The algorithm is then repeated until a k-club is reached. CONSTELLATION performs well for low density graphs when $k = 2$, with an overall complexity of $O(k(|V| + |E|))$.

While these heuristics find good solutions efficiently for relatively large graphs, exact algorithms have also been developed within the framework of Branch-and-Bound (B&B), attempting to solve the maximum k-club problem on rather smaller graphs. Bourjolly et al. [20] proposed using a single iteration of the DROP heuristic to guide the branching step with upper bounds relying on solutions to the maximum stable set problem for an auxiliary graph. The auxiliary graph in this procedure consists of all vertices in G with edges between vertices if their distance in G exceeds k. While DROP can be performed efficiently, the bounding step at each node of the B&B tree may be expensive, since the maximum stable set problem is known to be NP-complete [39, 44]. This algorithm solved instances with no more than 200 vertices to optimality and reported more computationally efficient solutions for denser graphs. A more recent approach has been developed by Mahdavi and Balasundaram [54] based on a dual coloring problem. Their algorithm employs DROP and CONSTELLATION to initialize the incumbent solution. Vertex dichotomy is proposed as a branching strategy whereby the vertex with minimum number of k-neighbors is branched upon at each node of the B&B tree. Upper bounding, on the other hand, employs a distance k-coloring approach. Proper coloring is achieved by two heuristics, the first of which is DSATUR (refer to [21]), applied at top levels of the B&B tree. For lower levels, a simple greedy heuristic is used, coloring the highest degree vertex in the power of the graph with the color not yet assigned to its neighbors. Computations were reported for the same test-bed of instances as [20] suggesting a better performance for $k = 3$ than for $k = 2$.

5.4.3 k-Core

The maximum k-core problem has been proven solvable in polynomial time. In fact, a simple greedy algorithm is capable of generating optimal solutions as follows: First, a vertex v of minimum degree $\delta(G)$ is picked, and if $\delta(G) \geq k$, then the whole graph is a k-core. If $\delta(G) < k$, then that vertex cannot be in a k-core. Hence, this vertex is deleted updating the graph $G := G - \{v\}$ and continuing recursively until a maximum k-core or the empty set is found.

Detecting the maximum k-core is often used as a pre-processing step for solving optimization problems seeking cliques or other clique relaxation structures. This is due to the fact that some of these structures are guaranteed to be a part of the largest k-core for a certain value of k. For example, since a k-plex of size s cannot contain a vertex of degree less than $s - k$, when searching for such a k-plex, we can recursively remove all vertices of degree less than $s - k$. Hence, any k-plex of size s is a part of the largest $(s - k)$-core. The process of computing the largest k-core as a preprocessing or scale-reduction step for solving another problem is known as *peeling* in the literature and has been successfully applied for solving the maximum clique problem [2] and the maximum k-plex problem [9].

5.4.4 k-Plex

The maximum k-plex problem was proven NP-hard for any fixed positive integer k in [9]. Because of the intractability of the problem, heuristics have been developed both to find "good" solutions and to assist branch and bound for finding exact solutions. McClosky [57] employed a simple heuristic to find cliques, which were then extended to maximal k-plexes, for use as a lower bound for branch and bound. In [80], a heuristic was used to help prune a branch and bound tree for k-plex. When the size of the maximum k-plex exceeds $2k-2$, it will have diameter 2. Assuming this to be true greatly reduced the candidate set as they built a k-plex one vertex at a time in branch and bound.

Most currently available exact approaches for solving the maximum k-plex problem are adaptations of either branch and bound or branch and cut. Balasundaram et al. [9] formulate the maximum k-plex problem as an integer linear program and use valid inequalities based on independent sets of size at least k. These inequalities were generated by a simple greedy algorithm both for the whole problem and at local branches in the search tree. They successfully ran a branch and cut algorithm on large-scale instances of real life social networks known as the Erdös graphs and successfully solved the maximum 2-plex problem on graphs with 80% density and 350 vertices in less than 8 h.

The focus of [61] is on the minimum d-bounded-degree deletion problem, which yields a k-plex in the compliment graph. It generalizes the relationship between clique and vertex cover. They use an algorithm that finds a kernel of the minimum d-bounded degree deletion problem to guide their branching in branch and bound. A kernel is a subgraph of vertices that must contain the optimal solution and finding it can significantly reduce the size of the branch and bound tree. They report results superior to [9] with guided branching on the Erdös graphs and comparable results for DIMACS graphs as those reported in [9].

In [57], some of the most successful techniques for solving the maximum clique problem were adapted as part of a branch and bound algorithm for k-plex. They develop a co-k-plex coloring problem that bounds the size of the k-plex in the same way the coloring problem serves to bound the clique number of a graph. They use this to help provide upper bounds in adaptations of the basic clique algorithm [25] and Östergard's algorithm [62] for solving the maximum k-plex problem. They successfully solved instances of the maximum k-plex problem on DIMACS graphs for $k = 2,3,4$ but in general the results were not as successful as the branch and cut algorithm of [9] for larger graphs.

The Carraghan–Pardalos and Östergard algorithms for the maximum clique problem were also the inspiration of a more general algorithm for solving any problem that displays heredity in [80]. The key difference from [57] is including a pruning technique based on bounds for the size of a k-plex within subgraphs induced by subsets of vertices of the form $\{v_i, \ldots, v_n\}$. These are developed naturally as the algorithm runs. If i is the lowest index on any member of the candidate set, we calculate the size of the largest k-plex in $\{v_i, \ldots, v_n\}$ as part of the algorithm, and while it may not be achieved for a given candidate set within $\{v_i, \ldots, v_n\}$, it is often significantly lower than the weight of the entire candidate set. In [57], they prune branches of the search tree where the weight of the current solution and the entire candidate set was less than the current optimal solution. Trukhanov et al. [80] use both this pruning technique and pruning where the weight of the candidate set is replaced with the known size of the optimal solution using the vertex with lowest index. The algorithm significantly outperformed the techniques of [9] and [57] in nearly every instance of DIMACS graphs on which they were run. Graphs of up to 500 vertices were solved optimally for $k = 2, 3, 4$ and even larger graphs were solved optimally for $k = 2$.

Recently, Wu and Pei [85] developed a parallel algorithm for enumeration of all the maximal k-plexes in the graph. Their work emphasizes the parallelization of the algorithm and comparison between serial and parallel algorithms performance. Unfortunately, they do not provide any numerical results for the well-known benchmark instances.

5.4.5 *Quasi-clique*

The maximum γ-quasi-clique problem was shown to be NP-hard for any $\gamma \in (0, 1)$ in [64]. Due to the lack of structural properties in quasi-cliques that are utilized in exact algorithms for other problems of interest, the maximum quasi-clique problem is extremely challenging to solve to optimality. In fact, to the best of our knowledge, the only published exact approaches for this problem are based on using the MIP formulations presented in Sect. 5.3.4 in conjunction with an off-the-shelf solver [64]. This allows to solve medium-scale problem instances with several hundreds of vertices and low edge density to optimality.

To solve large-scale instances, heuristics are commonly used in practice. Since quasi-clique possesses the weak heredity property, greedy randomized adaptive search procedures (GRASP) [36, 37] appear to be a natural method of choice. In short, GRASP is a heuristic framework based on the following idea: at each iteration it constructs a greedy randomized solution and searches the neighborhood of each constructed solution in order to find the corresponding local optimum. The effectiveness of this method for solving the maximum quasi-clique problem is confirmed by the results of experiments with very large networks reported in the literature. One remarkable example is due to Abello et al. [2,3], who analyzed the so-called *call network*. This network represents phone call data; it has phone numbers as its vertices, and two vertices are connected by an edge if there was a phone call between the corresponding numbers within a specified period of time. Abello et al.

analyzed a call network representing the data from AT&T telephone billing records. This one-day call network had 53,767,087 vertices and over 170 millions of edges. The network had 3,667,448 connected components, 302,468 of which had more than three vertices. The largest connected component had 44,989,297 vertices. First, Abello et al. attempted to solve the maximum clique problem in this giant connected component. For this purpose, they ran 100,000 GRASP iterations on ten parallel processors, which took about one and a half days. Only 14,141 of the 100,000 cliques they generated were distinct, with the largest clique detected being of size 32. They also used GRASP to solve the maximum γ-clique problem for $\gamma = 0.9, 0.8, 0.7$, and 0.5 in the giant connected component. The largest γ-cliques they detected had the sizes of 44, 57, 65, and 98, respectively.

5.5 Extensions

While this chapter is motivated by developments in social network analysis, clique-like structures naturally arise in many other important applications of complex networks, in which one is looking for large, "tightly knit" groups of elements. Depending on the particular application, such groups are often referred to as *clusters*, *modules*, *complexes*, *communities*, etc. If the elements in the considered complex system are represented as vertices (nodes) and the relationships between the elements are represented as edges (links, arcs), then, depending on the structure of interest for the particular application, clusters can be naturally described using some of the clique relaxation concepts described above.

Recent work exploiting some of the clique relaxation models in this emerging area of *network-based data mining* [14, 30] has been abundant. In particular, these concepts have been used in studying structural properties of stock markets [16, 17], unraveling molecular structures to facilitate drug discovery and compound synthesis [24, 38], and for identifying frequently occurring patterns in data sets (modeled as graphs) [15, 82]. In biology, quasi-cliques have been used to detect large clusters in protein interaction networks [13]. A survey of applications of clique detection models in biochemistry and genomics is given in [24]. In internet research, cohesive subgroups correspond to collections of densely connected web sites [77]. This helps to organize topically related web sites and thus facilitate faster search and retrieval of information from the web. In wireless communication, clustering the *connectivity graph* of a wireless network is used to introduce a hierarchy, which facilitates routing of information through the network [76]. Clique and other low diameter models have been used to define a cluster in a wireless network [28, 50]. These are just a few examples of recent publications in the rapidly growing body of literature dealing with applications of cliques and clique relaxations in a wide range of settings.

5.6 Conclusion

Social network analysis is emerging as an important tool in network-based data mining, which is applied to a wide variety of settings ranging from biological systems to finance, and studying clique relaxation models is one of the cornerstones of this methodology. The aim of this chapter is to open the door to this rich and, at the same time, largely unexplored research avenue for an interested reader. Even though most of the discussed clique relaxation models have been around for several decades in the social network literature, many of the theoretical, algorithmic, computational, and applied aspects of these models are only beginning to be addressed.

To illustrate the potential impact the research on clique relaxations may have on areas beyond social network analysis, we turn to history. While complete subgraphs had been studied by mathematicians earlier, introduction of the term "clique" to graph theory was triggered by the developments in social sciences mentioned in the beginning of this chapter. Since then, the concept of a clique has been widely used in a variety of applied settings and is central to some major theoretical and algorithmic developments in graph theory, computer science and operations research. We believe that clique relaxation models provide an excellent opportunity for even more important developments.

References

1. E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons Incorporated, Chichester, UK, 1989.
2. J. Abello, P.M. Pardalos, and M.G.C. Resende. On maximum clique problems in very large graphs. In J. Abello and J. Vitter, editors, *External memory algorithms and visualization*, volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 119–130. American Mathematical Society, 1999.
3. J. Abello, M.G.C. Resende, and S. Sudarsky. Massive quasi-clique detection. In S. Rajsbaum, editor, *LATIN 2002: Theoretical Informatics*, pages 598–612, London, 2002. Springer-Verlag.
4. R.D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3:113–126, 1973.
5. D. Applegate and D.S. Johnson. Maximum clique solver implementation, dfmax.c. ftp://dimacs.rutgers.edu/pub/challenge/graph/solvers/.
6. L. Babel. Finding maximum cliques in arbitrary and in special graphs. *Computing*, 46(4): 321–341, 1991.
7. E. Balas and J. Xue. Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica*, 15:397–412, 1996.
8. E. Balas and C. Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal of Computing*, 15:1054–1068, 1986.
9. B. Balasundram, S. Butenko, and I.V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59:133–142, 2011.
10. B. Balasundram, S. Butenko, and S. Trukhanov. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, 10:23–29, 2005.

11. R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. Technical report, Algorithmica, 2001.
12. N. Berry, T. Ko, T. Moy, J. Smrcka, J. Turnley, and B. Wu. Emergent clique formation in terrorist recruitment. *The AAAI-04 Workshop on Agent Organizations: Theory and Practice, July 25, 2004, San Jose, California*, 2004. Online: http://www.cs.uu.nl/ virginia/aotp/papers.htm.
13. M. Bhattacharyya and S. Bandyopadhyay. Mining the largest quasi-clique in human protein interactome. In *IEEE International Conference on Artificial Intelligence Systems*, pages 194–199, Los Alamitos, CA, USA, 2009. IEEE Computer Society.
14. V. Boginski. Network-based data mining: Operations research techniques and applications. In J. Cochran, editor, *Encyclopedia of Operations Research and Management Science*. Wiley, 2011.
15. V. Boginski, S. Butenko, and P. Pardalos. Network models of massive datasets. *Computer Science and Information Systems*, 1:79–93, 2004.
16. V. Boginski, S. Butenko, and P. Pardalos. Statistical analysis of financial networks. *Computational Statistics & Data Analysis*, 48:431–443, 2005.
17. V. Boginski, S. Butenko, and P. Pardalos. Mining market data: a network approach. *Computers & Operations Research*, 33:3171–3184, 2006.
18. I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 1–74, Dordrecht, The Netherlands, 1999. Kluwer Academic Publishers.
19. J.M. Bourjolly, G. Laporte, and G. Pesant. Heuristics for "nding k-clubs in an undirected graph. *Computers and Operations Research*, 27:559–569, 2000.
20. J.M. Bourjolly, G. Laporte, and G. Pesant. An exact algorithm for the maximum k-club problem in an undirected graph. *European Journal of Operational Research*, 138:21–28, 2002.
21. D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22: 251–256, 1979.
22. C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques on an undirected graph. *Communications of ACM*, 16:575–577, 1973.
23. M. Brunato, H. Hoos, and R. Battiti. On effectively finding maximal quasi-cliques in graphs. In V. Maniezzo, R. Battiti, and J.P. Watson, editors, *Proc. 2nd Learning and Intelligent Optimization Workshop, LION 2*, volume 5313 of *LNCS*. Springer Verlag, 2008.
24. S. Butenko and W. Wilhelm. Clique-detection models in computational biochemistry and genomics. *European Journal of Operational Research*, 173:1–17, 2006.
25. R. Carraghan and P. Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9:375–382, 1990.
26. M. Chams, A. Hertz, and A. de Werra. Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research*, 32:260–266, 1987.
27. H. Chen, W. Chung, J. J. Xu, G. Wang, Y. Qin, and M. Chau. Crime data mining: A general framework and some examples. *Computer*, 37(4):50–56, 2004.
28. Y. P. Chen, A. L. Liestman, and J. Liu. Clustering algorithms for ad hoc wireless networks. In Y. Pan and Y. Xiao, editors, *Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing*, pages 145–164. Nova Science Publishers, New York, 2005.
29. N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14:210–223, 1985.
30. D. J. Cook and L. B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000.
31. R. H. Davis. Social network analysis: An aid in conspiracy investigations. *FBI Law Enforcement Bulletin*, 50(12):11–19, 1981.
32. A. H. Dekker. Social network analysis in military headquarters using CAVALIER. In *Proceedings of Fifth International Command and Control Research and Technology Symposium*, pages 24–26, Canberra, Australia, 2000.
33. R. Diestel. *Graph Theory*. Springer-Verlag, Berlin, 3 edition, 2006.
34. DIMACS. Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge. http://dimacs.rutgers.edu/Challenges/, 1995.

35. E. Durkheim. *The Division of Labor in Society*. Free Press, New York, [1893] 1984. Translated by W. D. Halls.
36. T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.
37. T. A. Feo, M. G. C. Resende, and S. H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878, 1994.
38. I. Fischer and T. Meinl. Graph based molecular data mining - an overview. In W. Thissen, P. Wieringa, M. Pantic, and M. Ludema, editors, *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, pages 4578–4582, Piscataway, NJ, 2004. IEEE.
39. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.
40. M. Gendreau, P. Soriano, and L. Salvail. Solving the maximum clique problem using a tabu search approach. *Ann. Oper. Res.*, 41:385–403, May 1993.
41. R. V. Gould. Multiple networks and mobilization in the paris commune, 1871. *American Sociological Review*, 56:716–729, 1991.
42. R. Hodson. *Dignity at Work*. Cambridge University Press, Cambridge, England, 2001.
43. S. Homer and M. Pinedo. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*, chapter Experiments with polynomial-time clique approximation algorithms on very large graphs, pages 147–167. American Mathematical Society, 1996.
44. D. J. Johnson and M. A. Trick, editors. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society, Boston, MA, USA, 1996.
45. D.S. Johnson and C.R. Aragon. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.
46. D.S. Johnson, C.R. Aragon, L.A. McGoech, and C. Shevon. Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39:378–406, 1991.
47. H.C. Johnston. Cliques of a graph – variations of Bron-Kerbosch algorithm. *Int. J. Comput. Inf. Sci.*, 5:209–238, 1976.
48. R. Kopf and G. Ruhe. A computational study of the weighted independent set problem for general graphs. *Foundations of Control Engineering*, 12:167–180, 1987.
49. V. Krebs. Mapping networks of terrorist cells. *Connections*, 24:45–52, 2002.
50. P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. *ACM SIGCOMM Computer Communication Review*, 27(2): 49–64, 1997.
51. E. Loukakis and C. Tsouros. A depth first search algorithm to generate the family of maximal independent sets of a graph lexicographically. *Computing*, 27:249–266, 1981.
52. R.D. Luce. Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 15:169–190, 1950.
53. R.D. Luce and A.D. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14:95–116, 1949.
54. F. Mahdavi and B. Balasundaram. On inclusionwise maximal and maximum cardinality k-clubs in graphs. *Preprint*, 2010.
55. K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *Proc. Ninth Scandinavian Workshop Algorithm Theory (SWAT '04)*, pages 260–272, 2004.
56. D. McAndrew. The structural analysis of criminal networks. In D. Canter and L. Alison, editors, *The Social Psychology of Crime: Groups, Teams, and Networks, Offender Profiling Series, III*, Aldershot, UK, 1999. Dartmouth.
57. B. McClosky and I. Hicks. Combinatorial algorithms for the maximum k-plex problem. *Journal of Combinatorial Optimization*, to appear.
58. G. A. Miller. WordNet, Princeton University. http://wordnet.princeton.edu, 2009.
59. R.J. Mokken. Cliques, clubs and clans. *Quality and Quantity*, 13:161–173, 1979.
60. J. Moody and D. R. White. Structural cohesion and embeddedness: A hierarchical concept of social groups. *American Sociological Review*, 68:103–127, 2003.

61. H. Moser, R. Niedermeier, and M. Sorge. Algorithms and experiments for clique relaxations - finding maximum *s*-plexes. In *Proceedings of the 8th International Symposium on Experimental Algorithms*, pages 233–244, 2009.
62. P. R. J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120:197–207, 2002.
63. J. Pattillo and S. Butenko. Clique, independent set, and graph coloring. In J. Cochran, editor, *Encyclopedia of Operations Research and Management Science*, pages 3150–3163. Wiley, 2011.
64. J. Pattillo, A. Veremyev, S. Butenko, and V. Boginski. On the maximum quasi-clique problem. Working paper.
65. P. Paxton. Is social capital declining in the united states? a multiple indicator assessment. *American Journal of Sociology*, 105:88–127, 1999.
66. F. J. Roethlisberger and W. J. Dickson. *Management and the Worker*. Harvard University Press, Cambridge, MA, 1939.
67. R. B. Rothenberg, J. J. Potterat, and D. E. Woodhouse. Personal risk taking and the spread of disease: Beyond core groups. *Journal of Infectious Diseases*, 174 (Supp. 2):S144–S149, 1996.
68. M. Sageman. *Understanding Terrorist Networks*. University of Pennsylvania Press, Philadelphia, PA, 2004.
69. R. J. Sampson and B. W. Groves. Community structure and crime: Testing social-disorganization theory. *American Journal of Sociology*, 94:774–802, 1989.
70. J. Scott. *Social Network Analysis: A Handbook*. Sage Publications, London, 2 edition, 2000.
71. S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.
72. S. B. Seidman and B. L. Foster. A graph theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978.
73. E. C. Sewell. A branch and bound algorithm for the stability number of a sparse graph. *INFORMS Journal on Computing*, 10(4):438–447, 1998.
74. P. Soriano and M. Gendreau. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge*, chapter Tabu search algorithms for the maximum clique problem, pages 221–242. American Mathematical Society, 1996.
75. P. Soriano and M. Gendreau. Diversification strategies in tabu search algorithms for the maximum clique problem. *Annals of Operations Research*, 63:189–207, 1996.
76. I. Stojmenovic, editor. *Handbook of Wireless Networks and Mobile Computing*. Wiley Inter-Science, 2002.
77. L. Terveen, W. Hill, and B. Amento. Constructing, organizing, and visualizing collections of topically related, web resources. *ACM Transactions on Computer-Human Interaction*, 6:67–94, 1999.
78. E. Tomita and T. Kameda. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *Journal of Global Optimization*, 37(1):95–111, 2007.
79. E. Tomita, S. Mitsuma, and H. Takahashi. Two algorithms for finding a near-maximum clique. Technical report, UEC-TR-C1, 1988.
80. S. Trukhanov, B. Balasundaram, and S. Butenko. Exact algorithms for hard node deletion problems in graphs. Working paper, 2011.
81. A. Veremyev and V. Boginski. Identifying large robust network clusters via new compact formulations of maximum *k*-club problems. *European Journal of Operational Research*, in revision, 2010.
82. T. Washio and H. Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.
83. S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge University Press, New York, 1994.
84. D. R. Wood. An algorithm for finding a maximum clique in a graph. *Operations Research Letters*, 21(5):211–217, 1997.
85. B. Wu and X. Pei. A parallel algorithm for enumerating all the maximal *k*-plexes. In J. G. Carbonell and J. Siekmann, editors, *Emerging Technologies in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, chapter 47, pages 476–483. Springer, Berlin, Germany, 2007.

Part II
Complex Communication Networks

Chapter 6
Application Traffic Activity Graph Analysis

Yu Jin, Esam Sharafuddin, and Zhi-Li Zhang

6.1 Introduction

Understanding and analyzing traffic characteristics are fundamental to the design, development and implementation of networks. The traditional emphasis of network traffic analysis has been on the statistical properties of traffic, leading to the important discoveries such as heavy-tails and long range dependence. As networking technologies continue to mature and evolve, and more sophisticated network applications are invented and deployed, operating, managing and securing networks have become increasingly challenging tasks, and require us to understand, analyze and model the *behavioral characteristics* of network traffic, such as communication patterns, interaction structures and trends of applications, users and other entities in the networks.

While traffic analysis for network security and management has been an active area of research, the majority of earlier work has focused on specific problems or aspects such as detecting heavy-hitters, identifying peer-to-peer (P2P) applications, and generating packet-level malware signatures (see, e.g., [9, 21, 29]) that are driven primarily by certain security application needs. There are relatively few studies which consider the traffic as a whole to extract general behavioral characteristics. Examples include individual (5-tuple) flow-level traffic clustering [19], aggregate PoP-level origin–destination (O-D) flow characterization and anomaly detection [11, 12], and host-level traffic behavior profiling using graph-theoretical [24] or information-theoretical [8] approaches.

In this chapter, we study *network-wide communication patterns* among hosts that are engaging in certain types of communications or applications using *traffic activity graphs* (TAGs). In a TAG, nodes are IP addresses (hosts) and edges are

Y. Jin, • E. Sharafuddin • Z. Zhang (✉)
Department of Computer Science and Engineering, University of Minnesota, Twin Cities
e-mail: yjin@cs.umn.edu; shara@cs.umn.edu; zhzhang@cs.umn.edu

observed flows that represent certain communications or interactions of interest among the IP addresses (hosts). Depending on the purpose of study, various criteria may be used to select flows of interest, and construct different TAGs that capture the relevant traffic activities among hosts under study. For example, using the NetFlow records collected at our campus border router, in this chapter we model the communication patterns and interactions between hosts within our campus network and those outside hosts in the observed traffic using *bipartite* TAGs, where one set of nodes represent the inside hosts and another set of nodes represent the outside hosts, and edges between these two sets of hosts represent certain flows *selected based on ports that are associated with an application of interest*. We refer to these (bipartite) graphs as *application* TAGs. Examples of such TAGs include HTTP, Email, DNS, peer-to-peer (P2P), online chat and gaming applications.

In general, TAGs derived from real network data are large, sparse, seemingly complex and richly connected. For instance, when the number of nodes is large, nearly all of them contain so-called *giant connected components* (GCCs), which link together a majority of hosts (IP addresses) observed in the traffic, a phenomenon that has been observed in many social network studies. What is particularly interesting is the observation that TAGs associated with different applications exhibit *distinct* patterns and structures. These properties of application TAGs[1] are first observed and studied in [7], where the authors propose several graph-theoretical (average or distributional) metrics to help characterize and distinguish such graphs. While these metrics are useful in summarizing the overall statistical properties of the graphs, in general, they shed little light on how TAGs are formed and how they can be *meaningfully* interpreted.

In this chapter, we propose a novel *(statistical) graph decomposition* method based on *orthogonal nonnegative matrix tri-factorization* (tNMF) to analyze and extract the "core" host interaction patterns and other key structural properties of application TAGs. This technique is motivated by the observation that the matrix representations of application TAGs exhibit clear *block* structures, which suggest that they are composed of a number of *dense* sub-graphs representing "dominant" host groups (or "communities") with more intense interactions. In a sense, these dense subgraphs collectively form the "core" of the TAGs, capturing the most significant interactions among the dominant host groups. We formalize these observations and intuitions in the context of the proposed tNMF graph decomposition framework. More specifically, the tNMF method produces a *co-clustering* of the inside and outside hosts as well as a low-rank "core" matrix that represents the *overall interaction structure* among these groups and their *interaction intensities*. Each pair of inside and outside host groups with strong interactions corresponds to a *dense* (bipartite) subgraph in the original TAG and the bipartite (hyper)graph *induced* by the low-rank "core" matrix is referred to as the (core) *latent*

[1]In [7] where packet traces of relative short durations are used, these TAGs are referred to as *traffic dispersion graphs*.

hypergraph of the original TAG. In other words, the tNMF method *approximately decomposes* a TAG into a series of dense subgraph components and a (core) latent hypergraph representing inter-connection structures among the graph components.

Applying the tNMF method to various application TAGs (derived from our campus network datasets) such as HTTP, Email, DNS, P2P, online chat, and gaming applications, we characterize and classify the typical structures of the resulting graph components and (core) latent hypergraphs. Through extensive experimental analyses, we demonstrate that the decomposition results not only capture the dominant structures in the original TAG, but also are amenable to meaningful interpretations. For instance, HTTP TAGs are largely formed by a series of star-like or mesh-structured dense graph components that are inter-connected primarily due to hosts appearing in multiple inside/outside host groups, but sometimes also through one inside/outside host group interacting with multiple outside/inside groups. The chat traffic graphs are formed by a series of much less dense subgraphs inter-connected by an overall star-like structure. In contrast, the P2P traffic graphs show more diverse structures, reflecting the diversity and complexity of P2P applications. Using these components and their structural properties, we also study the evolution of TAGs over time. Moreover, we also provide two examples to illustrate the potential utility of our tNMF method in practical network management tasks such as unknown application identification and suspicious/anomalous traffic activity (or application) detection. In summary, our tNMF-based framework provides an *easy-to-understand*, *interpretable* and *quantifiable* means to analyze and characterize key structural properties of large, sparse, complex and richly connected TAGs that are otherwise hard to visualize and comprehend.

6.2 Traffic Activity Graphs

In this section, we introduce the *(bipartite) traffic activity graphs* (*TAGs*) defined in the context of the NetFlow data collected in our campus network and present some *visual* and *graph-theoretical* characteristics of such graphs. Further, using their matrix representations, we highlight the *block structures* inherent in the traffic activity graphs which motivate the statistical graph decomposition framework proposed in this chapter.

Datasets. The primary datasets used in our study are non-sampled, Cisco NetFlow records from/to our campus network (with three class-B or /16 address blocks) to/from the rest of the Internet, collected at our campus border router over a month period. We also have access to several (tier-1) ISP datasets which contain *sampled* NetFlow records collected at various routers inside the ISP networks (One of the ISP datasets is used in Sect. 6.6 in our study of the Storm worm activities, as an example to illustrate the utility of our proposed tNMF decomposition method). For ease of exposition, we will introduce the notion of traffic activity graphs and present the proposed tNMF decomposition method in the context of our campus network datasets. Nonetheless, we remark that the proposed methodology and associated concepts are equally applicable to ISP datasets. Further, the overall observations and insights gained from our campus network datasets also hold for ISP datasets, although the specific results and their interpretations may vary.

6.2.1 Traffic Activity Graphs and their Overall Characteristics

Using the campus network datasets, we introduce and define *application(-specific)* (or rather, *port-specific*) traffic activity graphs (TAGs) as follows. Given a set of *service ports* $\mathbb{P}$ associated with an application of interest (e.g., TCP ports 80 or 443 for Web applications[2]), let $\mathbb{F}$ be a collection of flows (observed during some time window) that use a port $p \in \mathbb{P}$ either in the source or destination port header field. The set of inside IP addresses (representing hosts inside our campus network, or inside hosts) and the set of outside IP addresses (outside hosts), which appear in $\mathbb{F}$ are denoted as $\mathbb{IH}$ and $\mathbb{OH}$, respectively (For the ISP datasets, we may refer to the set of subscribers as $\mathbb{IH}$ and the set of other Internet hosts as $\mathbb{OH}$. In addition, other application specific definition of $\mathbb{IH}$ and $\mathbb{OH}$ is also applicable, see Sect. 6.6). The ($\mathbb{P}$-specific) TAG $\mathbb{G} := \{\mathbb{V}, \mathbb{E}\}$ is a *bipartite* graph[3] with the vertex set $\mathbb{V}$ and edge set $\mathbb{E}$, where $\mathbb{V} = \mathbb{IH} \cup \mathbb{OH}$, and $e_{ij} \in \mathbb{E}$ if at least one flow from ih_i to oh_j exists in $\mathbb{F}$ (Depending on the purpose of analysis, a *weighted* version of such a graph can also be defined where the weight w_{ij} associated with the $e_{ij} \in \mathbb{E}$ represents, say, the number of flows from ih_i to oh_j in $\mathbb{F}$). We remark here that in addition to application/port-specific TAGs defined above, other types of TAGs can also be defined, e.g., using other criteria for filtering and selecting flows from the (original) NetFlow datasets. Clearly what types of TAGs should be defined and used will depend on the purpose of study. For instance, as an application of the tNMF method to anomaly detection, in Sect. 6.6 we introduce two different types of TAGs to study the "anomalous" Storm worm activities.

Traffic activity graphs capture the *network-wide* communication and *interaction* patterns between inside and outside hosts of a network. They are primarily driven by the user activities or behaviors, moderated in part by the inherent "application

[2]Depending on the applications and/or focus of the study, $\mathbb{P}$ may contain one or multiple ports. For instance, by considering TCP port 80 only, we focus on HTTP-only Web traffic, while including TCP port 443, we also include HTTPs traffic in the study. In some cases, ports in a given service port set $\mathbb{P}$ may be used by some "non-standard" applications other than the "well-known" application; e.g., port 80 may be used by some P2P applications to penetrate firewalls. Since the majority of flows using port 80 are generated by Web applications/HTTP protocols, for convenience we refer to the TAGs derived from flows with ports 80 and 443 as "HTTP" TAGs. The same remark applies to other similarly named TAGs such as "Email" TAGs. Note also that unless otherwise stated in this chapter, all ports refer to TCP ports.

[3]We note that, since the data used in our work is collected at the border router of a large campus network, we only observe traffic between the inside hosts and the outside hosts. Therefore, the corresponding TAGs are bipartite. However, our method can be readily extended to regular network activity graphs.

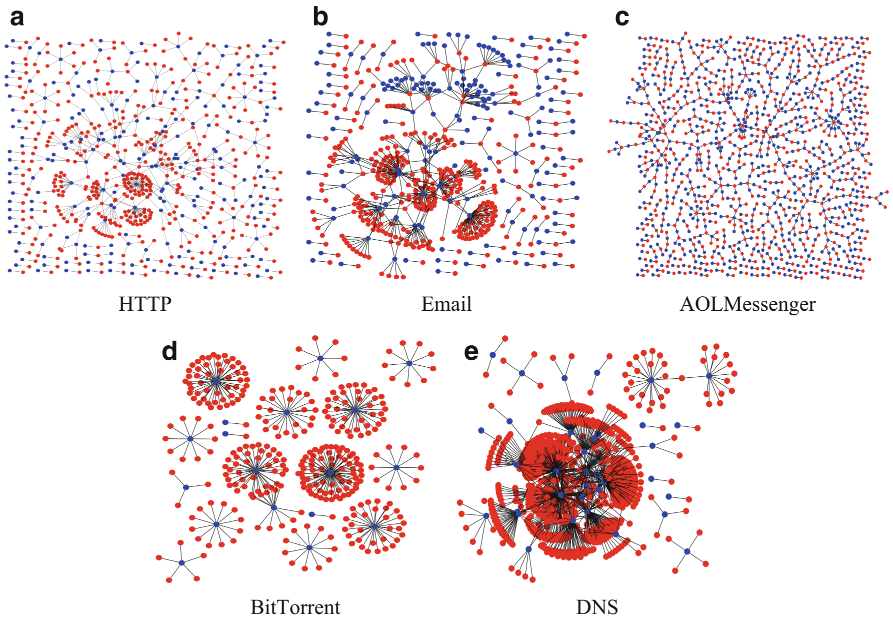


Fig. 6.1 Application TAGs with 1,000 flows: *blue* and *red* points denote inside and outside hosts, respectively

structures" which determine how users or clients interact with other users or servers. Hence, we would intuitively expect that different applications (e.g., client–server-based Web applications vs. P2P applications) may exhibit distinct graph structures or shapes. As first reported in [7], this is indeed the case. Using the Graphviz tool (node radius = 0.1, edge weight = 2.0), in Fig. 6.1 we present five representative application TAGs (derived from our campus network dataset that begins at 10:00 am on 02/17/2006): HTTP (port 80 or 443), Email (port 25 or 993), AOL messenger (port 5190), BitTorrent (port 6881) and DNS (port 53, UDP). For clarity of graphing, here we only consider outgoing flows where the inside hosts are service requesters (accessing the specific service ports) and outside hosts are service providers (opening the service ports for access). The inside hosts (service requesters) are represented by *blue* dots, while the outside hosts are represented by *red* dots – hence, the graphs are best viewed on a computer screen or a colored print-out. Figure 6.1 shows the five example application TAGs using the *first 1,000 flows*, while Fig. 6.2 using the *first 3,000 flows*.

Clearly, these application TAGs display distinct shapes or structures. For example, HTTP and Email traffic graphs contain a number of more richly connected star-structures (centered either at an outside or inside host), while BitTorrent traffic graph contains a few (apparently isolated) dense star-structures centered at inside hosts only. In contrast, such structures disappear in the AOL messenger traffic graph, where all nodes are characterized with low degrees. Comparing the corresponding

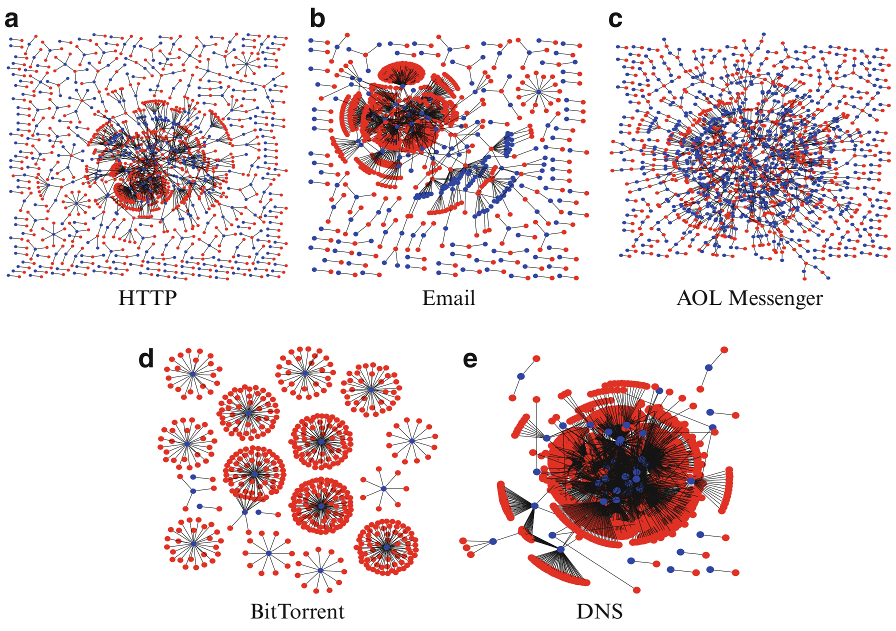


Fig. 6.2 Application TAGs with 3,000 flows: *blue* and *red* points denote inside and outside hosts, respectively

application tags in Figs. 6.1 and 6.2, we see that with more flows added to the graphs, the basic characteristics of these graphs appear to persist, with the core star-structures in the HTTP, Email, BitTorrent and DNS traffic graphs becoming denser. In all cases, some originally disconnected parts of the graphs start to connect and merge together – this phenomenon leads to the so-called *giant connected components* that we will discuss shortly.

Table 6.1 lists some key statistics[4] for a few selected application TAGs, each generated from a flowset of $|\mathbb{F}|$=10,000. More specifically, using the port(s) listed in the 2nd column of the table, we extract 10,000 *unique* flows containing the port(s) from the NetFlow dataset beginning at 10:00 am on 02/17/2006 to generate the corresponding flowset $\mathbb{F}$ for each application TAG. The approximate duration spanned by the flows in each flowset is listed in the 3rd column.[5] The 4th ($|\mathbb{IH}| \times |\mathbb{OH}|$) column shows the number of nodes of the resulting bipartite traffic

[4]Several other graph-theoretical metrics such as node degree distribution, joint degree distribution, depth, rich club connectivity, and so forth are used in [7] to characterize and distinguish various TAGs.

[5]Note that since flows with ports 80 and 443 comprise the majority of the traffic, we have 10,000 flows within a time period of roughly 2.7 min. Likewise, for the Email we have 10,000 flows within a time period of roughly 58 min, whereas for other applications, up to 2 h are needed to obtain 10,000 flows.

Table 6.1 Characteristics of different application TAGs using 10,000 unique flows

Type	Port(s)	Duration	$\lvert\mathbb{IH}\rvert \times \lvert\mathbb{OH}\rvert$	Density(10^{-3})	$\bar{d}(ih)$	$\bar{d}(oh)$
HTTP	80,443	2.7 min	1,193×3,054	2.73	8.38	3.27
Email	25,993	58 min	289×5,262	6.58	34.60	1.9
AOL Msgr	5190	2.1 h	2,047×1,221	4.00	4.89	8.19
BitTorrent	6881	57 min	84×8,610	13.83	103.76	1.01
DNS	53 (UDP)	2 min	57×5,634	31.14	175.44	1.77
eMule	4662	81 min	33×9,987	30.34	303.03	1
Gnutella	6346,6348	73 min	136×9,760	7.53	73.53	1.02
MSN Msgr	1863	2.3 h	1,603×712	8.76	6.24	14.04

Table 6.2 GCC properties of different application TAGs using 10,000 unique flows

Type	Port(s)	Duration	GCC(%)	GCC(inside×outside)	GCC(edges)
HTTP	80,443	2.7 min	87.5	961×2,756	9,660
Email	25,993	58 min	94.1	111×5,114	9,790
AOL Msgr	5190	2.1 h	99.4	2,039×1,209	9,988
BitTorrent	6881	57 min	89.6	35×7,751	9,013
DNS	53 (UDP)	2 min	99.7	50×5,626	9,992
eMule	4662	81 min	96.7	12×9,690	9,702
Gnutella	6346,6348	73 min	94.5	55×9,299	9,538
MSN Msgr	1863	2.3 h	92.2	1,562×572	9,856

graphs derived from the flowsets. The *density* values of the graphs, defined as $\lvert\mathbb{E}\rvert/(\lvert\mathbb{IH}\rvert \times \lvert\mathbb{OH}\rvert)$, are listed in the 5th column. We see that all of these graphs are extremely sparse. The next two columns list the average node degrees, $\bar{d}(ih)$ and $\bar{d}(oh)$, of the inside and outside hosts (IP addresses), respectively. A large $\bar{d}(ih)$ in general indicates the existence of popular (high-degree) inside hosts; P2P applications, such as BitTorrent, eMule and Gnutella, e.g., have higher $\bar{d}(ih)$ values. In contrast, online chat applications, such as MSN Messenger and AOL Messenger, show no dominance of any host.

The most interesting characteristic is perhaps the existence of the so-called *giant connected component (GCC)* (i.e., the *largest connected component* in the TAG) that connects a majority of nodes in a TAG, when the number of flows (or the observation time interval) becomes sufficiently large. As illustrated in Figs. 6.1 and 6.2, when the number of flows is increased from 1,000 to 3,000, the (core) connected region in the HTTP, Email and AOL Messenger TAGs expands to connect more nodes and generally grows denser. From the 4th column in Table 6.2, for each application TAG with 10,000 flows (i.e., edges), the GCC connects 87.5% of all the nodes in the HTTP TAG, 94.1% in the Email TAG, 99.4% in the AOL Messenger TAG, etc. Even for the BitTorrent TAG which appears to be comprised of a few disconnected star structures (see Fig. 6.2), the largest connected component eventually connects 89.6% of all the nodes. The last two columns in Table 6.2 list the size (numbers of inside/outside hosts and edges) of the GCC for each application TAG.

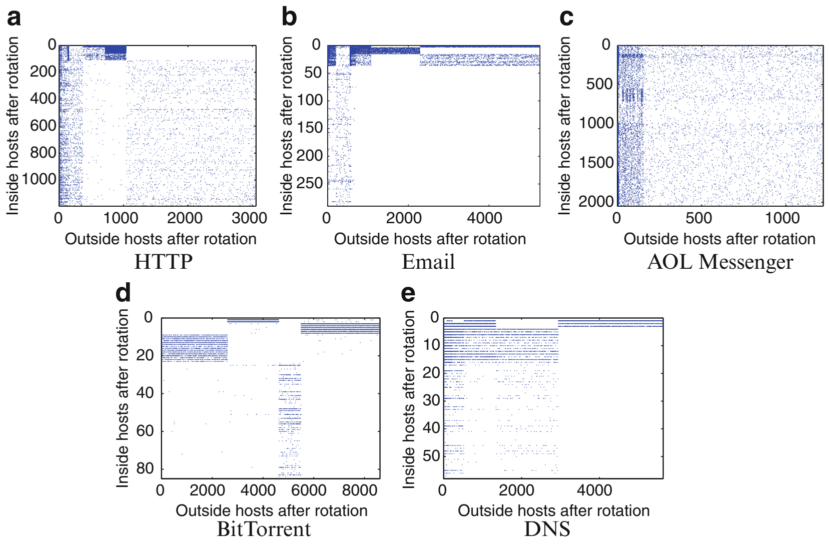


Fig. 6.3 Block structures after rotating rows and columns of adjacency matrices using 10,000 flows

6.2.2 Block Structures in TAGs

It is well known in random graph theory that for a fixed number of nodes, as the probability of (uniform) edge generation increases, a giant connected component emerges almost surely in such (uniformly generated) random graphs [2]. On the other hand, the application TAGs (and their resulting GCCs) show high diversity and variability (e.g., as manifested by their degree distributions), suggesting that their formation is not purely random. In fact, these graphs show a strong cluster effect, or contain "latent structures" underlying the applications TAGs. We show the existence of such structures by using the matrix representation of the TAGs.

Given a bipartite TAG $\mathbb{G}$, we construct its adjacency matrix $A = [a_{ij}]$, where the rows and columns of A correspond to the hosts in $\mathbb{IH}$ and $\mathbb{OH}$, and $a_{ij} := 1$ if $e_{ij} \in \mathbb{E}$. While we know that A is a very sparse matrix (see Table 6.1), we permute the rows and columns of A to show that there exist "dense" blocks or sub-matrices in A. Figure 6.3 presents the results for five example traffic graphs, where their corresponding adjacency matrices are displayed after we have selectively rotated their rows and columns. The block structures in the matrices are clearly visible, with certain areas far denser than others. The existence of dense vs. sparse blocks suggests that some groups of inside hosts tend to communicate or interact with certain groups of outside hosts, while rarely with other outside hosts. The block structures of A for different applications also show distinct patterns.

The existence of such latent structures is not surprising. Intuitively, they represent the underlying communication patterns or "social interactions" of users and services inside and outside our campus network. Such interactions lead to the formation of various "communities of interest" with distinct communication patterns or behaviors. For example, HTTP applications may provide different utilities, e.g., search engines, news services, blogs, photo or video sharing service, etc. Requesters who are looking for a specific utility will connect to the providers of such a utility, and due to the role of search engines and "social influence," they are also more likely to connect to a few other popular providers. The service requesters and providers thus together form a distinct community in the HTTP traffic graph. The dense block structures shown in Fig. 6.3 also provide hints as to why and how the GCCs are formed. We see that for some inside groups, sometimes a subset of the group members communicate with more than one outside group (or a sub-group therein) and vice versa, resulting in various dense components to be connected with varying degrees of connectivity.

The block structures suggest that despite their sparsity, the application TAGs are composed largely of connected, dense sub-graphs that are not formed randomly, but represent certain latent host interaction patterns, or shared interests of various user communities. These observations and insights motivate us to identify and extract these "dense subgraphs" and the inside/outside host groups associated with them, so as to better understand and characterize network traffic graphs. In the remainder of the chapter, we present a statistical graph decomposition technique based on *orthogonal nonnegative matrix tri-factorization*, referred to as *tNMF*, and discuss the experimental results.

6.3 Graph Decomposition Using tNMF

In this section, we present a statistical graph decomposition technique based on *orthogonal nonnegative matrix tri-factorization* (*tNMF*), and apply it to extract dominant "graph structure components" in an application TAG. Each such component is a *dense* bi-partite subgraph consisting of a pair of inside host group and outside host group that are more strongly connected than any host not part of the inside/outside group. The collection of these subgraph components (together with their inter-connection) constitutes, in a sense, the "core" of the application TAG, capturing the dominant communication patterns or interaction structures between the inside and outside hosts.

6.3.1 The tNMF Method

Given an application TAG $\mathbb{G}$ representing the interaction patterns of m inside and n outside hosts, let $A_{m\times n}$ be the corresponding (binary) adjacency matrix A defined earlier. The problem of extracting the strongly connected subgraphs from $\mathbb{G}$, or equivalently the "dense" sub-matrices in A, can be formulated as a *co-clustering* problem where inside hosts and outside hosts are *jointly* clustered into k groups and l groups, respectively (in general $k, l \ll \min(m,n)$). Each subgraph is now defined as a sub-matrix covered by a pair of inside/outside host groups, and the *dense* subgraphs can be identified from these $k \times l$ sub-matrices.

As illustrated in [5], this co-clustering problem can be formulated as an *orthogonal nonnegative matrix tri-factorization* (tNMF) problem: Given a nonnegative matrix $A_{m\times n}$, we factorize it (or more precisely, *approximately decompose* it) into three *low-rank nonnegative* matrices, $R_{m\times k}$, $H_{k\times l}$, and $C_{n\times l}$ so as to minimize the following objective function J subject to the orthogonality constraints on R and C:

$$\min_{R\geq 0, C\geq 0, H\geq 0} J(R,H,C) = ||A - RHC^T||_{\mathrm{F}}^2$$
$$s.t. R^T R = I \text{ and } C^T C = I, \tag{6.1}$$

where $||\cdot||_{\mathrm{F}}$ is the Frobenius norm, and $k, l \ll \min(m,n)$. The NMF problem and its solution were first introduced in [13, 14] as an alternative approach to singular value decomposition (SVD) and other matrix decomposition methods (see Sect. 6.7 for a brief discussion on these and other related methods), and has been successfully applied in various machine learning applications [10, 13, 27]. The *orthogonal* constraints for R and C distinguish tNMF from other NMF algorithms and enable it to simultaneously cluster the rows and columns of a matrix[17].

The solution to the tNMF problem employs an iterative optimization procedure. We first initialize R, C and H to contain only positive random entries, and we then keep updating R, C and H using the following updating rules until the change in the relative square error (RSE), $RSE := ||A - RHC^T||_{\mathrm{F}}^2 / ||A||_{\mathrm{F}}^2$, falls below a predefined threshold θ, say, $\theta = 10^{-7}$,

$$R_{ip} \leftarrow R_{ip} \frac{(ACH^T)_{ip}}{(RR^T ACH^T)_{ip}},$$
$$C_{jq} \leftarrow C_{jq} \frac{(A^T RH)_{jq}}{(CC^T A^T RH)_{jq}},$$
$$H_{pq} \leftarrow H_{pq} \frac{(R^T AC)_{pq}}{(R^T RHC^T C)_{pq}}. \tag{6.2}$$

It has been shown [5] that using the above updating rules, the RSE will monotonically decrease and converge to a local minimum. In general, it takes $O(t(k+l)mn)$ for the optimization procedure to converge, where t is the number of iterations. Therefore, in practice, sampling approaches can be used to reduce the size of the TAG before the decomposition method is applied. In Sect. 6.3.3, we will discuss several important practical issues in applying this tNMF method to decompose application TAGs and in improving the convergence rate of the proposed algorithm.

6.3.2 Interpretation of tNMF Results

In the context of decomposing TAGs, we propose a novel interpretation of the tNMF decomposition results as follows. The orthogonal low-rank, nonnegative matrices R and C divide the rows and columns into k inside and l outside host groups, where $R_{\cdot p}$, $p = 1, \dots, k$, and $C_{\cdot q}$, $q = 1, \dots, l$, serve, respectively, as the "membership indicator" functions of the row groups and column groups. Since entries in R and C are nonnegative real numbers, this naturally gives rise to a *soft* co-clustering of the inside and outside hosts: R_{ip} (after row normalization) can be viewed as the "likelihood" of inside host i belonging to inside host group p, and C_{jq} the "likelihood" of outside host j belonging to outside host group q. R and C can also be used to construct a *hard* co-clustering where each inside/outside host is assigned to *at most one* inside/outside host group. For simplicity of exposition, this is the interpretation we will adopt in the remainder of this chapter (although a soft clustering interpretation can also be used, which we leave as future work). In the following, we show how the hard clustering is constructed.

Using R and C we define the inside/outside host group membership indicator matrices $\hat{R}$ and $\hat{C}$ as follows: $\hat{R}_{ip} = 1$ if $p = arg\max_j \{R_{ij} : R_{ij} > 0\}$, and 0 otherwise. In other words, we assign an inside host i to the inside host group p associated with the largest (nonzero) R_{ip} value. In particular, if all R_{ip}'s are zero, then host i is not assigned to any inside host group. In addition, when multiple groups are associated with the largest value, we randomly assign the host to one of these groups to resolve ties. The indicator matrix $\hat{C}$ is defined similarly. With $\hat{R}$ and $\hat{C}$ thus defined, we use IG_p to denote the pth inside host group, and OG_q the qth outside host group. Further, let $\mathbb{I}G := \{IG_1, \dots, IG_k\}$ and $\mathbb{O}G := \{OG_1, \dots, OG_l\}$ denote the collection of these inside and outside host groups.

We now introduce the *group density* matrix $\hat{H} = \{\hat{H}_{pq}\}$ where

$$\hat{H}_{pq} := \frac{(\hat{R}^T A \hat{C})_{pq}}{||\hat{R}_{\cdot p}||_1 \cdot ||\hat{C}_{\cdot q}||_1}, 1 \le p \le k,\ 1 \le q \le l, \tag{6.3}$$

and $||\cdot||_1$ is the L_1-norm. For a given bipartite subgraph corresponding to host groups IG_p and OG_q, the denominator in (6.3) represents the maximum number of possible edges between IG_p and OG_q assuming the subgraph is complete; while the numerator stands for the actual number of edges within this subgraph. In this way, we see that $\hat{H}_{pq}$ is the density of the (bi-partite) subgraph representing the interaction patterns between the members in IG_p and OG_q, where $0 \le \hat{H}_{pq} \le 1$. A large $\hat{H}_{pq}$ value (e.g., close to 1) indicates a strongly connected bipartite subgraph (i.e., the subgraph is nearly a complete graph), while a small or zero $\hat{H}_{pq}$ value suggests that only a few edges exist between some members of these two groups, or no edge at all. Using $\hat{H}$ we can identify and extract "dense" bi-partite subgraphs in the TAG $\mathbb{G}$. Formally, we say a bipartite subgraph $\mathbb{S}_{pq}$ in $\mathbb{G}$, where $\mathbb{S}_{pq} = \{[a_{ij}] | a_{ij} \in A, \hat{R}_{ip} = 1, \hat{C}_{jq} = 1, 1 \le i \le m, 1 \le j \le n\}$, is *dense* if $\hat{H}_{pq} \ge \delta$, for some appropriately chosen *density threshold* $\delta \in (0,1]$, say, $\delta = 0.5$. These dense subgraphs in $\mathbb{G}$ thus represent dominant communication patterns among the inside/outside hosts with strong interaction structures. We refer to these dense subgraphs, $\mathbb{S}_{pq}$'s, of $\mathbb{G}$ as the *significant graph components*, or simply *graph components* of $\mathbb{G}$. We will analyze their structures and interpret their meanings in the context of various application TAGs in Sect. 6.4.

Furthermore, the group density matrix $\hat{H}$ induces a (weighted) bi-partite (hyper)graph $\mathbb{H} := \{\mathbb{IG} \cup \mathbb{OG}, \mathbb{E}_{\hat{H}}\}$, where the nodes represent the inside and outside groups, IG_p and OG_q, $1 \le p \le k$ and $1 \le q \le l$, and an edge $e_{pq} \in \mathbb{E}_{\hat{H}}$ if $\hat{H}_{pq} > 0$, and the weight associated with edge e_{pq} is exactly $\hat{H}_{pq}$. More generally, we can also define an unweighted (hyper)graph $\mathbb{H}_\delta$ where an edge $e_{pq} \in \mathbb{E}_{\hat{H}}$ *if and only if* $\hat{H}_{pq} > \delta$, i.e., if the density of the corresponding subgraph $\mathbb{S}_{pq}$ is at least δ. Hence, the induced hypergraph $\mathbb{H}$ represents the interaction patterns and their intensities between the various inside/outside host groups, and $\mathbb{H}_\delta$ captures the dominant interactions among the core host groups (or communities) of the inside/outside hosts. It is in this sense that we refer to the hypergraph $\mathbb{H}$ ($\mathbb{H}_\delta$) as the *(core) latent hypergraph* underlying (or generating) the original TAG $\mathbb{G}$. In particular, using $\mathbb{H}_\delta$ and the corresponding dense graph components $\mathbb{S}_{pq}$'s, we obtain an *approximate* "core" $\hat{\mathbb{G}}$ of the original $\mathbb{G}$ that captures the dominant interaction patterns among significant inside/outside host groups.

6.3.3 *Practical Issues*

We briefly discuss several key practical issues in applying the tNMF method to the (statistical) decomposition of application TAGs, and highlight the solutions we employ to: i) select the rank k and l and the density threshold δ, and ii) improve the convergence rate and avoid local minima.

6.3.3.1 Selection of Rank and Density

Without loss of generality, we set $k = l$, which is an input parameter for the tNMF algorithm, and specifies an upper bound on the desired or expected groups formed by inside and outside hosts.[6] As discussed earlier, the density threshold δ is a parameter for identifying dense subgraphs. Since the selection of appropriate k and δ depends on specific applications, in the context of TAG decomposition, our criteria for choosing k and δ are twofold: to obtain *stable* graph components which contain sufficient number of edges in the original TAG, i.e., with a good *edge coverage*.

[6]In this section, we assume $k = l$ for ease of exposition. In our experiment, k can be different from l depending on the types of TAGs under study. We always search for the best k and l values using the KS test introduced in this section.

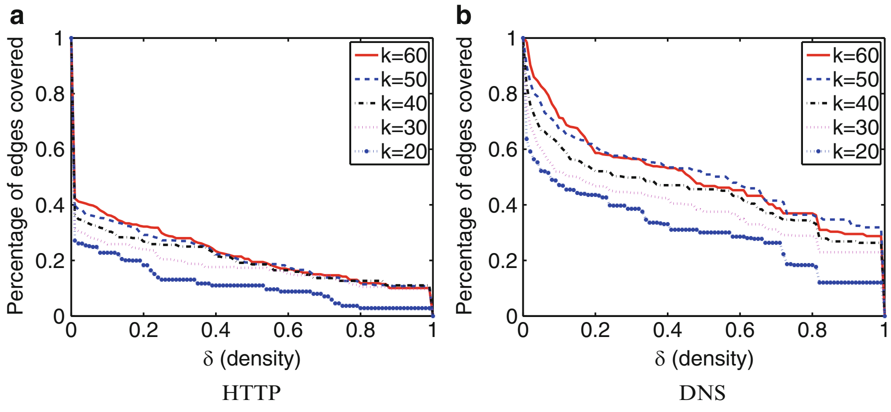


Fig. 6.4 Edge coverage for various k and δ

We first study the edge coverage of extracted graph components by varying k and δ. Figure 6.4a, b show the edge coverage for HTTP and DNS graphs with different k and δ, respectively. In general, either increasing k or reducing δ will result in an increase of the edge coverage. However, we observe the number of covered edges converging when $\delta > 0.5$ and k exceeds 40 for HTTP and above 50 for DNS. This implies that when k exceeds a certain threshold, the "dense" subgraphs (with $\delta > 0.5$) become stable (similar observations are made in terms of other TAGs), hence we choose $\delta = 0.5$ in all the experiments for identifying significant subgraphs. We then apply a linear search of k's starting at 20 and with a small increment at a time. We note that different increments often lead to similar results since the dense subgraphs become persistent with a sufficiently large k (In case of $k \neq l$, we may search for the appropriate parameters by increasing k and l iteratively). To balance the accuracy and efficiency, we choose an increment of five in our experiments. We then use the two-way Kolmogorov–Smirnov goodness-of-fit test to compare edge coverage curves between consecutive k's. The null hypothesis is that two edge coverage curves are identical. We choose rank k if the KS test fails to reject the null hypothesis (i.e., the P-value is above 0.05). For example, we choose $k = 40$ for the HTTP TAG as in Fig. 6.4a, since the P-value equals 0.0215 between $k = 35$ and $k = 40$, but the P-value becomes 0.3499 between $k = 40$ and $k = 45$. As another example, we choose $k = 45$ for the DNS graph (Fig. 6.4b) due to the P-value of 0.0001 between $k = 40$ and $k = 45$, and P-value of 0.0715 between $k = 45$ and $k = 50$.

So far, we have shown how to select k and δ to produce a good edge coverage of the original TAG. Now we investigate whether these edges lead to stable graph components. Let $\mathbb{E}_i$ and $\mathbb{E}_{i-1}$ be the extracted edge sets corresponding to k_i and k_{i-1}, respectively. We define the edge similarity as $(\mathbb{E}_i \cap \mathbb{E}_{i-1})/(\mathbb{E}_i \cup \mathbb{E}_{i-1})$. Fixing $\delta = 0.5$, we increase k by 5 at a time ($k_i = k_{i-1} + 5$). Figure 6.5a shows the edge similarities (y-axis) associated with different k's (x-axis) for the HTTP and

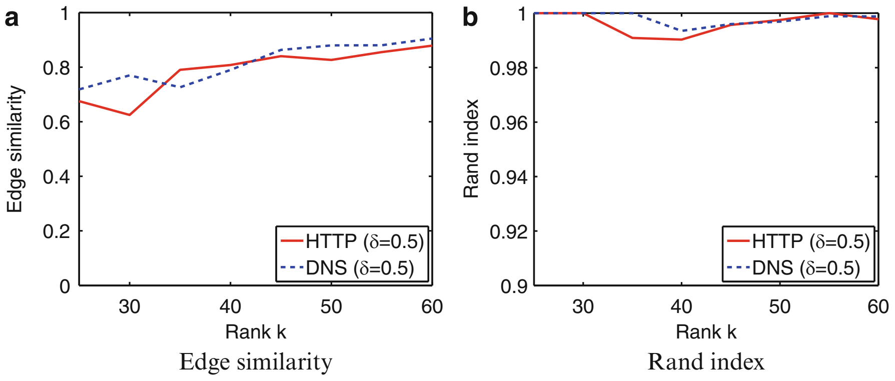


Fig. 6.5 Similarity of graph components

DNS graphs, respectively. We observe that the edge sets become more similar as k increases. For the chosen k ($k = 40$ for HTTP and $k = 45$ for DNS), the edge similarity is close to 85%. This again implies that as the k is sufficiently large, the extracted dense subgraphs become stable.

To evaluate the similarity of graph components formed by these extracted edge sets, we use Rand index as a measurement. Due to the high similarity of the edge sets, we compute the Rand index using the edges in $\mathbb{E}_i \cap \mathbb{E}_{i-1}$. Let $\mathbb{C}_i$ and $\mathbb{C}_{i-1}$ be the sets of components associated with edge set $\mathbb{E}_i$ and $\mathbb{E}_{i-1}$. The Rand index is defined as:

$$\mathrm{Rand}(\mathbb{C}_i, \mathbb{C}_{i-1}) := (a+b) / \binom{n}{2}$$

where a and b represent the number of edge pairs that are in the same or different cluster in $\mathbb{C}_i$ that are also in the same or different cluster in $\mathbb{C}_{i-1}$. The denominator is the total number of edge pairs and $n = |\mathbb{E}_i \cap \mathbb{E}_{i-1}|$. The Rand index ranges from 0 to 1, with higher value indicating more similar clustering results. Figure 6.5b displays the Rand index corresponding to different k's for the HTTP and DNS graphs. The Rand index values are always close to 1, implying persistent dense clusters or subgraphs formed by these extracted edges for various k's.

6.3.3.2 Low Convergence Rate and Local Minima

Though the optimal solution of tNMF is unique, the random initialization of R, C and H in the basic tNMF optimization algorithm usually lead to both a low convergence rate and an unsatisfactory local minima solution. In addition, since the complexity of tNMF algorithm is $O(mnr)$, where m-by-n is the matrix size and r is the total number of iterations until convergence. Hence, a low convergence rate results in a higher complexity of the algorithm. In this chapter, we address this problem by employing a singular value decomposition (SVD) based initialization approach. The basic idea is to first apply the rank-k singular value decomposition

(SVD) on A for spectral co-clustering [3]. We then project the rows onto the resulting k-dimensional subspace spanned by the top k (rows or columns) principal components, and perform k-mean clustering to obtain an initial clustering of inside/outside host clusters. We initialize R by perturbing the inside host cluster membership vectors to obtain an all-positive matrix, namely, setting $R := R + \varepsilon$, where ε is a small positive constant to avoid zero entries. The initialization of C is done similarly. H is initialized with $R^{-1}AC^{-1^T}$, where R^{-1} and C^{-1} stand for the pseudo-inverse of R and C.

Through extensive experiment analysis, we find that our SVD-based initialization method not only improves the convergence rate significantly, but also enables our algorithm to find the "best" optimization solution. Using Email TAG as an example (with 100 experiments), the RSE using the SVD-based initialization is 0.34 ± 0.009, in comparison to 0.39 ± 0.027 using random matrix initialization. In addition, using the SVD-based initialization, the number of iterations to reach convergence is only 170.10 ± 71.09, while for the random matrix initialization is 323.57 ± 134.16. Hence, our SVD-based initialization method not only increases the approximation accuracy, but also enhances the speed of convergence.

6.4 Results and Interpretations

We apply the tNMF method to various application TAGs derived from our NetFlow datasets to extract their core latent hypergraphs and associated significant graph components. In this section, we analyze the structures of these graph components and interpret their meanings. We also investigate how these graph components are connected to form the core latent hypergraphs. These decomposition results provide a meaningful and quantifiable way to understand, analyze and distinguish the structures of large traffic graphs that are otherwise hard to visualize and comprehend. In particular, it also sheds light on how the giant connected components (GCCs) of these graphs may be formed.

6.4.1 Graph Component Structures

Applying the tNMF method to the application TAGs listed in Table 6.1, the sizes of the resulting (significant) graph components of each TAG are shown in Fig. 6.6. In the figures, each point (x, y) represents a single graph component, where x is the number of inside hosts in each inside host group, and y is the number of outside hosts in each outside host group. The locations of points lead us to define three basic types of graph component structures. We refer to the graph components corresponding to the points on the line $x = 1$ (i.e., the inside host group containing only one inside host) as having an *in-star* structure, i.e., a star structure centered at an inside host.

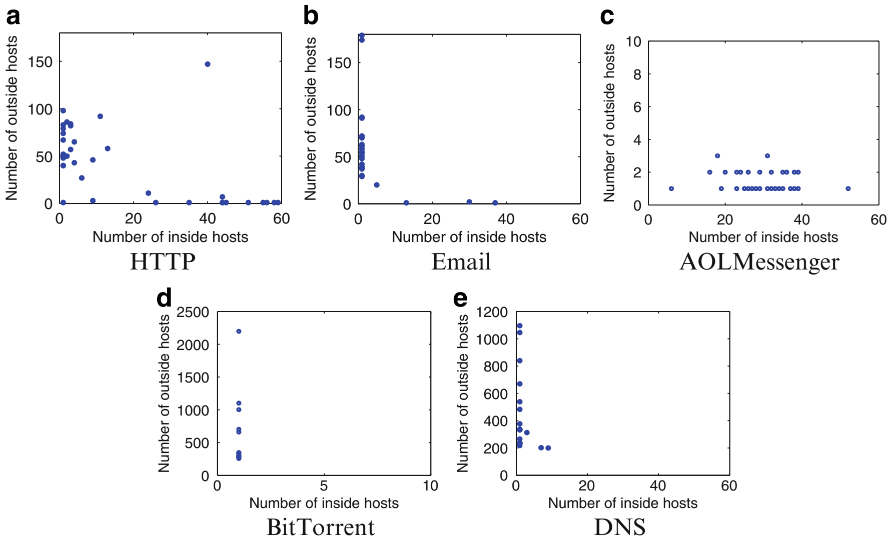


Fig. 6.6 Sizes of significant graph components for five example application TAGs ($\delta = 0.5$)

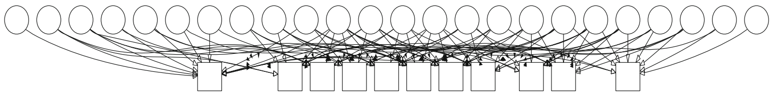


Fig. 6.7 A bi-mesh structure from the HTTP TAG

Similarly, we refer to the graph components corresponding to the points on the line $y = 1$ (i.e., the outside group containing only one outside host) as having an *out-star* structure, i.e., a star structure centered at an outside host.

The remaining points correspond to graph components which have at least two members in both its inside and outside host groups. We refer to them as having a *bi-mesh* structure, which represents fairly complex interactions or connectivity between the inside and outside hosts. An example of a bi-mesh structure is shown in Fig. 6.7. This bi-mesh structure consists of 24 inside hosts and 11 outside hosts, where inside hosts are represented by circles and outside hosts are denoted by squares. It contains more than 140 edges and most of the inside and outside hosts in their respective groups also have relatively high degrees, suggesting strong interactions between the members of these two inside/outside groups.

Table 6.3 summarizes the number of these three graph component structures for each of the TAGs. We see that different application TAGs exhibit great diversity in their graph component structures. For example, HTTP TAG contains a large number of bi-mesh structures and a few star structures. These bi-mesh structures may consist of hundreds of hosts as shown in the decomposition results of the HTTP TAG (Fig. 6.6a). In comparison, instant messaging application TAGs such as AOL messenger contain many out-star structures as well as a large number of

Table 6.3 Graph components of various TAGs.

Type	$\|instar\|$	$\|outstar\|$	$\|bimesh\|$	Edge coverage
HTTP	10	9	15	19.2%
Email	23	2	2	23.5%
AOL Msgr.	0	27	17	17.7%
BitTorrent	12	0	0	77.9%
DNS	11	0	3	56.3%

relatively small bi-mesh structures, which usually consist of two outside hosts and 20–40 inside hosts. On the other hand, P2P application TAGs contain mostly in-star structures; the richness in connectivity of these TAGs seem to manifest in the "interconnections" among the graph components, not within, unlike HTTP TAGs (see Sect. 6.4.3). The last column in Table 6.3 shows the percentage of edges covered by these extracted graph components in each TAG.

It is interesting to note that while TAGs are associated with vastly different applications (e.g., HTTP vs. P2P), most TAGs associated with the same or similar applications (e.g., various on-line chat applications) show very similar patterns. This observation suggests that the graph component structures capture the distinct characteristics of underlying application structures that determine how inside/outside hosts interact with each other.

6.4.2 Graph Component Interpretations

Using the IP addresses, their DNS names (if known) and other exogenous information sources (e.g., information on server Web sites), we have done extensive investigation to interpret and validate various graph components, namely, the inside/outside groups and their interactions. Recall that the TAGs in question are derived from outgoing flows originating from inside hosts of our campus network to outside hosts, where the inside hosts are "service requesters" while the outside hosts are "service providers." In other words, the port(s) used in identifying an "application" appear as destination ports in the flow records only.

6.4.2.1 HTTP

Due to the dominant volume of HTTP traffic and many activities associated with it, we observe a great variety of HTTP graph components representing different types of HTTP interactions. In HTTP TAGs, the in-stars and out-stars together account for 60% of all the components. The majority of the out-star structures are rooted at popular servers belonging to *Google*, *Yahoo*, etc., and the remaining are rooted at IP addresses belonging to CDN servers like *Akamai* or advertising sites like *DoubleClick*. Different from the out-star structures, the in-star structures tend to be rooted at IP addresses of NAT boxes, proxy servers and wireless access points.

In comparison to the star structures, the bi-mesh structures depict more sophisticated interactions between groups of service requesters and service providers. We are particularly interested in understanding the correlation of service providers that attract clients to access them simultaneously. Based on DNS names and other auxiliary information, we categorize various bi-mesh structures of HTTP TAGs derived from flow datasets at different times, and present their interpretations in Table 6.4. Because we rely heavily on external information such as DNS names, providing interpretation for all bi-mesh structures is not always achievable. In fact, we are able to explain 86.6% bi-mesh structures observed in an entire day. From Table 6.4, we conclude that the majority of bi-mesh structures in HTTP TAGs are formed due to three major reasons.

The first reason is the server farm effect (row 1), where servers belonging to a large Web site balance the workload by serving requests in turn or by only responding to requests for specific content. In this way, a client may establish connections with multiple server machines to complete one access to the Web site, and bi-mesh structures are formed by clients incidentally sharing several servers. The second reason is correlated service providers. For example, Web sites often collaborate with CDN providers for fast content delivery (row 2), e.g., *Yahoo* with *Akamai*, and *Facebook* with *Limelight Networks*. As another example, Web sites correlate with advertisement delivery networks (row 3) like *DoubleClick* and *Advertising.com*. In both cases, when clients connect to a particular Web site, they will be redirected to hosts in CDN network for further service or they will retrieve ad contents from advertisement delivery networks automatically. Such server sharing behavior also leads to bi-mesh structures.

In both of the above cases, the formation of bi-mesh structures is determined by HTTP servers. However, the shared interest of clients is the third major cause of bi-mesh structures. For example, in Table 6.4 row 4 to row 10, we observe interest groups related to news, media and photo sharing, shopping and online social networks. This suggests that clients tend to access Web sites delivering similar types of content.

6.4.2.2 Email and DNS

We find that the Email TAG is decomposed mostly into in-stars corresponding to Email servers of the university or several "big" departments (e.g., CS, IT, Math, Medical School within our campus). The out-stars are mainly rooted at popular Email servers such as *Gmail*. In Fig. 6.3b, one bi-mesh structure is caused by server relays for load balancing as in the HTTP case. The other interesting bi-mesh structure consists of inside Email servers belonging to some research labs and smaller departments, which talk to Email servers of a few academic institutions, and mail relays (some in Asia). We check them against DNS based blacklists, and find that two of the addresses are blacklisted and quite a few others belong to domains that no longer exist. Hence, we suspect that this bi-mesh may be formed due to Email spams. The DNS TAG looks similar to the Email graph, with a large number

Table 6.4 HTTP bi-mesh structure categorization

Category description	Examples	Pct.
Server farms	Lycos, Yahoo, Google	13.0
Content delivery networks	LLNW, Akamai, SAVVIS, Level3	24.5
Advertising providers	DoubleClick, Advertising.com TribalFusion	14.8
News related	WashingtonPost, New York Times, Cnet	1.6
Media and photo sharing	ImageShack, tl4s2.com, casalmedia.com	4.9
Broadcasters of news and music	MusicMatch, Napster, BBC	3.3
Job-search related	monster.com, careersite.com	1.6
Online shopping related	Ebay, CostCo, Walmart	3.3
Social network related	FaceBook, MySpace	18.0
Blogs and review sharing	LiveJournal, xpc-mii.net	1.6
Unknown	-	13.4

of in-stars and out-stars rooted at DNS servers. There are three bi-meshes, where the inside hosts consist of at least one DNS server along with a few Email servers (including our CS mail servers). These Email servers appear to be configured either to serve also as DNS servers, or perhaps more plausibly, to perform queries to outside DNS servers for reverse DNS look-ups to filter non-registered spam Email servers (as in the case of our CS Email servers).

6.4.2.3 Instant Messaging and P2P Applications

The TAGs associated with Microsoft, Yahoo and AOL messengers have similar structures which are distinct from those of HTTP and Email. They are decomposed into mostly small-size bi-meshes with many cross-connections between them, indicating that members of inside groups communicate with members of multiple outside groups. All hosts in the outside groups are associated with the same (top two-level) domain name, meaning that these small-size bi-meshes are indeed the effect of server relays. In contrast, P2P applications such as BitTorrent, eMule and Gnutella contain a majority of in-star structures. A few of them are somewhat loosely connected, indicating that the inside hosts at which the in-stars are centered also happen to share a number of destination hosts.

6.4.3 Latent Hypergraphs and GCC Formation

The group density matrix $\hat{H}$ and the induced core latent hypergraph $\mathbb{H}$ capture the (dominant) interactions among various inside/outside host groups, and shed

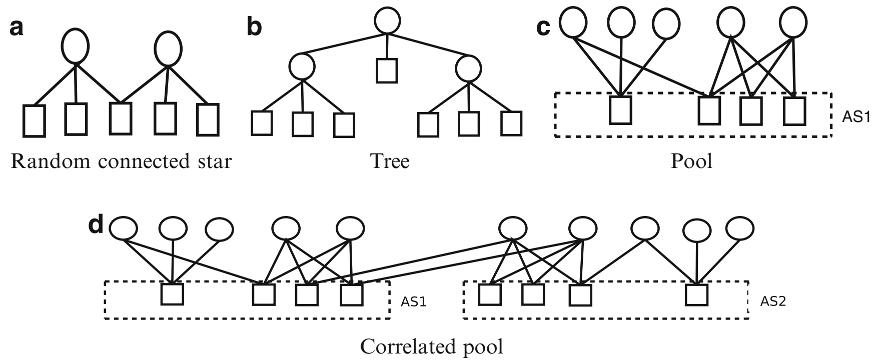


Fig. 6.8 Schematic depiction of typical latent hypergraph formation

light on the formation of giant connected components of various application TAGs. Through detailed analysis of the latent hypergraphs (and the *reconstructed core graphs*) of various application TAGs using our campus flow datasets (as well as the ISP flow datasets), we find four (inter-connection) structures that are most prevalent in the latent hypergraph structures. Most latent hypergraphs (and the resulting reconstructed core graphs) are formed predominantly using one or two of such structures. In Fig. 6.8a–d, we provide a *schematic depiction* of these four typical structures that inter-connect two graph components. In the figure, we use circles and boxes to represent respectively inside hosts (service requesters or clients) and outside hosts (service providers or servers). An edge between two hosts indicates interactions between them (i.e., with observed flows between them). In the following, we provide some plausible interpretations of these four structures.

- *Randomly Connected Star Structure*, where a hypergraph is formed by various high degree in-stars rooted at inside hosts/clients randomly sharing leaf nodes (outside hosts/servers). P2P TAGs usually fall into this category.
- *Tree Structure*, where some star roots behave like clients to connect to other stars. This structure shows up typically in IRC, Email, and DNS TAGs. We note that this structure does not appear in the hypergraphs of the application TAGs derived from our campus flow datasets, due to *limited visibility* (namely, we cannot observe the interactions among outside hosts); however, they do appear in application TAGs such as IRC, Email, and DNS TAGs obtained from the ISP flow datasets. For sake of completeness, we include this structure here.
- *Pool Structure*, where bi-meshs and out-stars are connected by a large number of low degree inside hosts/clients randomly communicating with the outside hosts/servers within these components. In addition, all the outside hosts/servers within these components share either the same (top two-level) domain name, or if their addresses are not DNS-resolvable, are associated with the same organization (based on the autonomous system number (ASN) using BGP routing

data look-up). This seems to suggest that the pool structure is likely caused by server relays. Many application TAGs such as messengers and online games contain this type of structure.

- *Correlated Pool Structure*, where multiple pool structures are connected by multiple inside host/clients communicating with a number of outside hosts/servers in different pools. HTTP TAGs are a typical example of this category. For example, CDNs or on-line ad service networks form multiple pool structures, as they provide service to a large number of (sometimes unrelated) Web sites. Inside hosts/clients belonging to different groups that are accessing these Web sites will also access the corresponding CDNs or ad service networks, thus interconnecting multiple graph components.

6.4.3.1 Overall Summary

By decomposing various applications TAGs into significant graph components and core latent hypergraphs that are more amenable to analysis and interpretation, our tNMF-based TAG decomposition method provides a powerful and valuable tool to characterize, classify and understand the *structural* properties of large, sparse, yet richly connected TAGs that otherwise seem too complex to comprehend. Further, it reveals the underlying application structures that determine how users access services or interact with each other, and sheds light on the formation of such large and complex graphs. Based on our analysis of "canonical" graph components and latent hypergraph structures, we summarize that: (1) For certain application TAGs such as Email, DNS and certain P2P applications, their main structures and rich connectivity can be decomposed into and captured by graph component structures with a diversity of "local" structures, some are simple (e.g., in-/out-star structures), others are complex (e.g., bi-mesh structures), which are inter-connected with relatively simple "global" structure, e.g., random star or tree structures; (2) However, for other applications such as online chat and game applications, the main structures and rich connectivity may be decomposed into and represented by the significant interactions (e.g., pool structures) of relatively simple graph component structures (e.g., mostly star-structures). For yet other applications (such as HTTP or Web), it is a combination of both components that contribute to their main structures and rich connectivity, for example, bi-mesh star structured graph components are inter-connected via correlated pool structures to form larger and more complex clusters, which may be then interconnected through random star structures. The structural complexity of HTTP or Web TAGs may not be too surprising in light that Web applications have evolved from the early simple client/server structure to today's "Web 2.0" driven by CDNs, search engines, on-line ad services, and Web services based on data centers and service-oriented architecture, forming a truly complex, dynamic, and inter-weaving Web.

6.5 Evolution of TAGs

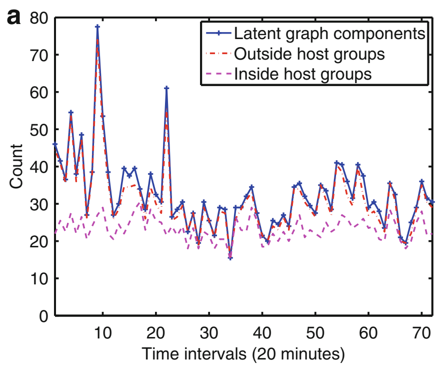
In the previous sections, we have studied the structural properties of various application TAGs as *static* objects: they are constructed by considering flows associated with an application of interest accumulated during a certain time window. Clearly, traffic activities are dynamic; hence the resulting TAGs evolve over time. In this section we investigate the *temporal* properties of TAGs. Due to their prevalence and the rich structures, we use HTTP TAGs as an example to study the evolution of TAGs over time.

6.5.1 *Metrics for Similarity Comparison*

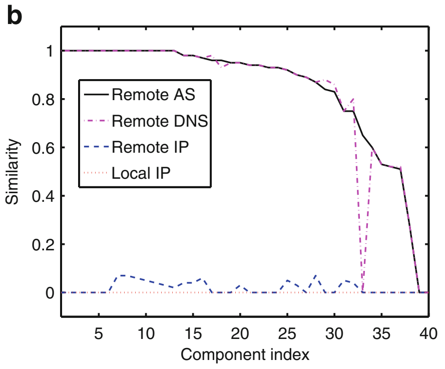
We take a one-day NetFlow dataset of our campus network and partition it into 20-min intervals (72 in total). We construct a sequence of HTTP TAGs, $\mathbb{G}_t$, $1 \leq t \leq 72$, using the first 10,000 unique (outgoing) HTTP flows (with destination ports 80 or 443) observed during each time interval, and study their evolution over time. Intuitively, we would expect that for inside/outside host groups that represent dominant and frequent interaction patterns, while their individual members (in particular, inside hosts or clients) are likely to fluctuate and vary over time, the corresponding graph components as well as the core latent graph structures should stay fairly stable most of time. To compare the decomposition results (thereby the inside/outside host groups and their interaction structures) derived from HTTP TAGs over time, we first introduce a few metrics.

Let $\mathbb{C}_s = \{C_i^s\}$ and $\mathbb{C}_t = \{C_j^t\}$ be the sets of significant graph components extracted from TAGs $\mathbb{G}_s$ and $\mathbb{G}_t$, $s < t$. Due to the dynamics of traffic activities and evolution of inherent "host community" structures (e.g., new members joining and old members leaving) as well as the artifact of the decomposition results, there is not necessarily a one-to-one correspondence between graph components in $\mathbb{C}_s$ and $\mathbb{C}_t$. For example, a graph component in $\mathbb{C}_s$ may be split or merged with other components in $\mathbb{C}_t$. In order to track the change of a particular graph component $C_i^s \in \mathbb{C}_s$ from time s to time t, we need to identify its (most likely) counterpart $C_j^t \in \mathbb{C}_t$. We adopt a simple "best-effort" matching algorithm as follows. Let $sim(C_i^s, C_j^t)$ denote an appropriately defined *similarity* metric for comparing components C_i^s and C_j^t at time intervals s and t. For each C_p^s, we say C_q^t is its counterpart (i.e., its "best match") if $q = arg\max_j sim(C_p^s, C_j^t)$ and $sim(C_p^s, C_q^t) \geq \eta$, where $\eta > 0$ is a pre-defined similarity threshold. If no such C_q^t is found, then C_p^s has no best match or counterpart at time interval t.

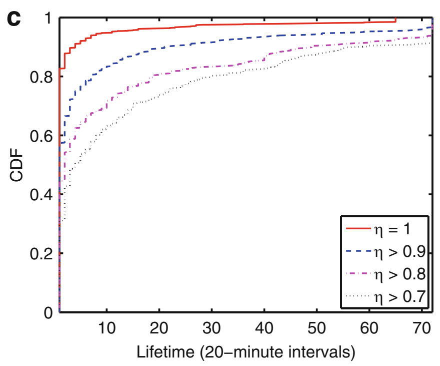
We introduce three similarity metrics to capture various relationships between two graph components (or their corresponding inside/outside host groups) over time. The *host-level similarity* (sim_h) is defined as the percentage of (inside or outside) hosts that C_i^s and C_j^t share in common, i.e., $\text{sim}_h(C_i^s, C_j^t) := |C_i^s \cap C_j^t| / |C_i^s \cup C_j^t|$. The *domain similarity* (sim_d) is defined as the percentage of hosts in two components



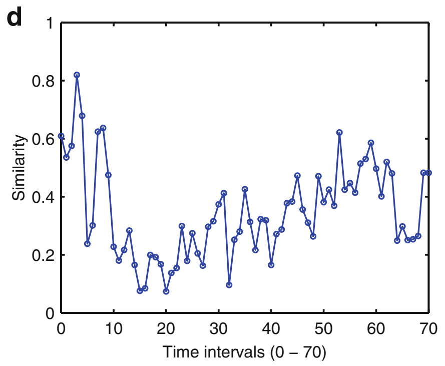
Number of latent components at different time intervals



Similarity of latent components at 6:40 pm and 7:00 pm



Lifetime of latent components



Average domain similarities at different times

Fig. 6.9 Evolution of HTTP TAGs

that share the same domain name suffix (the top 3-level domain names for an address ending with country code and top 2-level domain names otherwise). Likewise, the *AS similarity* (sim_{as}) is defined as the percentage of hosts in two components that belong to the same AS. All three similarity metrics range between 0 and 1, with 1 indicating exactly identical components at the corresponding similarity level. Obviously, all inside hosts (local IPs) have the same domain name suffix and belong to the same AS owned by the university. Hence, the last two similarity metrics are only useful in quantifying the similarity of outside host (remote IP) groups. However, the host-level similarity metric can be applied to both the inside and outside host groups associated with various graph components, yielding two similarity measures, one for the inside host (local IP) groups and one for the outside host (remote IP) groups.

Applying the tNMF decomposition to the HTTP TAGs $\mathbb{G}_t$, $1 \leq t \leq 72$, we obtain the graph components of each TAG. Figure 6.9a displays the number of resulting graph components as well as the number of associated inside/outside host groups as a function of time, where $t = 0$ corresponds to the first 20 min in the 0th hour of the day. The figure shows that the number of graph components fluctuates and evolves over time. In particular, compared to business hours, the number of components during the wee hours of the morning tends to fluctuate more widely and are thus less stable. Using the similarity metrics defined above, we apply the best-effort matching algorithm to the graph components of two consecutive HTTP TAGs at 6:40 pm and at 7:00 pm, and find their best matches (for the purpose of exposition, we set $\eta = 0$). Figure 6.9b shows the similarity scores of the graph components and their best matches, where the curves labeled "local IP" and "remote IP" are the host-level similarity scores of inside and outside host groups, respectively, while those labeled "remote DNS" and "remote AS" are the domain and AS similarity scores of outside host groups.

From Fig. 6.9b we see that the membership (generally clients) of inside host groups typically change frequently over time (all host similarities are equal to 0), which is not surprising. Outside hosts exhibit similar variability as that of inside hosts. However, because some of these outside hosts represent gateways (such as *Facebook* and *Myspace*), their similarity values are not as low as that of the inside hosts. Further, we notice that for most outside host groups, the new outside hosts tend to belong to the same domain or AS, indicating they provide the same/similar services or function in similar roles as before. Hence, for the HTTP TAGs derived from our campus network flow datasets, the outside host groups are good indicators for tracking the evolution of graph components over time. In addition, the two curves for the domain and AS similarity metrics are quite similar. We notice that for one of the domain name groups, the similarity values are close to zero while AS similarity is quite high ($\text{sim}_{as} = 0.65$). By investigating these domain names, we find that they indeed belong to the same AS, but are associated with different domain names, such as *questionmarket.com* and *Advertising.com*.

In the following we will use the domain similarity metric to study the temporal stability of graph components over time.

6.5.2 Temporal Stability of Graph Components

Given a graph component C_p^t observed during the time interval t, we say that it also appears at time s, $s \neq t$ and $1 \leq s \leq 72$, if its best match is C_q^s at time interval s such that $\text{sim}_d(C_p^t, C_q^s) \geq \eta$. (Note that the domain similarity of two graph components is determined solely by the domain similarity of their associated outside host groups.) We define the *lifetime* of a graph component C as the number of time intervals that C appears in. With η ranging from 0.7 to 1, Fig. 6.9c plots the corresponding CDF of the lifetimes of various graph components observed during a one-day period. We see that even with $\eta = 1$, a few graph components appear in more than 65 time intervals, and with $\eta = 0.9$, about 10% of graph components appear in all 72 time intervals (whole day). Using $\eta = 0.9$, we say a graph component is *persistent* if it has a lifetime more than 6 h (i.e., if it appears in at least 18 time intervals), otherwise it is referred to as *transient*.

Using the domain names of the outside host groups, we examine what constitutes the majority of persistent graph components. We find that a majority of them are associated with popular Web sites/services such as *Google*, *Yahoo*, *Facebook* as well as CDNs such as *LimeLight Network (LLNW)* and *Akamai*, where the outside host groups represent part of the server farms. Some of these persistent components contain also "correlated" servers/services, e.g., *Yahoo*, *DoubleClick* and *LLNW*. A few persistent components also represent groups of related Web sites such as *dictionary.com*, *thesaurus.com*, and *lexico.com*, or *govideocodes.com* (video site) and *photobucket.com*, which appear to represent user interests. In other words, inside hosts that access one site are also likely to access the other sites in the (outside host) group. In contrast, most transient components seem to represent correlated Web sites, services and outside hosts that reflect user interests, some of which appear in multiple time intervals during the day, while others only appear in a short period of the day. In addition to some examples listed in Table 6.4, other examples include *cnet* (software news voice broadcast), *onvoy* (voice services) and *apple.com*; music services including *musicmatch*, *napster*, *moontaxi*, and *live365*; or travel-related services *grandex.com* and *weather.com*.

Furthermore, there is an implicit correlation between the "cohesiveness" of graph components and user interests and activities during different time periods of the day. In Fig. 6.9d, we plot the average of the (domain) similarities between the graph components observed at time interval t with their "best matches" at $t + 1$ ($\eta = 0$ is used for this purpose) as a function t. We see that between midnight and 2 am or so ($t = 0$ to $t = 8$), the average similarities of the graph components are generally higher than other times. We find that an overwhelming majority of the graph components that appear during these periods, are associated with popular media sharing sites and other common Web services such as *Google*, *Yahoo*, *Microsoft*, and *AOL*. Examining the inside hosts associated with these graph components reveal that most of them come from the residential hall subnets. Thus, graph components during these time periods reflect activities and interests of residential hall students. During the business hours and evening ($t = 30 - 60$), many graph components are associated with common Web services, e.g., news, weather, etc., which appear to reflect dominant interests of users during those periods. On the other hand, during the wee hours of the morning, the traffic activities are much lower, and graph components appear to be more mixed: more (outside) service groups show up, each attracting roughly similar number of users (inside hosts), without any type of services dominating. The inside hosts associated with these graph components are also more diverse, including hosts in residential subnets, departmental machines, mail servers and other servers that appear to be running automated and scheduled processes, and the corresponding outside hosts vary from academic institutions to news sites and government agencies. Because traffic activities are less intense and more diverse, the graph components extracted by the tNMF method tend to be less cohesive, resulting in lower similarities among the graph components during two consecutive time periods. This also helps explain why we see a large number of graph components appearing during some of these time periods and they also tend to be less stable (see Fig. 6.9a).

6.6 Applications

In previous sections, we have analyzed the typical structures of (significant) graph components and (core) latent hypergraphs produced by our tNMF-based TAG decomposition method, as well as how they evolve over time. In this section, we demonstrate how these analyses of graph components and hypergraph structures can be employed to identify, classify and understand "unknown" applications and their structures by using two examples.

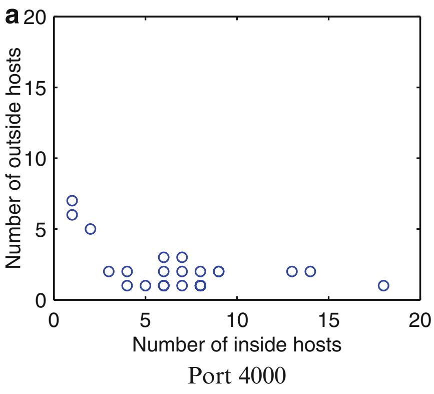
The tNMF-based graph decomposition method not only helps us understand the structural properties of application TAGs associated with *known* applications (or service ports), but these structural properties can also be applied to facilitate "unknown" application identification as well as to analyze "anomalous" behaviors in known/unknown application TAGs. We briefly illustrate these two applications of the tNMF method via two examples.

6.6.1 Analysis of UDP Port 4000 Traffic

As an example of "unknown" application identification, we apply the tNMF method to the TAG formed by outgoing flows towards UDP port 4000 (i.e., as the destination port in the flows) within a certain time window in our campus flow datasets. Given limited information of application(s) running on port 4000, we decompose the TAG and analyze the structures of the resulting graph components and latent hypergraphs. Figure 14.1a shows the size of extracted graph components, which contains 13 bi-meshs, 2 in-stars and 15 out-stars (some are overlapping in the figure). These components are connected to form an approximate pool structure. Further investigation shows these destination IP addresses belong to the same AS in China, indicating this TAG is likely associated with a messenger or game type of application. Googling these destination addresses reveals that they are associated with a messenger software (OICQ) that mainly uses UDP 4000 as the service port.

6.6.2 Analysis of Storm Worm Traffic

Storm worm is now a notorious and well-studied giant botnet in which bots communicate with each other through a P2P network (Overnet). It first appeared



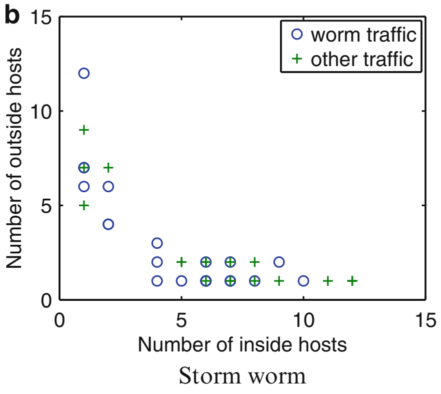


Fig. 6.10 Example applications using tNMF

in late 2006 or early 2007. The ISP flow datasets were collected in early summer of 2007, during which the Storm worm is known to be highly active. As an example to illustrate the utility of the tNMF method, we apply it to analyze the structural properties of Storm worm TAGs and investigate how they may deviate from those of "normal" P2P applications. For this purpose, we have obtained a list of "known" bot addresses culled from the P2P queries to/from a Storm worm bot captured in a honeynet, and used this list to extract all flows in the ISP flow datasets that contain the IP addresses (either as source or destination) on the list. Note that unlike application TAGs discussed earlier, here, we ignore the port information when extracting the flowset. We construct two TAGs[7] from the resulting flow set: one TAG is constructed using flows containing both source and destination IP addresses on the list, thus it represents the communications among bots themselves (referred to as "worm traffic" in Fig. 14.1b); the other TAG is constructed using flows between bots (i.e., those IP addresses on the list) and "non-bot" hosts (IP addresses not on the list), thus it represents the communications between bots and non-bots (referred to as "other traffic" in Fig. 14.1b).

Applying the tNMF method to these two TAGs associated with the Storm worm, we obtain the graph components which are shown in Fig. 14.1b. The "bot communication" TAG contains 8 bi-meshs out of 22 components ("o" points in the figure). The more common appearance of bi-mesh structures distinguishes it from other "normal" P2P networks where only randomly connected in-star structures

[7]In the first (Storm worm communication) TAG, we construct a bipartite graph by putting source IP addresses on one side (rows in the adjaceny matrix) and destination IP addresses on the other side (columns in the adjaceny matrix). The resulting graph is not *strictly* bipartite, but nearly so, as there are only 10 (0.1%) IP addresses that appear as both source and destination in the flows. In the second (bots communicating with non-bots) TAG, a bipartite graph is constructed with bot IP addresses on one side and non-bot IP addresses on the other side.

are observed[8]. This indicates that the Storm worm bots tend to communicate more frequently with each other than peers in other "normal" P2P applications. We provide one plausible explanation for this structural difference or *anomaly*: the Storm worm botnet has a hierarchical structure (see [23]), where bots acquire commands from the botmaster through a set of supernodes. The role of the P2P communications between Storm worm bots is to query for the addresses of the supernodes. Hence, the bi-meshes are likely due to the bots periodically sending queries to a few bots that store the addresses of the supernodes. In other words, the P2P communications in the Storm worm botnet are of certain mutual interest. This is in contrast to the host behaviors in most "normal" P2P applications, where users search for and share content in a "random" fashion. The "bots communicating with non-bots" TAG ("+" points in Fig. 14.1b) also contains a significant percentage of bi-mesh structures (7/20). Examining the DNS names and other relevant information (e.g., via *Google*) associated with non-bot IP addresses and ports used in the communication, we find many of the non-bots are (or function as) mail or http servers (perhaps functioning as distributed supernodes to provide proxies for communication between bots and the botmaster). The large number of bi-mesh structures reveals that Storm worm bots tend to exhibit somewhat correlated behaviors, either "collaborating" in accessing mail relays or http servers, or engaging in other "coordinated" malicious activities. Most of the communications with non-bot hosts seem to be Email spam activities.

6.7 Related Work

Analysis of complex network graphs has recently received considerable attention in the literature, mostly due to the success of online social network applications. Many approaches have been proposed to help directly visualize complex graphs, e.g., [28] and [26]. These methods enable us to directly visualize and understand complex graphs, however, they do not provide us a way to characterize and quantify different graphs.

Among various properties of complex network graphs, the community structures attract the majority of interests. Newman [20] surveyed widely applied methods in physics and social sciences to extract community structures from complex graphs. Recently, a lot of work in computer science focuses on analyzing community structures from Web data [6] and social network data [15]. Nonnegative matrix factorization algorithms are closely related to these algorithms [5, 16]. In recent years, NMF algorithms have been applied for identifying communities in large complex graphs and demonstrate superior results over other well-known community detection algorithms [22, 25].

[8]That bi-mesh structures are generally rare in "normal" P2P TAGs is likely due to the random peer selection method used by many P2P applications. Hence, the probability of two P2P hosts sharing many peers repeatedly is typically very small in a short time window.

One related work on host-level communities is in [18], where historical communication patterns are used as "normal" profiles for preventing propagation of malwares/worms. In contrast, we are not only interested in characterizing application specific communication patterns, but also explain the formation of these communities.

Our work is also related to other matrix factorization algorithms which can potentially provide graph partitioning or co-clustering results. For example, the SVD-based spectral graph partitioning algorithm, introduced in [3], has been proved to provide optimal bi-partitioning. However, the assumption of diagonal Σ matrix forces each inside host group to only communicate with one outside group, which does not describe the rich interactions among host groups. In addition, information-theoretic co-clustering methods perform simultaneous clustering over rows and columns of a specific matrix. For example, [4] obtains such co-clustering by minimizing the loss of mutual information between a low-rank approximation and the original matrix, while [1] achieves this goal by minimizing the codelength for the original matrix after rotating its rows and columns. In the scenario of traffic activity graphs in this chapter, we are interested in identifying and interpreting the "core" community structures or densely connected subgraphs. The edges not belonging to these core communities are considered as noise and are expected to be filtered by the decomposition algorithm. The tNMF algorithm fulfills this task by enforcing that each host only belongs to one particular community through the orthogonality constraint, and relax this rigorous orthogonality constraint slightly during the optimization process (see [5] for details). In this way, the noise (i.e., the hosts across multiple communities) after factorization are associated with lower weights and hence can be easily filtered by the group membership assignment process.

6.8 Conclusions

In this chapter, we have studied the network traffic behaviors using traffic activity graphs (TAGs) associated with various types of communications. These TAGs are large, sparse, seemingly complex and richly connected, making them hard to visualize and comprehend as a whole. Based on the observation of prevalent block structures in such TAGs, we proposed the tNMF method for decomposing TAGs, and devised a systematic method for extracting the dominant substructures and characterizing their structural properties. We applied our method to various application TAGs derived from our campus NetFlow datasets such as HTTP, Email, DNS, various chat, P2P, and online gaming applications. Through extensive experimental analyses, we demonstrated that the tNMF graph decomposition method provides an easy-to-understand, interpretable and quantifiable means to characterize and quantify the key structural properties of various TAGs as well as to study their formation and evolution. As examples to illustrate the utility of the proposed tNMF method, we also briefly touched on how they can be used for unknown application identification and anomalous traffic activity detection.

References

1. Chakrabarti, D., Papadimitriou, S., Modha, D., Faloutsos, C.: Fully automatic cross-associations. In: Proc. of ACM KDD (2004)
2. Chung, F., Lu, L.: Connected components in random graphs with given expected degree sequences. Annals of Combinatorics **6**, 125–145 (2002)
3. Dhillon, I.: Co-clustering documents and words using bipartite spectral graph partitioning. In: Proc. of KDD (2001)
4. Dhillon, I., Mallela, S., Modha, D.: Information-theoretic co-clustering. In: Proc. of ACM KDD (2003)
5. Ding, C., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix t-factorizations for clustering. In: Proc. of ACM KDD (2006)
6. Dourisboure, Y., Geraci, F., Pellegrini, M.: Extraction and classification of dense communities in the web. In: Proc. of WWW (2007)
7. Iliofotou, M., Pappu, P., Faloutsos, M., Mitzenmacher, M., Singh, S., Varghese, G.: Network monitoring using traffic dispersion graphs (tdgs). In: Proc. of ACM IMC (2007)
8. K. Xu, Z.-L. Zhang and S. Bhattacharyya: Profiling Internet backbone traffic: behavior models and applications. In: Proc. of ACM SIGCOMM (2005)
9. Karagiannis, T., Broido, A., Faloutsos, M., claffy, K.: Transport layer identification of p2p traffic. In: Proc. of ACM IMC (2004)
10. Kim, H., Park, H.: Extracting unrecognized gene relationships from the biomedical literature via matrix factorizations using a priori knowledge of gene relationships. In: Proc. of ACM TMBIO (2006)
11. Lakhina, A., Crovella, M., Diot, C.: Characterization of network-wide anomalies in traffic flows. In: Proc. of ACM IMC (2004)
12. Lakhina, A., Papagiannaki, K., Crovella, M., Diot, C., Kolaczyk, E., Taft, N.: Structural analysis of network traffic flows. In: Proc. of ACM SIGMETRICS (2004)
13. Lee, D., Seung, H.: Learning the parts of objects. by non-negative matrix factorization. In: Nature (1999)
14. Lee, D., Seung, H.: Algorithms for non-negative matrix factorization. In: Proc. of NIPS (2000)
15. Leskovec, J., Lang, K., Dasgupta, A., Mahoney, M.: Statistical properties of community structure in large social and information networks. In: Proc. of WWW (2008)
16. Li, T.: Clustering based on matrix approximation: a unifying view. Knowl. Inf. Syst. **17**, 1–15 (2008)
17. Li, T., Ding, C.: The relationships among various nonnegative matrix factorization methods for clustering. In: Proc. of IEEE ICDM (2006)
18. McDaniel, P., Sen, S., Spatscheck, O., der Merwe, J.V., Aiello, B., Kalmanek, C.: Enterprise security: a community of interest based approach. In: Proc. of NDSS (2006)
19. Moore, A., Zuev, D.: Internet traffic classification using bayesian analysis techniques. In: Proc. of ACM SIGMETRICS (2005)
20. Newman, M.E.J.: Detecting community structure in networks. In: Eur. Phys. J. B 38, 321-330 (2004)
21. Newsome, J., Karp, B., Song, D.: Polygraph: automatically generating signatures for polymorphic worms. Proc. of Security and Privacy, IEEE Symposium (2005)
22. Psorakis, I., Roberts, S., Sheldon, B.: Efficient bayesian community detection using non-negative matrix factorisation (2010)
23. Stewart, J.: Inside the storm: Protocols and encryption of the storm botnet. http://www.blackhat.com/presentations/bh-usa-08/Stewart/BH_US_08_Stewart_Protocols_of_the_Storm.pdf
24. T. Karagiannis, K. Papagiannaki and M. Faloutsos: BLINC: Multilevel traffic classification in the dark. In: Proc. of ACM SIGCOMM (2005)
25. Wang, F., Li, T., Wang, X., Zhu, S., Ding, C.: Community discovery using nonnegative matrix factorization. Data Mining and Knowledge Discovery (2010)

26. X. Yang and S. Asur and S. Parthasarathy and S. Mehta: A visual-analytic toolkit for dynamic interaction graphs. In: Proc. of ACM SIGKDD (2008)
27. Xu, W., Liu, X., Gong, Y.: Document clustering based on non-negative matrix factorization. In: Proc. of SIGIR (2003)
28. Y. Jia and J. Hoberock and M. Garland and J. Hart: On the visualization of social and other scale-free networks. In: Proc. of IEEE InfoVis (2008)
29. Zhang, Y., Singh, S., Sen, S., Duffield, N., Lund, C.: Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications. In: Proc. of ACM IMC (2004)

Chapter 7
Localized Bridging Centrality

Soumendra Nanda and David Kotz

Abstract Centrality is a concept often used in social network analysis (SNA) to study different properties of networks that are modeled as graphs. Bridging nodes are strategically important nodes in a network graph that are located in between highly-connected regions. We developed a new centrality metric called Localized Bridging Centrality (LBC) to allow a user to identify and rank bridging nodes. LBC is a distributed variant of the Bridging Centrality (BC) metric and both these metrics are used to identify and rank bridging nodes. LBC is capable of identifying bridging nodes with an accuracy comparable to that of the BC metric for most networks, but is an order of magnitude less computationally expensive. As the name suggests, we use only local information from surrounding nodes to compute the LBC metric. Thus, our LBC metric is more suitable for distributed or parallel computation than the BC metric. We applied our LBC metric on several examples, including a real wireless mesh network. Our results indicate that the LBC metric is as powerful as the BC metric at identifying bridging nodes. We recently designed a new SNA metric that is also suitable for use in wireless mesh networks: the Localized Load-aware Bridging Centrality (LLBC) metric. The LLBC metric improves upon LBC by detecting critical bridging nodes while taking into account the actual traffic flows present in a communications network. We developed the SNA Plugin (SNAP) for the Optimized Link State Routing (OLSR) protocol to study the potential use of LBC and LLBC in improving multicast communications and our initial results are promising. In this chapter, we present an introduction to SNA centrality metrics with a focus on our contributed metrics: LBC and LLBC. We also present some initial results from applying our metrics in real and emulated wireless mesh networks.

S. Nanda (✉)
BAE Systems, 6 New England Executive Park, Burlington, MA 01821, USA
e-mail: soumendra.nanda@baesystems.com

D. Kotz
Dartmouth College, 6211 Hinman, Hanover, NH 03755, USA
e-mail: kotz@cs.dartmouth.edu

7.1 Introduction

Wireless mesh networks are an emerging type of wireless network technology that are related to mobile ad hoc networks (MANETs). They are used to provide network access to wireless clients (such as laptops for e.g.), but unlike cellular wireless networks (where cell towers communicate with each other through a wired backbone), the mesh nodes communicate with each other through wireless channels and in a distributed and decentralized manner with the help of a MANET routing protocol. Optimized Link State Routing (OLSR) [16] is one such MANET routing protocol, but there are hundreds of others in the literature that could be used for mesh networks. Most wireless networks are represented as graphs with the mesh nodes as vertices and with the links between neighboring nodes as edges. Link quality values are often used as weights on these edges.

During the course of a PhD dissertation at Dartmouth College in Hanover, NH, we developed a new distributed management system for wireless mesh networks, called Mesh-Mon, that can help a team of system administrators (sysadmins) manage a stationary or mobile wireless mesh network [18, 20]. Mesh-Mon is designed to provide scalable monitoring of large unplanned wireless mesh networks, by allowing mesh nodes to cooperate locally to monitor each other and detect faults and anomalies in a decentralized manner. Most traditional network analysis tools for computer networks are centralized in nature. To complement the distributed nature of mesh networks and of our management platform, we sought to develop new distributed metrics and tools that may assist in network analysis and could enhance the design of future network routing protocols.

We provide below a list of questions asked from a sysadmin's point of view, that we initially set out to answer and consider relevant to mesh networks:

1. Which nodes should the system administrator be most concerned about from a robustness point of view? That is, the loss of which nodes would have a significant impact on the connectivity of the network?
2. How many nodes can fail before the network is partitioned into multiple parts?
3. Which nodes are the most "important" in the network?
4. If the sysadmin could or should add or move a node to enhance the network, which one should it be?
5. Similarly, if the sysadmin had to update a subset of nodes and reboot them, in which order should the update be performed in?

A sysadmin is generally concerned about which nodes are more "critical" and require more scrutiny in the network. One technique to identify the nodes that are critical from a network topology management perspective is to identify all "articulation points" and "bridges" in the network, since, upon failure, these nodes will partition a network [10, 24]. When applied to wireless mesh networks, in our experience, we found that articulation points are rare in practice in mesh topologies (unless the network is sparse and weakly connected). Thus, this technique is less helpful when applied in the analysis of mesh networks. Furthermore, Depth First

Search (DFS) of the entire network is an essential computational step and it can only be implemented efficiently in a centralized manner.

While most network management issues are absolute in nature (such as dealing with faulty hardware or incorrectly configured devices), there are many situations when relative management decisions must be made. For example, consider the following questions: If the system administrator had to update a subset of nodes sequentially, then in which order should he or she perform the update? or Which nodes are the most and least “important” in my network?

Centrality is a concept often used in social network analysis (SNA) to study relative properties of social networks. These social networks are typically modeled as graphs. Our approach is to apply techniques adapted from SNA to answer relativistic questions. In a wireless mesh network context, a system administrator should pay attention to “bridging nodes” since they are important from a robustness perspective (as they help bridge connected components together) and their failure will increase the risk of network partitions.

This chapter provides an introduction to centrality metrics used in social network analysis and describes some of our own recently published contributions: the development and evaluation of two new SNA-based centrality metrics: the *Localized Load-aware Bridging Centrality (LLBC)* metric, and the *Localized Bridging Centrality (LBC)* metric [18–21]. Our second contribution is the development of an OLSR plugin to study the applicability of LBC, LLBC and EigenVector Centrality (EVC) in mobile networks and evaluation via simulations in an emulated 802.11 environment using the Extendable Mobile Ad-hoc Emulator (EMANE) by CenGen Inc [8]. Both LLBC and LBC provide functionality that is comparable to or better than the Bridging Centrality (BC) metric [15] at identifying bridging nodes, yet can be calculated quickly and in a distributed manner. BC is calculated in a centralized manner using the entire network graph and has an order of magnitude higher computational complexity. To calculate its own LBC value, each node only needs to know its 1-hop neighbor set and the degree of each of its neighbors. Additionally, for LLBC calculations, each node only requires the aggregate traffic summary of its direct neighbors.

7.2 Social-Network Analysis

Our initial motivation for this work was to discover metrics and develop tools that can help a system administrator manage a wireless mesh network or would allow an automated management system understand the state of a network. We use “centrality” metrics from social-network analysis to study the roles of individual nodes in the network and the relationship of these nodes to their neighbors. Social-network analysis is normally applied to the study of social networks of people and organizations. In our domain, we are interested in the positions and roles of individual mesh nodes and the relationships between them, which like social networks are often represented as graphs.

Many social-network analysis techniques and metrics are based on graph theory. Humans tend to form clusters of communities within social networks. Similarly, mesh networks may have a complex network structure with groups of nodes that share a common relationship or structure that is worth identifying. Although we refer to wireless mesh networks as our primary domain for the application of our techniques, they are general enough to apply to other domains that use graphs to represent complex networks.

The connectivity relationship between different mesh nodes can be characterized in many different ways, such as direct or indirect, and weak or strong. The position a node occupies in a network can play a role in the node's ability to control or impact the flow of information in the network. Centrality indices help characterize the position of a node in different ways.

The most common centrality metrics used in social network analysis are degree centrality, closeness centrality, eigenvector centrality (EVC) [2] and sociocentric betweenness centrality [13]. Several other definitions of centrality measures exist. A popular software package for experimenting with Social Network Analysis metrics is provided by Analytic Tech [5], but there are many other tools available. We focus on sociocentric betweenness centrality and use it to develop our own centrality measures later in this text.

We believe that techniques borrowed and enhanced from the domain of social network analysis can help in providing answers to some of the questions we pose. We aim to use "centrality" metrics from social-network analysis to study the roles of individual nodes in the network and the relationship of these nodes to their neighbors. Social-network analysis is normally applied to the study of social networks of actors, usually people and their relationships with other people. In our domain, we are interested in the positions and roles of individual mesh nodes and the relationships between them.

7.2.1 Degree Centrality

One simple way to characterize an individual node in a topological graph is by its *degree*. The *degree* of a node in a graph in the mesh context is the number of links the node shares with its neighbors and which are available for routing purposes. A well-connected mesh network is a healthy network. If a node has many neighbors then the failure of a single neighbor should not affect the routing health of the regional network adversely. A node with a high degree can be considered as being well connected and a node with a relatively low degree can be considered weakly connected. The degree of an individual node and the minimum, maximum and average degree over all the nodes are standard characterization metrics in graph theory.

If the global topology is available at a central location, then all the nodes can be quickly ranked according to their degree. However, this degree-based ranking does not convey a good picture of the nature of connectivity in the network since

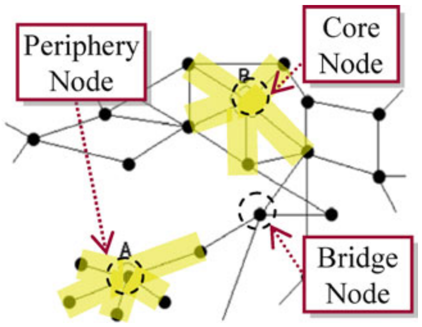


Fig. 7.1 Limitations of degree centrality. Note that nodes A and B each have degree 5 but are likely to have different impact on a network. (Adapted from [1])

all links are rarely identical. For instance, different links may have varying capacity levels and different latencies. In addition, the existence of neighbor links and their respective qualities fluctuate over time. In a wireless network, a link with a poor-quality connection has lower effective capacity and a link using a lower bit-rate may have a higher latency (Fig. 7.1).

Furthermore, two nodes may have the same degree, but they may have very different characteristics due to their relative positions [1]. There are other centrality metrics, such as eigenvector centrality, that can help distinguish between nodes A and B that have the same degree centrality.

7.2.2 Eigenvector Centrality

Eigenvector Centrality (EVC) is a concept often used in social-network analysis and was first proposed by Bonacich [2,3]. Eigenvector Centrality is defined in a circular manner. The centrality of a node is proportional to the sum of the centrality values of all its neighboring nodes. In the social-network context, an important node (or person) is characterized by its connectivity to other important nodes (or people). A node with a high centrality value is a well-connected node and has a dominant influence on the surrounding network. Similarly, nodes with low centrality values are less similar to the majority of nodes in the topology and may exhibit similar characteristics and behavior and share common weaknesses. Google uses a similar centrality ranking technique (called Pagerank) to rank the relevance of hyper-linked pages in search results [7].

Eigenvector centrality is calculated using the adjacency matrix to find central nodes in the network. Let v_i be the ith element of the vector $\mathbf{v}$, representing the centrality measure of node i, where $N(i)$ is the set of neighbors of node i and let A be the $n \times n$ adjacency matrix of the undirected network graph. Eigenvector centrality is defined using the following formulas:

$$v_i \propto \sum_{j \in N(i)} v_j \tag{7.1}$$

which can be rewritten as

$$v_i \propto \sum_{j=1}^{n} A_{ij} v_j \tag{7.2}$$

which can be rewritten in the form

$$A\mathbf{v} = \lambda \mathbf{v} \tag{7.3}$$

Since A is an n x n matrix, it has n eigenvectors (one for each node in the network) and n corresponding eigenvalues. One way to compute the eigenvalues of a square matrix is to find the roots of the characteristic polynomial of the matrix. It is important to use symmetric positive real values in the matrix used for calculations [4].

The principle eigenvector is the eigenvector with the highest eigenvalue. The principle eigenvector is recommended for use in rank calculations [2]. After the principle eigenvector is found, its entries are sorted from highest to lowest values to determine a ranking of nodes. The most central node has the highest rank and most peripheral node has the lowest rank.

This metric is often used in the study of the spread of epidemics in human networks. In the mesh context, a node with a high eigenvector centrality represents a strongly connected node. A worm or virus propagated from the most central node could spread to all reachable nodes in the most efficient manner as opposed to one that was spreading from a node on the extreme periphery. Thus, the central node is a prime target for preventive inoculation or for prioritized software update.

In any network, and especially in an ad hoc or mesh network where nodes must cooperate with each other to route packets, the connectivity of a node depends on the connectivity of its neighbors and EVC can help capture this property. The main drawback of eigenvector centrality is that it can only be calculated in a centralized manner.

7.2.3 *Sociocentric Betweenness Centrality*

The betweenness centrality of a node is calculated as the fraction of shortest paths between all node pairs that pass through a node of interest. A node with a high betweenness centrality value is more likely to be located on the shortest paths between multiple node pairs in the network, and thus more information flow must travel through that node (assuming a uniform distribution of information across node pairs). Since all pairs of shortest paths must be computed, the time complexity is $\theta(n^3)$, where n is the number of nodes in the entire network. Brandes presents a fast technique to compute betweenness centrality that runs in $O(VE)$ time and uses $O(V+E)$ space for undirected unweighted graphs with V nodes and E edges [6].

7.2.4 Egocentric Betweenness Centrality

A more computationally efficient approach is to calculate betweenness on the "egocentric" (or ego) network, rather than the global network topology. In social networks, an egocentric network is defined as network of a single actor (or node) together with the actors (also referred to as alter-egos) that this actor are directly connected to, that is, the node's neighbors in the graph and all the corresponding links between these nodes. For application in wireless networks, the ego network is exactly the same as the 1-hop adjacency matrix representation of the connectivity graph centred around any given node of interest.

Thus, for wireless mesh networks we calculate egocentric betweenness on the one-hop adjacency matrix of a node and not the entire network (as would be the case for calculating betweenness in the sociocentric betweenness metric). This egocentric betweenness metric can be calculated in a fully distributed manner and the computational complexity is $\theta(k^2)$ where k is size of the 1-hop neighborhood and thus this computation is one order of magnitude faster than computing the global betweenness centrality score.

Sociocentric betweenness centrality is a key component of the bridging centrality metric, while our metrics LBC and LLBC are based on egocentric betweenness centrality.

The simple structure of an ego network allows for fast computation of the egocentric betweenness centrality (also referred to as ego betweenness). Everett and Borgatti developed the following fast technique (and illustrative example) to calculate ego betweenness [12]. If the ego network is represented as a $n \times n$ symmetric matrix A where 1 represents a link between node ii and node j and 0 represents the absence of a link, then the betweenness of the ego node is the some of the reciprocals of the entries of $A^2[1-A]_{i,j}$. This value has to be divided by two if the graph is symmetric.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.4}$$

For a faster implementation, first note which entries in $A^2[1-A]_{i,j}$ will be non-zero (for a symmetric matrix, we only need to consider the zero entries above the leading diagonal) and calculate $A^2[1-A]_{i,j}$ only for those entries.

$$A^2[1-A]_{i,j} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & 2 & 1 \\ * & * & * & * & 1 \\ * & * & * & * & 1 \\ * & * & * & * & * \end{bmatrix} \tag{7.5}$$

The ego betweenness is the sum of the reciprocals of the entries of interest and in this example it is $\frac{1}{2} + 1 + 1 + 1 = 3.5$.

7.2.5 Bridging Centrality

Bridging Centrality (BC) is a centrality metric introduced by Hwang et al. [15]. Bridging centrality can help discriminate *bridging nodes*, that is, nodes with higher information flow through them, and locations between highly connected regions (assuming a uniform distribution of flows).

The Bridging Centrality of a node is the product of its sociocentric betweenness centrality C_{Soc} and its bridging coefficient $\beta(v)$. The Bridging Centrality $\text{BC}(v)$ for a node v of interest is defined as:

$$\text{BC}(v) = C_{\text{Soc}}(v) \times \beta(v) \tag{7.6}$$

The bridging coefficient of a node describes how well the node is located between high-degree nodes. The bridging coefficient of a node v is defined as:

$$\beta(v) = \frac{\frac{1}{d(v)}}{\sum_{i \in N(v)} \frac{1}{d(i)}} \tag{7.7}$$

where $d(v)$ is the degree of node v, and $N(v)$ is the set of neighbors of node v.

According to the authors, betweenness centrality indicates the importance of a given node from an information-flow standpoint, but it does not consider the topological position of the node. On the other hand, the bridging coefficient measures only how well a node is located between highly-connected regions, but does not consider information flow. “Bridging nodes” should be positioned between clusters and also located on important positions from an information-flow standpoint. Thus, their BC metric is an attempt to combine these two distinct metrics by giving equal weight to both factors. Based on their empirical studies, the authors recommend labeling the top 25th percentile of nodes as ranked by BC as “bridging nodes,” that is, nodes that are more bridge-like and lie between different connected modules [15].

We note that these bridging nodes are different from the *articulation points* of a graph that one can discover during topological analysis (via DFS), though some bridging nodes are articulation points. These bridging nodes provide the system administrator with a prioritized set of critical nodes to monitor from a robustness perspective (as they help bridge connected components together) and their failure may increase the risk of network partitions. BC can only be calculated in a centralized manner with global information.

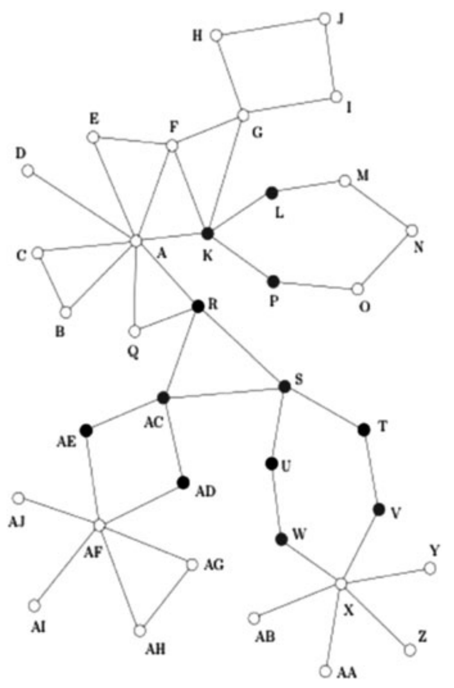


Fig. 7.2 Top bridging nodes as identified and ranked by Localized Bridging Centralityn with higher ranked nodes *shaded darker*. (Adapted from [15])

7.2.6 Localized Bridging Centrality

In previous work, we introduced our distributed equivalent of Bridging Centrality that we call Localized Bridging Centrality (LBC) [19]. As the name suggests, we define $\mathrm{LBC}(v)$ of a node v using only local information, as the product of egocentric betweenness centrality $C_{\mathrm{Ego}}(v)$ and its bridging coefficient $\beta(v)$. The definition of $\beta(v)$ is unchanged from (7.7). LBC is thus defined as (Fig. 7.2):

$$\mathrm{LBC}(v) = C_{\mathrm{Ego}}(v) \times \beta(v) \tag{7.8}$$

Marsden [17] and Everett and Borgatti [12] showed empirically that egocentric betweenness values have a strong positive correlation to sociocentric betweenness values (calculated on the complete network graph) for many different network examples. While these networks were derived from social networks, in many cases they are similar to wireless mesh networks. Our LBC results also benefit from similar correlations and are thus nearly as accurate as BC results, while being easier to compute with only local information. As you can see in Sect. 7.2.5, LBC is able to identify key bridge nodes and articulation points.

Prior to us, Daly and Haahr applied egocentric betweenness centrality to develop SimBet, a distributed routing protocol in a mobile delay-tolerant network (DTN) [11]. Their approach too benefits from the strong correlation between egocentric and sociocentric betweenness, but is designed for a DTN routing protocol. Our work focuses on real-time network routing protocols like OLSR and for distributed network management for a MANET.

The LBC metric can help the system administrator identify the bridging nodes in the mesh network, as well as clusters and their boundaries, but its distributed nature makes it suitable for use in routing protocol design as well. While individual nodes can calculate their own LBC metric in a fully distributed manner, to determine the global rank of each node, a central node must aggregate all LBC values or all nodes must use a distributed consensus-based ranking algorithm.

7.2.7 Localized Load-Aware Bridging Centrality

Betweenness centrality implicitly assumes that all paths between all node-pairs are equally utilized. Thus, both the BC and LBC metrics assume that a uniform distribution of traffic flows will exist between all node-pairs in the network. In a real mesh network used to provide last-mile Internet access, the distribution of traffic flows will almost certainly be non-uniform and gateway nodes will experience relatively higher traffic loads.

Taking the traffic load into consideration, we developed our new Localized Load-aware Bridging Centrality (LLBC) metric designed for distributed analysis of bridging nodes in wireless mesh networks [21]. We compute the traffic load (measured in bytes) in each node locally as the sum of all bytes originating at the node (Out), destined for the node (In), and twice the number of bytes forwarded (Fwd) by that node. We count the forwarded bytes twice in the summation since they are both received and sent by the node. In effect, this metric represents the load on the node's network interface.

$$\mathrm{Load}(v) = \mathrm{In}(v) + \mathrm{Out}(v) + 2 \times \mathrm{Fwd}(v) \tag{7.9}$$

We use the measured traffic load to calculate the Load Coefficient (β_{t}) as the ratio of the traffic load of a given node to the sum of the traffic loads of its one-hop neighbors. As the load of a node increases (relative to that of its neighbors' loads), so do the chances of the node becoming a traffic bottleneck.

$$\beta_{\mathrm{t}}(v) = \frac{\mathrm{Load}(v)}{\sum_{i \in N(v)} \mathrm{Load}(i)} \tag{7.10}$$

We define LLBC as the product of Ego-Betweenness and the Load Coefficient.

$$\mathrm{LLBC}(v) = C_{\mathrm{Ego}}(v) \times \beta_{\mathrm{t}}(v) \tag{7.11}$$

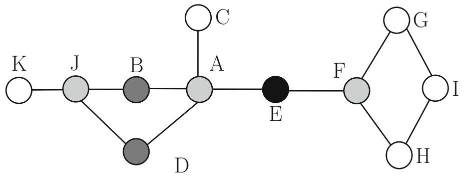


Fig. 7.3 A small synthetic network example with its top six high bridging score (via LBC) nodes *shaded* (Adapted from [15])

Thus, the LLBC metric takes into account both the current traffic load and the relative position of nodes, and (like the LBC metric) can be calculated in a fully distributed manner. Over time, the measured traffic load at different nodes will change and nodes that reboot will have their counters reset to zero. Thus, it makes sense to periodically sample LLBC values and to consider the traffic load during the sampling period instead of cumulative values.

It is important to remember that centrality measures can only provide *relative* measures that can be used to compare nodes against each other at that instant of time for that specific network topology. This ranking allows a system administrator to prioritize management tasks on several nodes or to identify which nodes are most likely to cause partitions through failure or mobility. Both of our metrics (LBC and LLBC) are easier to compute than the BC metric. A similar load-based bridging centrality can be applied to the study of road networks and airline paths. For wireless networks with multiple interfaces the load should be weighted relative to the available capacity of that link.

7.3 Evaluation

We now present our results from the application of the BC, LBC and LLBC metrics on the topology of a wireless mesh network we deployed in our department. We verified all calculations using UCINET, a popular SNA tool [5]. Two or more nodes with the same centrality value were assigned the same rank.

7.3.1 Synthetic Network Example

We first tested our LBC metric using a synthetic network, presented in Fig. 7.3 (this network was also used by Hwang et al. [15]). The rankings produced by Bridging Centrality and Localized Bridging Centrality shown in Table 7.1 are nearly identical, although we note that the BC and LBC values are clearly not identical, nor are the betweenness measures used. Since both BC and LBC are used as a "relative" measure of how nodes differ from each other, the induced ranking is more important than the magnitude of the BC or LBC value and thus in this example our LBC metric is exactly equivalent to the BC metric.

Table 7.1 Top six centrality values for the network shown in Fig. 7.3, including sociocentric betweenness (C_{Soc}), egocentric betweenness (C_{Ego}), bridging coefficient (β), bridging centrality (*BC*) and localized bridging centrality (*LBC*)

Node	Degree	C_{Soc}	C_{Ego}	β	BC	LBC	Rank of BC	Rank of LBC
E	2	0.533	1	0.857	0.457	0.857	1	1
B	2	0.155	1	0.857	0.133	0.857	2	1
D	2	0.155	1	0.857	0.133	0.857	2	1
F	3	0.477	3	0.222	0.106	0.666	4	4
A	4	0.655	6	0.100	0.065	0.600	5	5
J	3	0.211	3	0.166	0.035	0.499	6	6

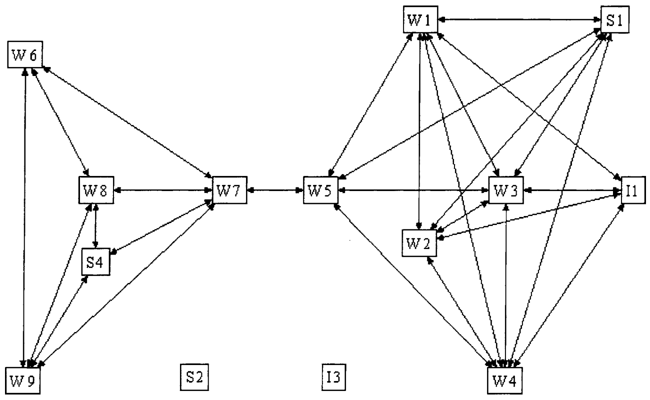


Fig. 7.4 Bank wiring room games example (Adapted from [17])

7.3.2 Social Network Example: Bank Wiring Room

This example (presented in Fig. 7.4) represents game-playing relationships in a bank wiring room and is popular in social-network studies. Marsden [17] presented this example to show how sociocentric and egocentric betweenness measures correlate. Again, the relative ranking of nodes calculated by BC and LBC as shown in Table 7.2 are identical. Visible inspection of Fig. 7.4 shows that nodes W5 and W7 are bridging nodes, and the tie between them is a bridge between two connected components.

7.3.3 Wireless Mesh Network Examples

We present actual topologies from a mesh network test bed (called Dart-Mesh) that we deployed on all three floors of our department building [20].The mesh nodes use the Optimized Link State Routing (OLSR) [9] mesh routing protocol implemented on Linux by Tønnesen [25]. In Fig. 7.5, the rectangles represent mesh nodes and

Table 7.2 Centrality values for the network shown in Fig. 7.4 sorted by BC values

Node	Degree	C_{Soc}	C_{Ego}	β	BC	LBC	Rank of BC	Rank of LBC
W5	5	30.00	4.00	0.222	6.667	0.889	1	1
W7	5	28.33	4.33	0.179	5.074	0.775	2	2
W1	6	3.75	0.83	0.140	0.528	0.117	3	3
W3	6	3.75	0.83	0.140	0.528	0.117	3	3
W4	6	3.75	0.83	0.140	0.528	0.117	3	3
S1	5	1.50	0.25	0.222	0.333	0.055	6	6
W8	4	0.33	0.33	0.223	0.073	0.073	7	7
W9	4	0.33	0.33	0.223	0.073	0.073	7	7
W2	5	0.25	0.25	0.210	0.052	0.052	9	9
W6	3	0	0	0.476	0	0	10	10
S4	3	0	0	0.476	0	0	10	10
I1	4	0	0	0.357	0	0	10	10
I3	0	0	0	0	0	0	10	10
S2	0	0	0	0	0	0	10	10

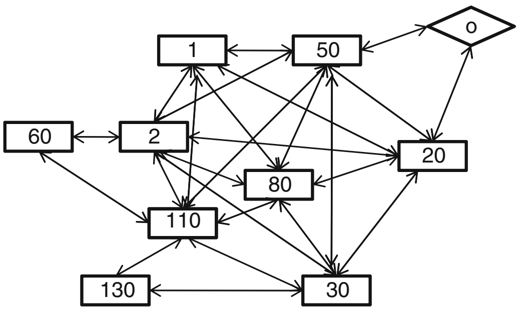


Fig. 7.5 A small OLSR mesh network that was deployed at Dartmouth College

Table 7.3 Ranked centrality values for Fig. 7.5 sorted by BC values

Node	Degree	C_{Soc}	C_{Ego}	β	BC	LBC	Rank of BC	Rank of LBC
110	7	6.367	5.75	0.078	0.496	0.448	1	1
50	7	4.733	3.40	0.096	0.454	0.326	2	3
30	6	3.367	2.75	0.132	0.444	0.363	3	2
2	7	4.067	3.4 0	0.096	0.391	0.326	4	3
20	6	2.867	2.25	0.126	0.361	0.283	5	5
80	6	0.400	0.40	0.173	0.069	0.069	6	6
1	5	0.200	0.25	0.262	0.052	0.065	7	7
60	2	0	0	1.615	0	0	8	8
130	2	0	0	1.650	0	0	8	8
0	2	0	0	1.615	0	0	8	8

are identified by the last octet of their individual IP addresses. The diamond-shaped box numbered zero is a virtual node representing the Internet. Nodes connected to the Internet are the Internet Gateways.

In the example in Fig. 7.5, while the two rankings produced (shown in Table 7.3) by the two metrics are not identical, they are quite similar. The top-five ranked nodes

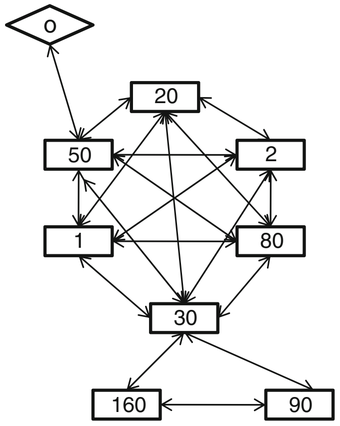


Fig. 7.6 A small real-world mesh network with one gateway at Node 50. Both Node 30 and node 50 are articulation points

are common to both metrics. If we remove any of these bridging nodes, then at least one of the other bridging nodes will become an articulation point. If that node is now removed, the network will be partitioned. For example, if node 110 (the top-ranked-node) fails, then both nodes 30 and 2 become articulation points. However, if you examine the original network graph in Fig. 7.5, you will find that it is bi-connected and has no articulation points. Thus, LBC can help detect nodes that may not be articulation points, but with certain perturbations in the network are the most likely candidates to become articulation points. Our LBC metric allows the system administrator to gather this information in a distributed manner using fewer computational resources than the BC metric.

7.3.4 *LLBC vs. LBC vs. BC*

7.3.4.1 Real-World Mesh Network with One Gateway

We now present examples where we considered how the volume and direction of traffic flowing in the network affected our metrics. We applied our Localized Load-aware Bridging Centrality (LLBC) metric on the network shown in Fig. 7.6. Node 50 was the sole Internet Gateway providing Internet connectivity to the whole mesh. The topology of the network did not change during this experiment, which was 10 min long. The BC, LBC, and LLBC results are presented in Table 7.4 and the nodes are sorted in decreasing order of LLBC values. The Load metric is given here in bytes.

During this experiment, node 80 had a high traffic load since we connected one of our mobile clients to that node, then proceeded to download large video files to that client from the Internet using node 50 as our Internet gateway. According to the LBC results, which only consider the topology of the network, node 30 was a more important "bridging node" than node 50. Node 30 is an articulation point in this

Table 7.4 Ranked centrality values for the network shown in Fig. 7.6, sorted by LLBC values

Node	Degree	Load	C_{Ego}	β	β_t	BC	LBC	LLBC
50	6	30871080	2	0.176	1.232	0.353	0.352	1.949
30	7	274027	10	0.0726	0.0043	0.8712	0.726	0.0438
80	5	30679118	0	0.219	0.962	0	0	0
1	5	262501	0	0.219	0.0042	0	0	0
2	5	238071	0	0.219	0.0038	0	0	0
20	5	218143	0	0.219	0.0035	0	0	0
160	2	94005	0	0.777	0.2571	0	0	0
90	2	91602	0	0.777	0.2488	0	0	0

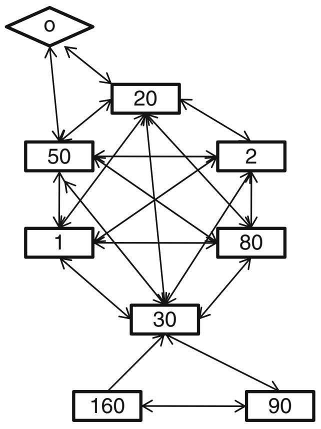


Fig. 7.7 A small mesh network with two gateways. Node 50 and node 20 are the two Internet gateways but Node 30 is an articulation point in the network

example. However, our LLBC results accurately show that node 50 was the most important bridging node by taking into consideration the traffic load on the network during our experiment.

7.3.4.2 Real-World Mesh Network Example with Two Gateways

We next applied our LLBC metric on a similar network topology similar to the one used in the last experiment by converting node 20 into an Internet gateway. The topology of this network is shown in Fig. 7.7, and now nodes 50 and 20 are the two Internet gateways. The BC, LBC and LLBC results are presented in Table 7.5 and the results are sorted in decreasing order of LLBC values.

Since there were two Internet gateways, traffic flowing to and from the Internet could go through either gateway, depending on the route selected by the routing protocol. LBC picked node 30 as its top bridging node. While this node was indeed a critical node, there was little traffic flowing through this node, so it had little influence on the traffic flowing in the network or on the majority of the nodes, most of which were forwarding Internet-bound traffic through the two gateways.

Table 7.5 Ranked centrality values for the network shown in Fig. 7.7, sorted by LLBC values

Node	Degree	Load	C_{Ego}	β	β_t	BC	LBC	LLBC
50	6	32989000	2	0.118	1.123	0.354	0.236	2.246
30	7	305327	10	0.0739	0.0049	0.8868	0.738	0.0489
20	6	1125000	2	0.118	0.0183	0.354	0.236	0.0367
80	5	16208854	0	0.219	0.3512	0	0	0
1	5	11011448	0	0.219	0.2144	0	0	0
2	5	722022	0	0.2282	0.01171	0	0	0
90	2	145226	0	0.7778	0.3358	0	0	0
160	2	127098	0	0.7778	0.2820	0	0	0

LLBC picked node 50, in fact the most-heavily-used gateway node, as the most important bridging node and indicated that node 30 (a non-gateway node) was a more important bridging node than the gateway node 20, even though node 30 had only one fourth the traffic load of node 20 in absolute terms. The importance ranking generated by LLBC is insightful. In this scenario, if node 30 failed, then nodes 90 and 160 would be partitioned from the rest of the network. Whereas, if node 20 failed, there was still a potential backup path to the Internet through 50; the LBC rankings were unable to capture this subtle complexity present in this network. The BC ranking was identical to the LBC ranking, and thus not as helpful as the LLBC metric in this scenario. The distributed manner in which LLBC is calculated also complements a distributed analysis engine, such as the one used in Mesh-Mon.

7.4 SNA Plugin (SNAP) for OLSR

To study the practical utility of LBC, LLBC, and EVC in an OLSR MANET for analysis and performance improvement, we developed a Social Network Analysis Plugin (SNAP) as shown in Fig. 7.8 [21]. While we use OLSR for our example, the same design can potentially benefit other MANET routing protocols.

OLSR is a unicast protocol but floods all multicast traffic via Multi-Point Relays (MPRs) in the Basic Multicast Forwarding (BMF) plugin extension. We developed a simple distributed algorithm that ranks 1-hop neighbors according to calculated LBC or LLBC scores and then each node locally adjusts its own advertised MPR-Willingess parameter slightly up or down as per its relative ranking.

The MPR-Willingness parameter is used by OLSR running on each node to decide if it should become an MPR and can range from 0 (never become an MPR) to 7 (always be an MPR). Having too many MPRs leads to excessive flooding and waste of spatial resources while having too few will lead to insufficient coverage and poor distribution of topological information in the network. We did not modify any of OLSR's internals. We only modify one parameter that can influence how

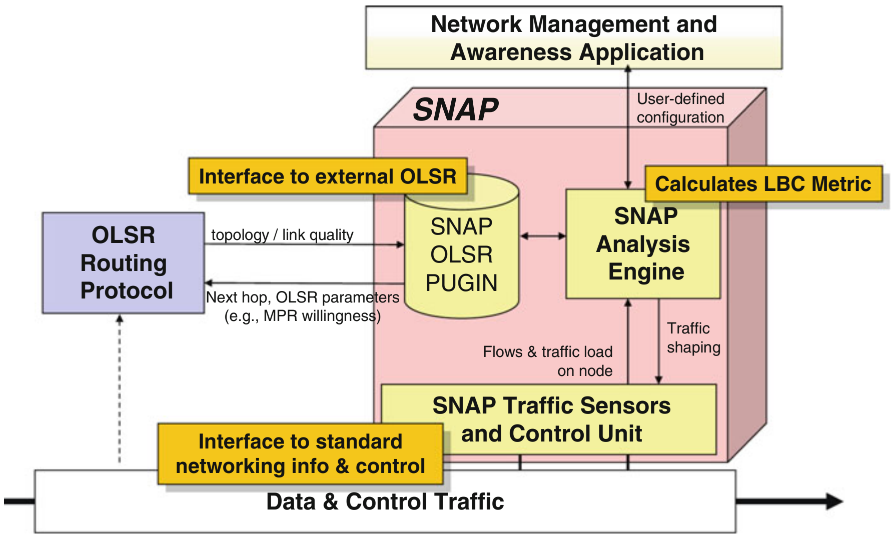

Network Management and Awareness Application
SNAP
User-defined configuration
Interface to external OLSR
Calculates LBC Metric
OLSR Routing Protocol
topology / link quality
SNAP OLSR PUGIN
SNAP Analysis Engine
Next hop, OLSR parameters (e.g., MPR willingness)
Flows & traffic load on node
Traffic shaping
SNAP Traffic Sensors and Control Unit
Interface to standard networking info & control
Data & Control Traffic

Fig. 7.8 SNA Plugin (SNAP) architecture

OLSR chooses the MPRs that it prefers to use for routing, topology distribution and multicast of packets.

Our SNA Plugin (SNAP) architecture uses a simple design and executes the following series of five sequential operations in a continuous loop every 10 s:

1. Acquire local network topology state from OLSR Routing Protocol
2. Calculate LBC metric (within the SNAP Analysis Engine)
3. Distribute local LBC metric to all 1-hop neighbors (single broadcast)
4. Compare local LBC metric value with that of all received LBC values from ego network
5. Adjust MPR Willingness locally based on relative differences in LBC scores in the ego network

Our initial hypothesis was that strong bridging nodes would serve as good MPRs for multicast communications. Our second hypothesis that we have not yet explored in depth is that LLBC can be used to enable better selection of load balanced paths in a mesh network (since LLBC can detect bottlenecks). Eigenvector Centrality (EVC) is also computed and reported for use in offline analysis in our plugin, but is not used to modify OLSR. Recently, Gao et al. [14] explored the use of a new centrality measure for Delay Tolerant Networks (DTNs) based on Poisson modelling of contacts using the egocentric network model to enhance multicast communications. Both approaches use a similar idea of selecting better and fewer relays to improve multicast delivery of data and the use of egocentric network models.

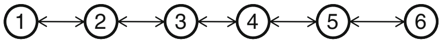


Fig. 7.9 Six node linear string topology. Any node can communicate only with its linked neighbors, but not with other nodes

SNAP-LBC and SNAP-LLBC results below show how our metrics compare against the default OLSR-BMF setup. SNAP-Degree results demonstrate the impact of substituting LBC (or LLBC) metric in our original SNAP algorithm described above with *degree centrality*. Degree centrality is a much simpler SNA metric (conceptually and computationally) than LBC or LLBC, since it is just the number of neighbors that a given node is connected to directly.

7.4.1 Initial SNAP Results

We tested our SNAP plugin on a few emulated 802.11b topologies while running a video multicast traffic application generated by the open-source Multi-Generator (MGEN) [22] software tool (developed by Naval Research Laboratory) and by using the Basic Multicast Forwarding (BMF) plugin version 1.7 for OLSR version 0.6.1 (available for download at www.olsr.org). Our SNAP plugin recomputed LBC and LLBC values and made changes to the local MPR_Willingness parameter every 10 seconds. Each source uses the MGEN tool to generate video traffic by sending packets of 1,024 bytes at 24 packets/second. For all video traffic, we added additional reliability using NACK-Oriented Reliable Multicast (NORM) protocols with Forward Error Correction with 2x redundancy (block 4, parity 4 settings) [23].

We compared performance on the basis of a custom video utility metric that was developed by an external third party. This metric takes into account a combination of the latency of packets received and number of frames that have been dropped. The maximum possible score is one indicating perfect video reception and minimum is zero. The video metric captures the notion that a video receiver is satisfied with a 10 seconds chunk of transmitted video if it receives 90% of the packets in that chunk within 4 seconds of end-to-end delay. The entire scenario is broken into 10 seconds of time slices and a time slice video utility score is computed for each slice for all intended destination receivers. All these time slice utilities are combined into an average scenario score for the whole experiment.

7.4.1.1 Six Node Static Linear Topology

Our first test for SNAP-enhanced version of OLSR was with a six node linear string topology with two sources for video traffic at opposite ends (node 1 and node 6 in Fig. 7.9) and with each source also acting as destination for the other source.

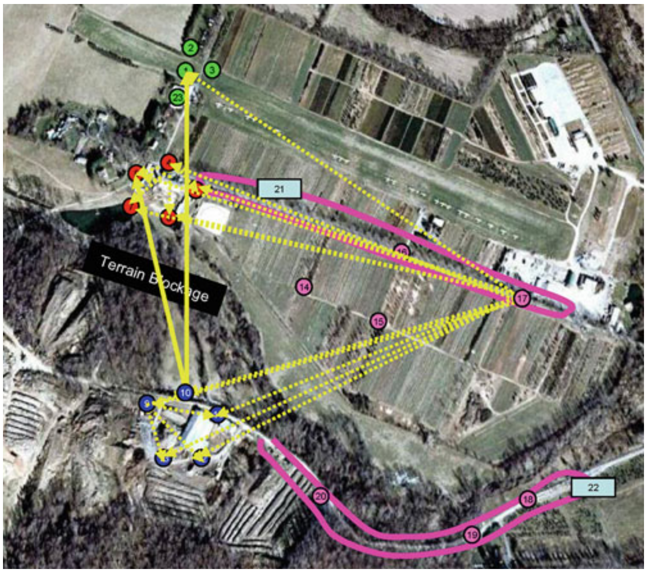


Fig. 7.10 Snapshot from the 23 node mobile test with four multicast video sources and 19 video destinations

In this example, we found no difference in performance between the default BMF multicast and the LBC or LLBC influenced multicast. We repeated each experiment three times and reported the average. The results match our expectations, since every node in the string must forward all multicast traffic that it receives and there is no alternate path for the multicast traffic and no room for optimization or improvement.

7.4.1.2 Twenty Three Node Real World Topology

We then tested our plugin on emulated scenarios with upto 23 mobile nodes (See Fig. 7.10) with upto four multicast video sources and upto 19 destinations for 300 s. The scenarios use GPS logs and pathloss recordings from an outdoor experiment with OLSR nodes and our emulation test range provides performance similar to that recorded in those real experiments. Most of the nodes moved at a slow walking speed and two nodes moved in two vehicles at 10 MPH (along the thicker curved shaded paths in Fig. 7.10). Traffic was sent to local clusters as well as clusters of nodes far away to ensure that we had multi-hop communications. We repeated each experiment three times and reported the average.

The average performance of our LBC and LLBC enhanced multicast strategy showed some significant improvements (See Table 7.6) over the default behavior of BMF and in particular, the performance of SNAP-LLBC was the best overall. We tested the performance of SNAP-LBC, SNAP-LLBC, and SNAP-degree at higher

Table 7.6 Video metric utility scores for SNAP

	OLSR-BMF	SNAP-LBC	SNAP-LLBC	SNAP-Degree
6 node linear static (1x load)	0.91	0.91	0.91	0.91
23 node mobile test (0.1x load)	1.0	1.0	1.0	1.0
23 node mobile test (0.5x load)	1.0	1.0	1.0	1.0
23 node mobile test (0.75x load)	0.27	0.81	0.92	0.36
23 node mobile test (1x load)	0.21	0.23	0.28	0.22
23 node mobile test (1.2x load)	0	0	0	0

and lower traffic loads, and found that when the load (1x = 24 packets per second) was reduced to 0.75x we found very large gains for SNAP-LBC and SNAP-LLBC, while SNAP-Degree and the defaults did relatively worse. When we increased the load to 1.2x, all protocols tested gave zero utility results.At the lowest loads, all protocols did equally well.

Our analysis of the individual experiment logs indicated that SNAP and BMF were initially selecting the same MPRs for forwarding multicast traffic. Upon further analysis and at higher loads, we found SNAP-LBC and SNAP-LLBC did better because they were more selective in their choice of MPRs and thus flooded less traffic than the other strategies that used more MPRs and sent traffic more often. In a busy saturated network, sending too much traffic leads to more collisions and that is where SNAP showed the most gains. We are uncertain if the heuristic used by SNAP-LBC or SNAP-LLBC was leading to an optimal MPR coverage (in our tested scenarios) but the results do provide ample evidence for our first hypothesis that bridging nodes have the right characteristics to be useful as MPRs for multicast traffic.

These results collectively helps better understand the benefits of choosing LBC and LLBC over other potential SNA centrality metrics for this application. We did not do studies with eigenvector centrality or sociocentric betweenness centrality measures because these two metrics can only be centrally calculated at a much higher computational expense and require complete topology information, whereas we require localized computations in a distributed environment for better efficiency.

We need to explore more topologies (real and simulated) and other alternative MPR selection strategies, before we can conclude whether the use of LBC or LLBC is always preferable to the default multicast strategy used by OLSR and to identify any potential drawbacks, but our initial results with SNAP using the LBC and LLBC metrics look very promising.

7.5 Conclusion

In this chapter, we demonstrated the use of novel social network metrics to solve the problem of identifying important nodes in wireless mesh networks for system administrators. The same tools can be used for other applications in any complex network represented as a graph. We introduced a new centrality metric called the Localized Load-aware Bridging Centrality (LLBC). Our evaluation of the LLBC and LBC metrics on a real mesh testbed running OLSR indicated their potential for use in routing and network analysis tools.

We demonstrated the usefulness of LLBC in identifying critical bridging nodes in a wireless mesh network from a network management perspective. Our initial results from our OLSR plugin shows that our SNA-based approach to selecting MPRs for multicast in OLSR when using the LLBC metric is beneficial in certain topologies and levels of traffic load. We are in the process of testing the properties of our new metrics on larger mesh data sets (both simulated and from real deployments) and exploring its utility in other scenarios and application domains.

We acknowledge that further evaluations are needed to validate our results. We are also exploring other variants of LLBC and LBC that take into account link-quality measures, link capacities, and other real-world effects. While we focus on the distributed analysis of a wireless mesh network topology in this chapter, our LBC and LLBC metrics have potential applications in other disciplines as well, such as for analysis of social networks, online collaboration tools, and identifying clusters and key components in complex biological structures or bottlenecks in transportation systems such as inter-state highways and flight plans.

Acknowledgements We thank Charles Tao at BAE Systems for his help in developing our OLSR plugin. We thank our families for their love, support, and patience. This research program was supported by a gift from Intel Corporation, by Award number 2000-DT-CX-K001 from the U.S. Department of Homeland Security, by Grant number 2005-DD-BX-1091 from the Bureau of Justice Assistance, and by Contract number N00014-10-C-098 from the Office of Naval Research (ONR). Points of view in this document are those of the authors, and do not necessarily represent or reflect the views of any of the sponsors, the US Government or any of its agencies.

References

1. Begnum, K., Burgess, M.: Principle Components and Importance Ranking of Distributed Anomalies. Machine Learning **58**(2), 217–230 (2005)
2. Bonacich, P.: Factoring and weighting approaches to status scores and clique identification. Journal of Mathematical Sociology **2**(1), 113–120 (1972)
3. Bonacich, P.: Power and Centrality: A Family of Measures. The American Journal of Sociology **92**(5), 1170–1182 (1987)
4. Bonacich, P., Lloyd, P.: Eigenvector-like measures of centrality for asymmetric relations. Social Networks **23**(3), 191–201 (2001)
5. Borgatti, S., Everett, M., Freeman, L.: UCINET for Windows: Software for Social Network Analysis. Harvard: Analytic Technologies (2002)
6. Brandes, U.: A faster algorithm for betweenness centrality. Journal of Mathematical Sociology **25**(2), 163–177 (2001)
7. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems **30**(1-7), 107–117 (1998)
8. CenGen Inc.: Extendable MobileAd-hoc Network Emulator. http://labs.cengen.com/emane/

9. Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental) (2003). URL http://www.ietf.org/rfc/rfc3626.txt
10. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, second edn. Cambridge, Mass.: MIT Press (2001)
11. Daly, E.M., Haahr, M.: Social network analysis for routing in disconnected delay-tolerant MANETs. In: Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), pp. 32–40. ACM Press, Montreal, Quebec, Canada (2007). DOI 10.1145/1288107.1288113
12. Everett, M., Borgatti, S.: Ego network betweenness. Social Networks **27**(1), 31–38 (2005)
13. Freeman, L.: A Set of Measures of Centrality Based on Betweenness. Sociometry **40**(1), 35–41 (1977)
14. Gao, W., Li, Q., Zhao, B., Cao, G.: Multicasting in delay tolerant networks: a social network perspective. In: Proceedings of the 10th ACM International Symposium on Mobile Ad Hoc Networking and Computing(MobiHoc), pp. 299–308 (2009)
15. Hwang, W., Cho, Y., Zhang, A., Ramanathan, M.: Bridging Centrality: Identifying Bridging Nodes in Scale-free Networks. Tech. Rep. 2006-05, Department of Computer Science and Engineering, University at Buffalo (2006)
16. Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., Viennot, L.: Optimized link state routing protocol for ad hoc networks. In: Proceedings of the IEEE International Multi Topic Conference on Technology for the 21st Century (INMIC), pp. 62–68 (2001). DOI 10.1109/INMIC.2001.995315
17. Marsden, P.: Egocentric and sociocentric measures of network centrality. Social Networks **24**(4), 407–422 (2002)
18. Nanda, S.: Mesh-mon: A monitoring and management system for wireless mesh networks. Ph.D. thesis, Dartmouth College, Hanover, NH (2008)
19. Nanda, S., Kotz, D.: Localized bridging centrality for distributed network analysis. In: Proceedings of the 17th International Conference on Computer Communications and Networks (ICCCN), pp. 1–6 (2008)
20. Nanda, S., Kotz, D.: Mesh-Mon: A multi-radio mesh monitoring and management system. Computer Communications **31**(8), 1588–1601 (2008)
21. Nanda, S., Kotz, D.: Social Network Analysis Plugin (SNAP) for Wireless Mesh Networks. In: Proceedings of the 11th IEEE Wireless Communications and Networking Conference (WCNC) (2011)
22. Naval Research Laboratory: Multi-Generator (MGEN). http://cs.itd.nrl.navy.mil/work/mgen/
23. Naval Research Laboratory: NACK-Oriented Reliable Multicast (NORM). http://cs.itd.nrl.navy.mil/work/norm/
24. Tarjan, R.: Depth-First Search and Linear Graph Algorithms. SIAM Journal on Computing **1**, 146 (1972)
25. Tønnesen, A.: Implementing and extending the Optimized Link State Routing protocol. Master's thesis, Master's thesis, University of Oslo, Norway (2004)

Chapter 8
On Throughput Maximization Problem for UWB-Based Sensor Networks via Reformulation–Linearization Technique

Yi Shi, Y. Thomas Hou, and Hanif D. Sherali

Abstract Nonlinear optimization problems (if not convex) are NP-hard in general. One effective approach to develop efficient solutions for these problems is to apply the branch-and-bound (BB) framework. A key step in BB is to obtain a tight linear relaxation for each nonlinear term. In this chapter, we show how to apply a powerful technique, called Reformulation–Linearization Technique (RLT), for this purpose. We consider a throughput maximization problem for an ultra-wideband (UWB)-based sensor network. Given a set of source sensor nodes in the network with each node generating a certain data rate, we want to determine whether or not it is possible to relay all these rates successfully to the base station. We formulate an optimization problem, with joint consideration of physical layer power control, link layer scheduling, and network layer routing. We show how to solve this nonlinear optimization problem by applying RLT and BB. We also use numerical results to demonstrate the efficacy of the proposed solution.

8.1 Introduction

In the first decade of the twenty-first century, there was a flourish of research and development efforts on UWB [17] for military and commercial applications. These applications include tactical handheld and network LPI/D radios, non-LOS LPI/D groundwave communications, precision geolocation systems, high-speed wireless

Y. Shi • Y.T. Hou (✉)
The Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA
e-mail: yshi@vt.edu; thou@vt.edu

H.D. Sherali
The Grado Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA
e-mail: hanifs@vt.edu

LANs, collision avoidance sensors, and intelligent tags, among others. There are some significant benefits of UWB for wireless communications, such as extremely simple design (and thus cost) of radio, large processing gain in the presence of interference, extremely low power spectral density for covert operations, and fine time resolution for accurate position sensing [14, 17].

In this chapter, we consider a UWB-based sensor network for surveillance and monitoring applications. For these network applications, upon an event detection, all sensing data must be relayed to a central data collection point, which we call a base-station. The multi-hop nature of a sensor network introduces some unique challenges. Specifically, due to interference from neighboring links, a change of power level on one link will produce a change in achievable rate in all neighboring links. As a result, the capacity-based routing problem at the network layer is deeply coupled with link layer and physical layer problems such as scheduling and power control. An optimal solution to a network level problem thus must be pursued via a cross-layer approach for such networks.

In this chapter, we study the data collection problem associated with a UWB-based sensor network. For such a network, although the bit rate for each UWB-based sensor node could be high, the total rate that can be collected by the single base-station is limited due to the network resource bottleneck near the base-station as well as interference among the incoming data traffic. Therefore, a fundamental question is the following: *Given a set of source sensor nodes in the network with each node generating a certain data rate, is it possible to relay all these rates successfully to the base-station?*

A naive approach to this problem is to calculate the maximum bit rate that the base station can receive and then perform a simple comparison between this limit with the sum of bit rates produced by the set of source sensor nodes. Indeed, if this limit is exceeded, it is impossible to relay all these rates successfully to the base station. But even if the sum of bit rates generated by source sensor nodes is less than this limit, it may still be infeasible to relay all these rates successfully to the base station. Due to interference and the fact that a node cannot transmit and receive at the same time and in the same band, the actual sum of bit rates that can be relayed to the base station can be substantially smaller than the raw bit rate limit that a base station can receive. Further, such limit is highly dependent upon the network topology, locations of source sensor nodes, bit rates produced by source sensor nodes, and other network parameters. As a result, testing for this feasibility is not trivial and it is important to devise a solution procedure to address this problem.

In this chapter, we study this feasibility problem through a cross-layer optimization approach, with joint consideration of physical layer power control, link layer scheduling, and network layer routing. The link layer scheduling problem deals with how to allocate resources for access among the nodes. Motivated by the work in [10], we consider how to allocate frequency sub-bands, although this approach can also be applied to a time-slot based system. For a total available UWB spectrum of W, we divide it into M sub-bands. For a given M, the scheduling problem considers how to allocate bandwidth to each sub-band and in which sub-bands a node should transmit or receive data. Note that a node cannot transmit and receive within the same sub-band. The physical layer power control problem considers how much power a node should use to transmit data in a particular sub-band. Finally, the routing problem at the network layer considers which set of paths a flow should take from the source sensor node toward the base-station. For optimality, we allow a flow from a source node to be split into sub-flows that take different paths to the base-station.

We formulate this feasibility problem as an optimization problem, which turns out to be a mixed-integer non-polynomial program. To reduce the problem complexity, we modify the integrality and the non-polynomial components in the constraints by exploiting a reformulation technique and the linear property between the rate and SNR, which is unique to UWB. The resulting new optimization problem is then cast into a form of a *non-linear program* (NLP). Since an NLP is NP-hard in general, our specific NLP is likely to be NP-hard, although its formal proof is not given in this chapter. The contribution of this chapter is the development of an approximation solution procedure to this feasibility problem based on a branch-and-bound framework and the powerful Reformulation–Linearization Technique (RLT) [19].

The remainder of the chapter is organized as follows. In Sect. 8.2, we give details of the network model for our problem and discuss its inherent cross-layer nature. Section 8.3 presents a mathematical formulation of the cross-layer optimization problem and a solution procedure based on the branch-and-bound and RLT procedures. In Sect. 8.4, we present numerical results to demonstrate the efficacy of our proposed solution procedure and give insights on the impact of the different optimization components. Section 8.5 reviews related work and Sect. 8.6 concludes this chapter.

8.2 Network Model

We consider a UWB-based sensor network. Although the size of the network (in terms of the number of sensor nodes N) is potentially large, we expect the number of simultaneous source sensor nodes that produce sensing data to be limited. That is, we assume the number of simultaneous events that need to be reported in different part of the network is not large. Nevertheless, the number of nodes involved in relaying (routing) may still be significant due to the limited transmission range of a UWB-based sensor node and the coverage of the network.

Within such a sensor network, there is a base-station (or sink node) to which all collected data from source sensor nodes must be sent. For simplicity, we denote the base-station as node 0 in the network.

Under this network setting, we are interested in answering the following questions:

- Suppose we have a small group of nodes $\mathscr{S}$ that have detected certain events and each of these nodes is generating data. Can we determine if the bit rates from

these source sensor nodes can be successfully sent to the base station (under the capacity limit)?
- If the answer "yes," how should we relay data from each source sensor node to the base station?

Before we further explore this problem, we give the following definition for the feasibility of a rate vector $\mathbf{r}$, where each element, r_i, of the vector corresponds to the rate of a continuous data flow produced by node $i \in \mathscr{S}$.

Definition 8.1. For a given rate vector $\mathbf{r}$ having $r_i > 0$ for $i \in \mathscr{S}$, we say that this rate vector is *feasible* if and only if there exists a solution such that all r_i, $i \in \mathscr{S}$, can be relayed to the base-station.

To determine whether or not a given rate vector $\mathbf{r}$ is feasible, there are several issues at different layers that must be considered. At the network layer, we need to find a multi-hop route (likely multi-paths) from a source node to the sink node. At the link and physical layers, we need to find a scheduling policy and power control for each node such that certain constraints are met satisfactorily. Clearly, this is a cross-layer problem that couples routing, scheduling, and power control. In the rest of this section, we will take a closer look at each problem. Table 8.1 lists notation used in this chapter.

8.2.1 Scheduling

At the link layer, our scheduling problem deals with how to allocate link media for access among the nodes. Motivated by Negi and Rajeswaran's work in [10], we consider how to allocate frequency sub-bands, although this approach can also be applied to time-slot based systems. For the total available UWB spectrum of $W = 7.5$ GHz (from 3.1 GHz to 10.6 GHz), we divide it into M sub-bands. Since the minimum bandwidth of a UWB sub-band is 500 MHz, we have $1 \le M \le 15$. For a given number of total sub-bands M, the scheduling problem considers how to allocate the total spectrum of W into M sub-bands and in which sub-bands a node should transmit or receive data. More formally, for a sub-band m with normalized bandwidth $\lambda^{(m)}$, we have

$$\sum_{m=1}^{M} \lambda^{(m)} = 1$$

and

$$\lambda_{\min} \le \lambda^{(m)} \le \lambda_{\max} \text{ for } 1 \le m \le M,$$

where $\lambda_{\min} = 1/15$ and $\lambda_{\max} = 1 - (M-1) \cdot \lambda_{\min}$.

Table 8.1 Notation

Symbol	Definition
b_{ij}^m	Achievable rate from node i to node j in sub-band m under transmission power p_{ij}^m
$\mathbf{b}$	The vector of b_{ij}^m, $1 \le i \le N, j \in \mathscr{N}_i, 1 \le m \le M$
b_{ij}	Total achievable rate from node i to node j in all sub-bands
f_{ij}	Flow rate from node i to node j
g_{ij}	Propagation gain from node i to node j
g_{jj}	Self-interference parameter at node j
g_{nom}	Propagation gain at a nominal distance
$\mathscr{I}_i$	The set of nodes that can produce interference on node i
K	The feasible scaling factor used in optimization problem formulation
$\mathscr{L}$	The problem list in the branch-and-bound procedure
LB_z	The lower bound of problem z in the branch-and-bound procedure
LB	The global lower bound among all problems in the branch-and-bound procedure
M	Total number of sub-bands for scheduling
N	Total number of sensor nodes in the network
$\mathscr{N}_i$	The set of one-hop neighboring nodes of node i
$p_{\max}$	$= W\pi_{\max}/g_{\text{nom}}$, the power limit
p_{ij}^m	Transmission power used by node i in sub-band m for transmitting data to node j
$\mathbf{p}$	The vector of p_{ij}^m, $1 \le i \le N, j \in \mathscr{N}_i, 1 \le m \le M$
q_j^m	Total power (signal and noise) received by node j in sub-band m
$\mathbf{q}$	The vector for q_j^m, $1 \le j \le N$, $1 \le m \le M$
r_i	Bit rate generated at source sensor node $i \in \mathscr{S}$
$\mathscr{S}$	The set of source sensor nodes in the network
UB_z	The upper bound of problem z in the branch-and-bound procedure
UB	The global upper bound among all problems in the branch-and-bound procedure
W	$= 7.5$ GHz, the entire spectrum for UWB networks
$\lambda^{(m)}$	Normalized length of sub-band m, $\sum_{m=1}^{M} \lambda^{(m)} = 1$.
Λ	The vector of $\lambda^{(m)}$, $1 \le m \le M$
$\lambda_{\min}$	The minimum value of $\lambda^{(m)}$
$\lambda_{\max}$	The maximum value of $\lambda^{(m)}$
η	Power spectral density of ambient Gaussian noise
$\pi_{\max}$	Limit of power spectral density at a node

8.2.2 Power Control

The power control problem considers how much power a node should use in a particular sub-band to transmit data. Denote p_{ij}^m as the transmission power that node i uses in sub-band m for transmitting data to node j. Since a node cannot transmit and receive data within the same sub-band, we have the following constraint: if $p_{ik}^m > 0$ for any node k, then p_{ji}^m must be 0 for each node j.

The power density limit for each node i must satisfy

$$\frac{g_{\text{nom}} \cdot \sum_{j \in \mathscr{N}_i} p_{ij}^m}{W \cdot \lambda^{(m)}} \le \pi_{\max},$$

where $\pi_{\max}$ is the maximum allowed power spatial density, g_{nom} is the gain at some fixed nominal distance $d_{\text{nom}} \geq 1$, and $\mathscr{N}_i$ is the set of one-hop neighboring nodes of node i (under the maximum allowed transmission power). A popular model for gain is

$$g_{ij} = \min\{d_{ij}^{-n}, 1\}, \tag{8.1}$$

where d_{ij} is the distance between nodes i and j and n is the path loss index. Note that the nominal gain should also follow the same propagation gain model (8.1). Thus, we have $g_{ij} = (\frac{d_{\text{nom}}}{d_{ij}})^n g_{\text{nom}}$, when $d_{ij} \geq 1$. Denote

$$p_{\max} = \frac{W \cdot \pi_{\max}}{g_{\text{nom}}}. \tag{8.2}$$

Then the total power that a node i can use at sub-band m must satisfy the following power limit,

$$\sum_{j \in \mathscr{N}_i} p_{ij}^m \leq p_{\max} \lambda^{(m)}. \tag{8.3}$$

Denote $\mathscr{I}_i$ as the set of nodes that can make interference at node i when they use the maximum allowed transmission power. The achievable rate from node i to node j within sub-band m is then

$$b_{ij}^m = W\lambda^{(m)} \cdot \log_2 \left(1 + \frac{g_{ij} \cdot p_{ij}^m}{\eta W \lambda^{(m)} + \sum_{k \in \mathscr{I}_j, l \in \mathscr{N}_k}^{(k,l) \neq (i,j)} g_{kj} p_{kl}^m} \right), \tag{8.4}$$

where η is the ambient Gaussian noise density. Denote b_{ij} as the total achievable rate from node i to node j among all M sub-bands. We have

$$b_{ij} = \sum_{m=1}^{M} b_{ij}^m. \tag{8.5}$$

8.2.3 Routing

The routing problem at the network layer considers the set of paths that a flow takes from the source node toward the base station. For optimality, we allow a flow from a source node to be split into sub-flows and take different paths to the base station. Denote the flow rate from node i to node j as f_{ij}. We have

$$f_{ij} \leq b_{ij},$$
$$\sum_{j \in \mathscr{N}_i} f_{ij} - \sum_{j \in \mathscr{N}_i} f_{ji} = r_i.$$

The constraint $f_{ij} \leq b_{ij}$ says that a flow's bit rate is upper bounded by the achievable rate on this link and the second constraint is for flow balance at node i.

8.3 Feasibility and Solution Procedure

We can develop an upper bound on the maximum rate (denoted as C_0) that the base-station can receive [21]. For a given source rate vector $\mathbf{r}$, where $r_i > 0$ denotes that node i is a source sensor node that produces sensing data at rate r_i, if $\sum_{i=1}^{N} r_i > C_0$, then the rate vector must be infeasible. But $\sum_{i=1}^{N} r_i \leq C_0$ does not guarantee the feasibility of the rate vector $\mathbf{r}$ and further determination is needed. Moreover, if we indeed find that a given rate vector $\mathbf{r}$ is feasible, we also would like to obtain a complete solution that implements $\mathbf{r}$ over the network, i.e., a solution showing the power control, scheduling, and routing for each node.

8.3.1 Rate Feasibility Problem Formulation

Our approach to this feasibility determination problem is to solve an optimization (maximization) problem, which aims to find the optimal power control, scheduling, and routing such that K, called *feasible scaling factor*, is maximized while $K \cdot \mathbf{r}$ is feasible. If the optimal solution yields $K \geq 1$, then the rate vector $\mathbf{r}$ is feasible; otherwise (i.e., $K < 1$), the rate vector $\mathbf{r}$ is infeasible.

Since a node is not allowed to transmit and receive within the same sub-band, we have that if $p_{jl}^m > 0$ for any $l \in \mathcal{N}_j$ then p_{ij}^m should be 0 for all $i \in \mathcal{N}_j$. Mathematically, this property can be formulated as follows. Denote x_j^m ($1 \leq j \leq N$ and $1 \leq m \leq M$) as a binary variable with the following definition: if sub-band m is used for receiving data at node j then $x_j^m = 1$; otherwise, $x_j^m = 0$. Since $\sum_{i \in \mathcal{N}_j} p_{ij}^m \leq |\mathcal{N}_j| p_{\max} \lambda^{(m)}$ and $\sum_{l \in \mathcal{N}_j} p_{jl}^m \leq p_{\max} \lambda^{(m)}$, we have the following constraints, which capture both the constraint that a node j cannot transmit and receive within the same sub-band m and the constraint on the power level.

$$\sum_{i \in \mathcal{N}_j} p_{ij}^m \leq |\mathcal{N}_j| \cdot p_{\max} \cdot \lambda^{(m)} \cdot x_j^m,$$

$$\sum_{l \in \mathcal{N}_j} p_{jl}^m \leq p_{\max} \cdot \lambda^{(m)} \cdot (1 - x_j^m).$$

The rate feasibility problem (RFP) can now be formulated as follows:

8.3.1.1 Rate Feasibility Problem

$$\begin{array}{ll} \text{Maximize} & K \\ \text{subject to} & \displaystyle\sum_{m=1}^{M} \lambda^{(m)} = 1 \end{array}$$

$$\sum_{j\in\mathcal{N}_i} p_{ij}^m - p_{\max}\lambda^{(m)} \le 0 \qquad (1\le i\le N, 1\le m\le M)$$

$$b_{ij}^m = W\lambda^{(m)}\log_2\left(1+\frac{g_{ij}p_{ij}^m}{\eta W\lambda^{(m)} + \sum_{k\in\mathcal{I}_j, l\in\mathcal{N}_k}^{(k,l)\neq(i,j)} g_{kj}p_{kl}^m}\right)$$
$$(1\le i\le N, j\in\mathcal{N}_i, 1\le m\le M)$$

$$\sum_{i\in\mathcal{N}_j} p_{ij}^m \le |\mathcal{N}_j|p_{\max}\lambda^{(m)}x_j^m \qquad (1\le j\le N, 1\le m\le M) \tag{8.6}$$

$$\sum_{l\in\mathcal{N}_j} p_{jl}^m \le p_{\max}\lambda^{(m)}(1-x_j^m) \qquad (1\le j\le N, 1\le m\le M) \tag{8.7}$$

$$\sum_{m=1}^{M} b_{ij}^m - f_{ij} \ge 0 \qquad (1\le i\le N, j\in\mathcal{N}_i)$$

$$\sum_{j\in\mathcal{N}_i} f_{ij} - \sum_{j\in\mathcal{N}_i} f_{ji} - r_iK = 0 \qquad (1\le i\le N)$$

$$\lambda_{\min} \le \lambda^{(m)} \le \lambda_{\max} \qquad (1\le m\le M)$$

$$x_j^m = 0 \text{ or } 1 \qquad (1\le j\le N, 1\le m\le M)$$

$$K, p_{ij}^m, b_{ij}^m, f_{ij} \ge 0 \; (1\le i\le N, j\in\mathcal{N}_i, 1\le m\le M).$$

The formulation for problem RFP is a *mixed-integer non-polynomial program,* which is NP-hard in general [5]. We conjecture that the RFP problem is also NP-hard, although its formal proof is not given in this chapter. Our approach to this problem is as follows. As a first step, we show how to remove the integer (binary) variables and the non-polynomial terms in the RFP problem formulation and reformulate the RFP problem as a non-linear program (NLP). Since an NLP problem remains NP-hard in general, in Sect. 8.3.3, we devise a solution by exploring a branch-and-bound framework and the so-called *Reformulation–Linearization Technique* (RLT) [19].

8.3.2 Reformulation of Integer and Non-Polynomial Constraints

The purpose of integer (binary) variables x_j^m is to capture the fact that a node cannot transmit and receive within the same sub-band, i.e., if a node j transmits data to any node l in a sub-band m, then the data rate that can be received by node j within this sub-band must be 0. Instead of using integer (binary) variables, we use the following approach to achieve the same purpose. We introduce a notion called *self-interference parameter* g_{jj}, with the following property:

$$g_{jj}\cdot p_{jl}^m \gg \eta W\lambda^{(m)}.$$

We incorporate this into the bit rate calculation in (8.4), i.e.,

$$b_{ij}^m = W\lambda^{(m)} \cdot \log_2\left(1 + \frac{g_{ij}p_{ij}^m}{\eta W\lambda^{(m)} + \sum_{k\in\mathscr{I}_j, l\in\mathscr{N}_k}^{(k,l)\neq(i,j)} g_{kj}p_{kl}^m + \sum_{l\in\mathscr{N}_j} g_{jj}p_{jl}^m}\right). \quad (8.8)$$

Thus, when $p_{jl}^m > 0$, i.e., node j is transmitting to some node l, then in (8.8), we have $b_{ij}^m \approx 0$ even if $p_{ij}^m > 0$. In other words, when node j is transmitting to any node l, the achievable rate from node i to node j is *effectively* shut down to 0.

With this new notion of g_{jj}, we can capture the same tramission/receiving behavior of a node without the need of using integer (binary) variables x_j^m as in the RFP formulation. As a result, we can remove constraints (8.6) and (8.7).

To write (8.8) in a more compact form, we re-define $\mathscr{I}_j$ to include node j as long as j is not the base-station node (i.e., node 0). Thus, (8.8) is now in the same form as (8.4). Denote

$$q_j^m = \sum_{k\in\mathscr{I}_j, l\in\mathscr{N}_k} g_{kj}p_{kl}^m. \quad (8.9)$$

Then we have

$$\begin{aligned} b_{ij}^m &= W\lambda^{(m)}\log_2\left(1 + \frac{g_{ij}p_{ij}^m}{\eta W\lambda^{(m)} + \sum_{k\in\mathscr{I}_j, l\in\mathscr{N}_k}^{(k,l)\neq(i,j)} g_{kj}p_{kl}^m}\right) \\ &= W\lambda^{(m)}\log_2\left(1 + \frac{g_{ij}p_{ij}^m}{\eta W\lambda^{(m)} + q_j^m - g_{ij}p_{ij}^m}\right). \end{aligned}$$

To remove the non-polynomial terms, we apply the low SNR property that is unique to UWB [16] and the linearity approximation of the log function, i.e., $\ln(1+x) \approx x$ for $x > 0$ and $x \ll 1$. We have

$$b_{ij}^m \approx \frac{W\lambda^{(m)}}{\ln 2} \cdot \frac{g_{ij}p_{ij}^m}{\eta W\lambda^{(m)} + q_j^m - g_{ij}p_{ij}^m},$$

which is equivalent to

$$\eta W\lambda^{(m)}b_{ij}^m + q_j^m b_{ij}^m - g_{ij}p_{ij}^m b_{ij}^m - \frac{W}{\ln 2} g_{ij}\lambda^{(m)}p_{ij}^m = 0.$$

Finally, without loss of generality, we let $\lambda^{(m)}$ conform the following property.

$$\lambda^{(1)} \leq \lambda^{(2)} \leq \cdots \leq \lambda^{(M)}.$$

Although this additional constraint does not affect the optimal result, it will help speed up the computational time in our algorithm.

With the above re-formulations, we can now re-write the RFP problem as follows:

8.3.2.1 RFP-2

$$
\begin{aligned}
&\text{Maximize} \quad && K \\
&\text{subject to} && \sum_{m=1}^{M} \lambda^{(m)} = 1 \\
& && \lambda^{(m)} - \lambda^{(m-1)} \geq 0 \qquad (2 \leq m \leq M) \\
& && \sum_{j \in \mathcal{N}_i} p_{ij}^m - p_{\max}\lambda^{(m)} \leq 0 \qquad (1 \leq i \leq N, 1 \leq m \leq M) \\
& && \sum_{k \in \mathcal{I}_j, l \in \mathcal{N}_k} g_{kj} p_{kl}^m - q_j^m = 0 \qquad (0 \leq j \leq N, 1 \leq m \leq M) \\
& && \eta W \lambda^{(m)} b_{ij}^m + q_j^m b_{ij}^m - g_{ij} p_{ij}^m b_{ij}^m - \frac{W}{\ln 2} g_{ij} \lambda^{(m)} p_{ij}^m = 0 \\
& && \qquad (1 \leq i \leq N, j \in \mathcal{N}_i, 1 \leq m \leq M) \qquad (8.10) \\
& && \sum_{m=1}^{M} b_{ij}^m - f_{ij} \geq 0 \qquad (1 \leq i \leq N, j \in \mathcal{N}_i) \\
& && \sum_{j \in \mathcal{N}_i} f_{ij} - \sum_{j \in \mathcal{N}_i} f_{ji} - r_i K = 0 \qquad (1 \leq i \leq N) \\
& && K, p_{ij}^m, b_{ij}^m, q_j^m, f_{ij} \geq 0 \; (1 \leq i \leq N, j \in \mathcal{N}_i, 1 \leq m \leq M) \\
& && \lambda^{(1)} \geq \lambda_{\min}, \; \lambda^{(M)} \leq \lambda_{\max}.
\end{aligned}
$$

Although problem RFP-2 is simpler than the original RFP problem, it is still a *non-linear program* (NLP), which remains NP-hard in general [5]. For certain NLP problems, it is possible to find a solution via its Lagrange dual problem. Specifically, if the objective function and constraint functions in the primal problem satisfy suitable *convexity* requirements, then the primal and dual problem have the same optimal objective value [2]. Unfortunately, such a duality-based approach, although attractive, is not applicable to our problem. This is because RFP-2 is a *non-convex* optimization problem (see (8.10)). There is likely a *duality gap* between the objective values of the optimal primal and dual solutions. As a result, a solution approach for the Lagrange dual problem cannot be used to solve our problem. Although there are efforts on solving non-convex optimization problem via a duality (see, e.g., [18] by Rubinov and Yang, where Lagrange-type dual problems are formulated with zero duality gap), we find that the complexity of such an approach is prohibitively high (much higher than the branch-and-bound solution approach proposed in this chapter).

In the next section, we develop a solution procedure based on the branch-and-bound framework [11] and the so-called *Reformulation–Linearization Technique* (RLT) [19, 20] to solve this NLP optimization problem.

8.3.3 A Solution Procedure

8.3.3.1 Branch-and-Bound

Using the branch-and-bound framework, we aim to provide a $(1-\varepsilon)$-optimal solution, where ε is a small pre-defined constant reflecting our tolerance for approximation in the final solution. Initially, we determine suitable intervals for each variable that appears in nonlinear terms. By using a *relaxation technique*, we then obtain an upper bound UB on the objective function value. Although the solution to such a relaxation usually yields infeasibility to the original NLP, we can apply a *local search algorithm* starting from this solution to find a feasible solution to the original NLP. This feasible solution now provides a lower bound LB on the objective function value.

If the distance between the above two bounds is small enough, i.e., $LB \geq (1-\varepsilon)UB$, then we are done with the $(1-\varepsilon)$-optimal solution obtained by the local search. Otherwise, we will use the branch-and-bound framework to find a $(1-\varepsilon)$-optimal solution. The branch-and-bound framework is based on the divide-and-conquer idea. That is, although the original problem is hard to solve, it may be easier to solve a problem with a smaller solution search space, e.g., if we can further limit $\lambda^{(1)} \leq 0.1$. So, we divide the original problem into sub-problems, each with a smaller solution search space. We solve the original problem by solving all these sub-problems. The branch-and-bound framework can remove certain sub-problems before solving them entirely and thus, can provide a solution much faster than a general divide-and-conquer approach.

During the branch-and-bound framework, we put all these sub-problems into a problem list $\mathscr{L}$. Initially, there is only Problem 1 in $\mathscr{L}$, which is the original problem. For each problem in the list, we can obtain an upper bound and a lower bound with a feasible solution, just as we did initially. Then, the global upper bound for all the problems in the list is $UB = \max_{z\in\mathscr{L}}\{UB_z\}$ and the global lower bound for all the problems in the list is $LB = \max_{z\in\mathscr{L}}\{\mathrm{LB}_z\}$. We choose Problem z having the current worst (maximum) upper bound $UB_z = UB$ and then partition this problem into two new Problems z_1 and z_2 that replace Problem z. This partitioning is done by choosing a variable and partitioning the interval of this variable into two new intervals, e.g., $0 \leq \lambda^{(1)} \leq 0.2$ to $0 \leq \lambda^{(1)} \leq 0.1$ and $0.1 \leq \lambda^{(1)} \leq 0.2$. For each new problem created, we obtain an upper bound and a lower bound with a feasible solution. Then we can update both UB and LB.

Once $LB \geq (1-\varepsilon)UB$, the current feasible solution is $(1-\varepsilon)$-optimal and we are done. This is the termination criterion. Otherwise, for any Problem z', if we have $(1-\varepsilon)UB_{z'} < LB$, where UB_z' is the upper bound obtained for Problem z', then we can remove Problem z' from the problem list $\mathscr{L}$ for future consideration. The method then proceeds to the next iteration.

Note that since we are interested in determining whether or not K is greater than or equal to 1 (to check feasibility), we can terminate the branch-and-bound framework if any of the following two cases holds: (1) if the upper bound of K is smaller than 1, then RFP-2 is infeasible or (2) if we find any feasible solution with $K \geq 1$, then RFP-2 is feasible.

8.3.3.2 Relaxation with RLT Technique

Throughout the branch-and-bound framework (both initially and during each iteration), we need a relaxation technique to obtain an upper bound of the objective function. For this purpose, we apply a method based on *Reformulation–Linearization Technique* (RLT) [19, 20], which can provide a linear relaxation for a polynomial NLP problem. Specifically, in (8.10), RLT introduces new variables to replace the inherent polynomial terms and adds linear constraints for these new variables. These new RLT constraints are derived from the intervals of the original variables.

In particular, for nonlinear term $\lambda^{(m)} b_{ij}^m$ in (8.10), we introduce a new variable y_{ij}^m to replace $\lambda^{(m)} b_{ij}^m$. Since $\lambda^{(m)}$ and b_{ij}^m are each bounded by $(\lambda^{(m)})_L \leq \lambda^{(m)} \leq (\lambda^{(m)})_U$ and $(b_{ij}^m)_L \leq b_{ij}^m \leq (b_{ij}^m)_U$, respectively, we have $[\lambda^{(m)} - (\lambda^{(m)})_L] \cdot [b_{ij}^m - (b_{ij}^m)_L] \geq 0$, $[\lambda^{(m)} - (\lambda^{(m)})_L] \cdot [(b_{ij}^m)_U - b_{ij}^m] \geq 0$, $[(\lambda^{(m)})_U - \lambda^{(m)}] \cdot [b_{ij}^m - (b_{ij}^m)_L] \geq 0$, and $[(\lambda^{(m)})_U - \lambda^{(m)}] \cdot [(b_{ij}^m)_U - b_{ij}^m] \geq 0$. From the above relationships and substituting $y_{ij}^m = \lambda^{(m)} b_{ij}^m$, we have the following RLT constraints for y_{ij}^m.

$$\begin{aligned}
(\lambda^{(m)})_L \cdot b_{ij}^m + (b_{ij}^m)_L \cdot \lambda^{(m)} - y_{ij}^m &\leq (\lambda^{(m)})_L \cdot (b_{ij}^m)_L \\
(\lambda^{(m)})_U \cdot b_{ij}^m + (b_{ij}^m)_L \cdot \lambda^{(m)} - y_{ij}^m &\geq (\lambda^{(m)})_U \cdot (b_{ij}^m)_L \\
(\lambda^{(m)})_L \cdot b_{ij}^m + (b_{ij}^m)_U \cdot \lambda^{(m)} - y_{ij}^m &\geq (\lambda^{(m)})_L \cdot (b_{ij}^m)_U \\
(\lambda^{(m)})_U \cdot b_{ij}^m + (b_{ij}^m)_U \cdot \lambda^{(m)} - y_{ij}^m &\leq (\lambda^{(m)})_U \cdot (b_{ij}^m)_U .
\end{aligned}$$

We, therefore, replace $\lambda^{(m)} b_{ij}^m$ with y_{ij}^m in (8.10) and add the above RLT constraints for y_{ij}^m into the RFP-2 problem formulation. Similarly, we let $u_{ij}^m = q_j^m b_{ij}^m$, $v_{ij}^m = p_{ij}^m b_{ij}^m$, and $w_{ij}^m = \lambda^{(m)} p_{ij}^m$. From $(p_{ij}^m)_L \leq p_{ij}^m \leq (p_{ij}^m)_U$ and $(q_j^m)_L \leq q_j^m \leq (q_j^m)_U$, we can obtain the RLT constraints for u_{ij}^m, v_{ij}^m, and w_{ij}^m as well.

Denote Λ, $\mathbf{p}$, $\mathbf{b}$, and $\mathbf{q}$ as vectors for $\lambda^{(m)}$, p_{ij}^m, b_{ij}^m, and q_j^m, respectively. After we replace all non-linear terms as above and add the corresponding RLT constraints into the RFP-2 problem formulation, we obtain the following LP.

$$\begin{aligned}
\text{Maximize} \quad & K \\
\text{subject to} \quad & \textstyle\sum_{m=1}^{M} \lambda^{(m)} = 1 \\
& \lambda^{(m)} - \lambda^{(m-1)} \geq 0 \qquad (2 \leq m \leq M) \\
& \textstyle\sum_{j \in \mathcal{N}_i} p_{ij}^m - p_{\max} \lambda^{(m)} \leq 0 \ (1 \leq i \leq N, 1 \leq m \leq M)
\end{aligned}$$

$$\begin{array}{ll}
\sum_{k\in\mathscr{I}_j, l\in\mathscr{N}_k} g_{kj} p_{kl}^m - q_j^m = 0 & (0 \le j \le N, 1 \le m \le M) \\
\eta W y_{ij}^m + u_{ij}^m - g_{ij} v_{ij}^m - \frac{W}{\ln 2} g_{ij} w_{ij}^m = 0 & (1 \le i \le N, j \in \mathscr{N}_i, 1 \le m \le M) \\
\text{RLT constraints for } y_{ij}^m, u_{ij}^m, v_{ij}^m, \text{ and } w_{ij}^m & (1 \le i \le N, j \in \mathscr{N}_i, 1 \le m \le M) \\
\sum_{m=1}^M b_{ij}^m - f_{ij} \ge 0 & (1 \le i \le N, j \in \mathscr{N}_i) \\
\sum_{j\in\mathscr{N}_i} f_{ij} - \sum_{j\in\mathscr{N}_i} f_{ji} - r_i K = 0 & (1 \le i \le N) \\
K, f_{ij}, y_{ij}^m, u_{ij}^m, v_{ij}^m, w_{ij}^m \ge 0 & (1 \le i \le N, j \in \mathscr{N}_i, 1 \le m \le M) \\
(\Lambda, \mathbf{p}, \mathbf{b}, \mathbf{q}) \in \Omega, &
\end{array}$$

where $\Omega = \{(\Lambda, \mathbf{p}, \mathbf{b}, \mathbf{q}) : (\lambda^{(m)})_L \le \lambda^{(m)} \le (\lambda^{(m)})_U, (p_{ij}^m)_L \le p_{ij}^m \le (p_{ij}^m)_U, (b_{ij}^m)_L \le b_{ij}^m \le (b_{ij}^m)_U, (q_j^m)_L \le q_j^m \le (q_j^m)_U\}$.

The details of the proposed branch-and-bound solution procedure with RLT are given in Fig. 8.1. Note that in Step 14 of the Feasibility Check Algorithm, the method chooses a partitioning variable based on the maximum relaxation error. Clearly, $\lambda^{(m)}$ is a key variable in the problem formulation. As a result, the algorithm will run much more efficiently if we give the highest priority to $\lambda^{(m)}$ when it comes to choosing a partitioning variable.[1]

8.3.3.3 Local Search Algorithm

In the branch-and-bound framework, we need to find a solution to the original problem from the solution to the relaxation problem (see Step 9 in Fig. 8.1). In particular, we need to obtain a feasible solution from $\hat{\Lambda}$ and $\hat{\mathbf{p}}$. We now show how to obtain such a feasible solution.

We can let $\Lambda = \hat{\Lambda}$. Note that in RFP-2, we introduced the notion of a self-interference parameter to remove the binary variables in RFP. Then in $\hat{\mathbf{p}}$, it is possible that $p_{il}^m > 0$ and $p_{ji}^m > 0$ for a certain node i within some sub-band m. Therefore, it is necessary to find a new $\mathbf{p}$ from $\hat{\mathbf{p}}$ such that no node is allowed to transmit and receive within the same sub-band. An algorithm that achieves this purpose is shown in Fig. 8.2. The basic idea is to split the total bandwidth used at node i into two groups of equal bandwidth: one group for transmission and the other group for receiving.

After we obtain Λ and $\mathbf{p}$ by the algorithm in Fig. 8.2, constraints on sub-band division, scheduling, and power control are all satisfied. Now we can compute b_{ij} from (8.4) and (8.5). Then, we solve the following simple LP for K.

[1]In our implementation of the algorithm, we give the highest priority to $\lambda^{(m)}$, the second highest priority to p_{ij}^m, and consider q_j^m last when we choose a partitioning variable. This does not hamper the convergence property of the algorithm [19].

```
    **Feasibility Check Algorithm**
1.  Initialize()
    {
2.      Let the lower bound $LB = -\infty$ and the initial problem list $\mathscr{L}$ include only the
        original problem, denoted as Problem 1.
3.      The initial value sets for $(\Lambda, \mathbf{p}, \mathbf{b}, \mathbf{q})$ are $\lambda_{min} \leq \lambda^{(m)} \leq \lambda_{max}, 0 \leq p_{ij}^m \leq p_{max} \cdot \lambda_{max}$,
        $0 \leq b_{ij}^m \leq \frac{g_{ij}}{\eta \ln 2} p_{max} \lambda_{max}$, and $0 \leq q_j^m \leq p_{max} \lambda_{max} \sum_{k \in \mathscr{I}_j} g_{kj}$.
4.      Solve the RLT relaxation for Problem 1 and obtain its solution $(\hat{\Lambda}, \hat{\mathbf{p}}, \hat{\mathbf{b}}, \hat{\mathbf{q}})$.
5.      The objective value of the solution is an upper bound $UB_1$ to problem 1.
    }
6.  MainIteration()
    {
7.      Select Problem $z$ that has the maximum $UB_z$ among all problems in list $\mathscr{L}$.
8.      Update the global upper bound $UB = UB_z$.
9.      Find a feasible solution $\psi_z$ from $\hat{\Lambda}$ and $\hat{\mathbf{p}}$ via a local search algorithm and denote
        its objective value as $LB_z$.
10.     If $(LB_z > LB)$ {
11.         Update $\psi^* = \psi$ and $LB = LB_z$.
12.         If $(LB \geq (1-\varepsilon)UB)$, we stop with the $(1-\varepsilon)$-optimal solution $\psi^*$.
13.         Otherwise, remove each Problem $z'$ with $(1-\varepsilon)UB_{z'} \leq LB$ from list $\mathscr{L}$. }
14.     Find the maximum relaxation error among $|\hat{\lambda}^{(m)} \hat{b}_{ij}^m - \hat{y}_{ij}^m|, |\hat{q}_j^m \hat{b}_{ij}^m - \hat{u}_{ij}^m|$,
        $|\hat{p}_{ij}^m \hat{b}_{ij}^m - \hat{v}_{ij}^m|$, and $|\hat{\lambda}^{(m)} \hat{p}_{ij}^m - \hat{w}_{ij}^m|$, for $1 \leq i \leq N, j \in \mathscr{N}_i, 1 \leq m \leq M$.
15.     In the case that the maximum relaxation error is $|\hat{\lambda}^{(m)} \hat{b}_{ij}^m - \hat{y}_{ij}^m|$,
16.         if $[(\lambda^{(m)})_U - (\lambda^{(m)})_L] \cdot \min\{\hat{\lambda}^{(m)} - (\lambda^{(m)})_L, (\lambda^{(m)})_U - \hat{\lambda}^{(m)}\} \geq$
            $[(b_{ij}^m)_U - (b_{ij}^m)_L] \cdot \min\{\hat{b}_{ij}^m - (b_{ij}^m)_L, (b_{ij}^m)_U - \hat{b}_{ij}^m\}$, we partition $\Omega_z$ into two
            new regions $\Omega_{z1}$ and $\Omega_{z2}$ by dividing $[(\lambda^{(m)})_L, (\lambda^{(m)})_U]$ into $[(\lambda^{(m)})_L, \hat{\lambda}^{(m)}]$
            and $[\hat{\lambda}^{(m)}, (\lambda^{(m)})_U]$.
17.         Otherwise, we partition $\Omega_z$ into two new regions by dividing $[(b_{ij}^m)_L, (b_{ij}^m)_U]$
            into $[(b_{ij}^m)_L, \hat{b}_{ij}^m]$ and $[\hat{b}_{ij}^m, (b_{ij}^m)_U]$.
18.     Similarly, we can perform a corresponding partition if the maximum relaxation
        error is $|\hat{q}_j^m \hat{b}_{ij}^m - \hat{u}_{ij}^m|$, $|\hat{p}_{ij}^m \hat{b}_{ij}^m - \hat{v}_{ij}^m|$, or $|\hat{\lambda}^{(m)} \hat{p}_{ij}^m - \hat{w}_{ij}^m|$.
19.     Solve the RLT relaxation for Problems $z_1$ and $z_2$ and obtain their upper bounds
        $UB_{z1}$ and $UB_{z2}$.
20.     Remove Problem $z$ from the problem list $\mathscr{L}$.
21.     If $(1-\varepsilon)UB_{z1} > LB$, add Problem $z_1$ into the problem list $\mathscr{L}$.
22.     If $(1-\varepsilon)UB_{z2} > LB$, add Problem $z_2$ into the problem list $\mathscr{L}$.
23.     If $\mathscr{L} = \emptyset$, stop with the $(1-\varepsilon)$-optimal solution $\psi^*$.
24.     Otherwise, go to Step 7 for the next iteration.
    }
```

Fig. 8.1 A solution procedure to the RFP-2 problem based on branch-and-bound and RLT

$$
\begin{array}{lll}
\text{Maximize} & K & \\
\text{subject to} & f_{ij} \leq b_{ij} & (1 \leq i \leq N, j \in \mathscr{N}_i) \\
 & \sum_{j \in \mathscr{N}_i} f_{ij} - \sum_{j \in \mathscr{N}_i} f_{ji} - r_i K = 0 & (1 \leq i \leq N) \\
 & K, f_{ij} \geq 0 & (1 \leq i \leq N, j \in \mathscr{N}_i).
\end{array}
$$

If an LP solution provides a $K \geq 1$, then this rate vector $\mathbf{r}$ is feasible.

```
    Power Update Algorithm
1.  Choose a node $i$ that meets one of the following requirements. If no such node exists, we
    are done and the updated power vector is $\mathbf{p}$.
2.      First, identify a node such that all nodes that receive data from this node already
        have their transmission power updated.
3.      If no such node exists, choose the node among all nodes that do not have their
        transmission power updated and is closest to the base-station.
4.  If there is no sub-band used for both transmission and receiving at node $i$ under $\hat{\mathbf{p}}$, then
    do not update its power and go to Step 1.
5.  Otherwise, define $\Gamma_{out}$ as the total bandwidth used by node $i$ for transmitting data and
    $\Gamma_{in-out}$ as the total bandwidth used by node $i$ only for receiving data.
6.  If $\Gamma_{out} > \Gamma_{in-out}$, node $i$ tries to release some sub-bands used for both transmission and
    receiving under $\hat{\mathbf{p}}$ and reduce the total used bandwidth to $(\Gamma_{out} + \Gamma_{in-out})/2$ in the
    following order.
7.      First it releases sub-band $m$ with $\sum_{j\in\mathcal{N}_i}(p_{ij}^m)_L = 0$, in non-decreasing order of
        $\sum_{j\in\mathcal{N}_i} p_{ij}^m$.
8.      Second it reduces the transmission power in sub-band $m$, i.e., from $p_{ij}^m$ to $(p_{ij}^m)_L$, in
        non-decreasing order of $\sum_{j\in\mathcal{N}_i}(p_{ij}^m)_L$.
9.  If node $i$ decides to use a sub-band, then all nodes should not use this sub-band to transmit
    data to node $i$.
10. Go to Step 1.
```

Fig. 8.2 An algorithm to obtain $\mathbf{p}$ from $\hat{\mathbf{p}}$

8.4 Numerical Results

8.4.1 Simulation Setting

In this section, we present numerical results for our solution procedure and compare it to other possible approaches. Given that the total UWB spectrum is $W = 7.5$ GHz and that each sub-band is at least 500 MHz, we have that the maximum number of sub-bands is $M = 15$. The gain model for a link (i, j) is $g_{ij} = \min(d_{ij}^{-2}, 1)$ and the nominal gain is chosen as $g_{\text{nom}} = 0.02$. The power density limit $\pi_{\max}$ is assumed to be 1% of the white noise η [16].

We consider a randomly generated network of 100 nodes (see Fig. 8.3) over a 50×50 area, where the distance is based on normalized length in (8.1). The base-station is located at the origin. The details for this network will be elaborated shortly when we present the results.

We investigate the impact of scheduling and routing. We are interested in comparing a cross-layer approach to a decoupled approach to our problem.

8.4.2 Impact of Scheduling

For the 100-node network shown in Fig. 8.3, there are eight source sensor nodes (marked as stars) in the network. The randomly generated data rate are $r_1 = 5$,

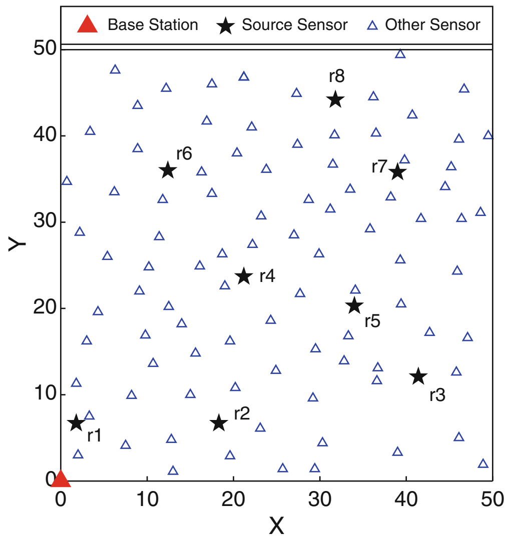


Fig. 8.3 Network topology for a 100-node network

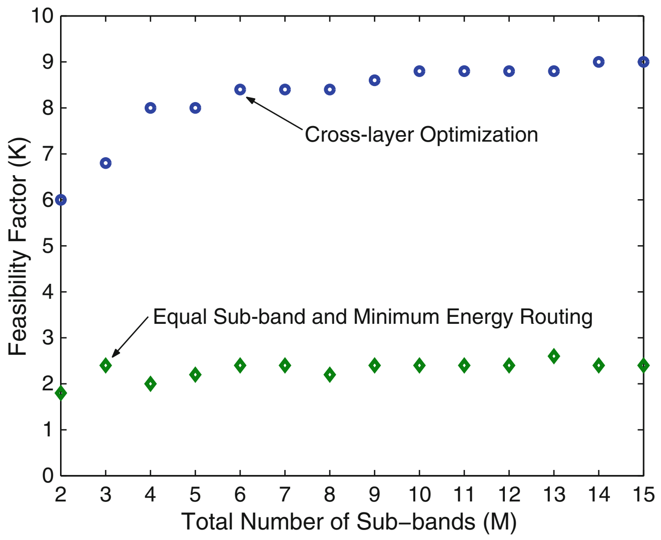


Fig. 8.4 The maximum achievable K as a function of M for the 100-node network

$r_2 = 2$, $r_3 = 2$, $r_4 = 4$, $r_5 = 5$, $r_6 = 3$, $r_7 = 3$, and $r_8 = 1$, with units defined in an appropriate manner. To show performance limits, we consider whether the network can transmit $K \cdot r_i$ from source sensor node i to the base station and investigate the maximum K (feasible scaling factor) under different approaches. Figure 8.4 (upper curve) shows the maximum achievable K for different M under our solution procedure. Clearly, K is a non-decreasing function of M, which states

Table 8.2 Performance of feasible scaling factor K under different spectrum allocations with $M = 5$ for the 100-node network

Spectrum allocation	K	Rate
Optimal: (0.4256, 0.2339, 0.1660 0.1066 0.0679)	8.0	200
Equal: (0.20, 0.20, 0.20, 0.20, 0.20)	4.2	105
Random 1: (0.36, 0.23, 0.20, 0.11, 0.10)	2.8	70
Random 2: (0.27, 0.24, 0.21, 0.17, 0.11)	4.2	105

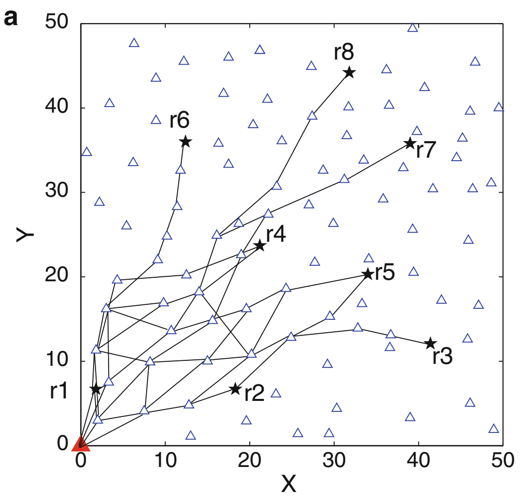
that the more sub-bands available, the larger traffic volume that the network can support. The physical explanation for this is that the more sub-bands available, the more opportunity for each node to avoid interference from other nodes within the same sub-band, and thus more throughput in the network. Also, note that there is a noticeable increase in K when M is small. But when $M \geq 4$, the increase in K is no longer significant. This suggests that for simplicity, we could just choose a small value (e.g., $M = 5$) for the number of sub-bands instead of the maximum $M = 15$.

To show the importance of joint optimization of physical layer power control, link layer scheduling, and network layer routing, in Fig. 8.4, we also plot K as a function of M for a pre-defined routing strategy, namely, the minimum-energy routing with equal sub-band scheduling. Here, the energy cost is defined as g_{ij}^{-1} for link (i, j). Under this approach, we find a minimum-energy path for each source sensor node and determine which sub-band to use for each link and with how much power. When a node cannot find a feasible solution to transmit data to the next hop, it declares that the given rate vector is infeasible. In Fig. 8.4, we find that the minimum-energy routing with equal sub-band scheduling approach is significantly inferior than the proposed cross-layer optimization approach.

Table 8.2 shows the results for K under different spectrum allocations for $M = 5$. The routes are the same as those obtained under optimal routing from our cross-layer optimal solution (see Fig. 8.5 (a)) and are fixed in this study. The first optimal spectrum allocation is obtained from the cross-layer optimal solution. The second is an equal spectrum allocation and the following two are random spectrum allocations. Clearly, the cross-layer optimal spectrum allocation provides the best performance among all these spectrum allocations. It is important to realize that in addition to the number of sub-bands M, the way how the spectrum is allocated for a given M also has a profound impact on the performance. In Table 8.3, we perform the same study for $M = 10$ and obtain the same conclusion.

8.4.3 Impact of Routing

We study the impact of routing on our cross-layer optimization problem under a given optimal schedule (obtained through our solution procedure). Table 8.4 shows the results in this study. In addition to our cross-layer optimal routing, we also



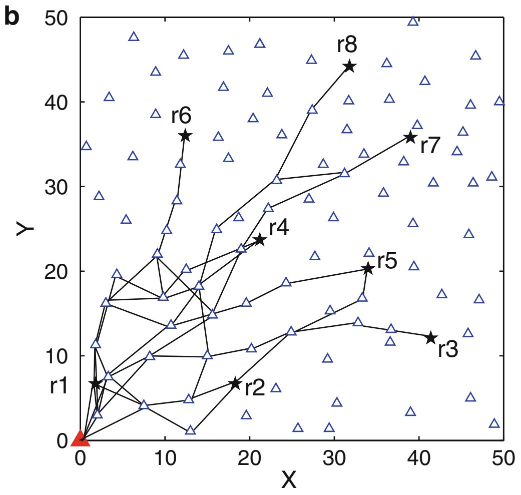


Fig. 8.5 Optimal routing for the 100-node network. (**a**) $M = 5$, (**b**) $M = 10$

consider the following two routing approaches, namely, minimum-energy routing and minimum-hop routing. The minimum-hop routing is similar to the minimum-energy routing, except the cost here is measured in the number of hops.

In Table 8.4, the spectrum allocation is chosen as the optimal spectrum allocation from our cross-layer optimal solution (see Tables 8.2 and 8.3) and is fixed. Clearly, the cross-layer optimal routing outperforms both minimum-energy and minimum-hop routing approaches. Both minimum-energy routing and minimum-hop routing are minimum-cost routing (with different link cost). Minimum-cost routing only uses a single-path, i.e., multi-path routing is not allowed, which is not likely to provide an optimal solution. Moreover, it is very likely that multiple source sensors share a "good" path. Thus, the rates for these sensors are bounded by the achievable

Table 8.3 Performance of feasible scaling factor K under different spectrum allocations with $M = 10$ for the 100-node network

Spectrum allocation	K	Rate
Optimal: (0.1551, 0.1365, 0.1283, 0.0962, 0.0952, 0.0916, 0.0901, 0.0702, 0.0689, 0.0679)	8.8	220
Equal: (0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10)	3.6	90
Random 1: (0.14, 0.13, 0.12, 0.11, 0.09, 0.09, 0.09, 0.08, 0.08, 0.07)	4.0	100
Random 2: (0.17, 0.13, 0.11, 0.10, 0.10, 0.09, 0.08, 0.08, 0.07, 0.07)	3.8	95

Table 8.4 Performance of feasible scaling factor K under different routing strategies for the 100-node network

	$M = 5$		$M = 10$	
Routing Strategy	K	Rate	K	Rate
Optimal Routing	8.0	200	8.8	220
Minimum-Energy Routing	2.2	55	2.4	60
Minimum-Hop Routing	1.4	35	2.0	50

rate of this path. Further, minimum-hop routing has its own problem. Minimum-hop routing prefers small number of hops (with a long distance on each hop) toward the destination node. Clearly, a long-distance hop will reduce its corresponding link's achievable rate, due to the distance gain factor.

8.5 Related Work

A good overview paper on UWB is given in [14]. Physical layer issues associated with UWB-based multiple access communications can be found in [3, 7, 8, 12, 22] and references therein. In this section, we focus on related work addressing networking problems with UWB.

In [9], Negi and Rajeswaran first showed that, in contrast to previously published results, the throughput for UWB-based ad hoc networks increases with node density. This important result is mainly due to the large bandwidth and the ability of power and rate adaptation of UWB-based nodes, which alleviate interference. More importantly, this result demonstrates the significance of physical layer properties on network layer metrics such as network capacity. In [1], Baldi et al. considered the admission control problem based on a flexible cost function in UWB-based networks. Under their approach, a communication cost is attached to each path and the cost of a path is the sum of costs associated with the links it comprises. An admissibility test is then made based on the cost of a path. However, there is no explicit consideration of joint cross-layer optimization of power control, scheduling, and routing in this admissibility test. In [4], Cuomo et al. studied a multiple access scheme for UWB. Power control and rate allocation problems were formulated for both elastic bandwidth data traffic and guaranteed-service traffic. The impact of routing, however, was not addressed.

The most closely related research to our work are [10] and [15]. In [10], Negi and Rajeswaran studied how to maximize proportional rate allocation in a single-hop UWB network (each node can communicate to any other node in a single hop). The problem was formulated as a cross-layer optimization problem with similar scheduling and power control constraints as in this chapter. In contrast, our focus in this chapter is on a feasibility test for a rate vector in a sensor network and we consider a multi-hop network environment where routing is also part of the cross-layer optimization problem. As a result, the problem in this chapter is more difficult. In [15], Radunovic and Le Boudec studied how to maximize the total log-utility of flow rates in multi-hop ad hoc networks. The cross-layer optimization space consists of scheduling, power control, and routing. As the optimization problem is NP-hard, the authors then studied a simple ring network as well as a small-sized network with pre-defined scheduling and routing policies. On the other hand, in this chapter, we have developed a novel solution procedure to our cross-layer optimization problem.

8.6 Conclusion

In this chapter, we studied the important problem of routing data traffic in a UWB-based sensor network. We followed a cross-layer optimization approach with joint consideration of physical layer power control, link layer scheduling, and network layer routing. We developed a solution procedure based on the branch-and-bound framework and the RLT technique. Our numerical results demonstrated the efficacy of our proposed solution procedure and substantiated the importance of cross-layer optimization for UWB-based sensor networks.

References

1. P. Baldi, L. De Nardis, and M.-G. Di Benedetto, “Modeling and optimization of UWB communication networks through a flexible cost function,” *IEEE Journal on Selected Areas in Communications,* vol. 20, no. 9, pp. 1733–1744, December 2002.
2. M.S. Bazaraa, H.D. Sherali, and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms,* second edition, John Wiley & Sons, Inc., New York, NY, 1993.
3. L.X. Cai, L. Cai, X. Shen, J.W. Mark, and Q. Zhang, “MAC protocol design and optimization for multi-hop ultra-wideband networks,” *IEEE Transactions on Wireless Communications,* vol. 8, no. 8, pp. 4056–4065, August 2009.
4. F. Cuomo, C. Martello, A. Baiocchi, and F. Capriotti, “Radio resource sharing for ad hoc networking with UWB,” *IEEE Journal on Selected Areas in Communications,* vol. 20, no. 9, pp. 1722–1732, December 2002.

5. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness, W. H. Freeman and Company,* pp. 245–248, New York, NY, 1979.
6. A. Goldsmith and S.B Wicker, "Design challenges for energy-constrained ad hoc wireless networks," *IEEE Wireless Communications,* vol. 9, pp. 8–27, August 2002.
7. IEEE 802.15 WPAN High Rate Alternative PHY Task Group 3a, http://www.ieee802.org/15/pub/TG3a.html.
8. *IEEE Journal on Selected Areas in Communications – Special Issue on Ultra-Wideband Radio in Multiaccess Wireless Communications,* Guest Editors: N. Blefari-Melazzi, M.G. Di Benedettio, M. Geria, H. Luediger, M.Z. Win, and P. Withington, vol. 20, no. 9, December 2002.
9. A. Rajeswaran and R. Negi, "Capacity of ultra wide band wireless ad hoc networks," *IEEE Transactions on Wireless Communications,* vol. 6, no. 10, pp. 3816–3824, October 2007.
10. A. Rajeswaran, G. Kim, and R. Negi, "Joint power adaptation, scheduling and routing for ultra wide band networks," *IEEE Transactions on Wireless Communications,* vol. 6, no. 5, pp. 1964–1972, May 2007.
11. G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization,* John Wiley & Sons, pp. 354–367, New York, NY, 1999.
12. R. Pilakkat and L. Jacob, "Scheduling and power control for MAC layer design in multihop IR-UWB networks," *International Journal of Network Management,* vol. 20, issue 1, pp. 1–19, January 2010.
13. D. Porcino, "Ultra-wideband radio technology: potential and challenges ahead," *IEEE Communications Magazine,* pp. 66–74, July 2003.
14. R.C. Qiu, H. Liu, and X. Shen, "Ultra-wideband for multiple access communications," *IEEE Communications Magazine,* pp. 80–87, February 2005.
15. B. Radunovic and J.-Y. Le Boudec, "Optimal power control, scheduling, and routing in UWB networks," *IEEE Journal on Selected Areas in Communications,* vol. 22, no. 7, pp. 1252–1270, September 2004.
16. A. Rajeswaran, G. Kim, and R. Negi, "A scheduling framework for UWB & cellular networks," *Springer Mobile Networks and Applications (MONET),* vol. 11, no. 1, pp. 9–20, 2006.
17. J.H. Reed, *An Introduction to Ultra Wideband Communication Systems,* Prentice Hall, 2005.
18. A. Rubinov and X. Yang, *Lagrange-type Functions in Constrained Non-convex Optimization,* Kluwer Academic Publishers, Norwell, MA, 2003.
19. H.D. Sherali and W.P. Adams, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems,* Kluwer Academic Publishers, Dordrecht/Boston/London, Chapter 8, 1999.
20. H.D. Sherali, "Tight relaxations for nonconvex optimization problems using the reformulation-linearization/convexification technique (RLT)," *Handbook of Global Optimization, Volume 2: Heuristic Approaches,* eds. P.M. Pardalos and H.E. Romeijn, Kluwer Academic Publishers, Dordrecht/London/Boston, pp. 1–63, 2002.
21. Y. Shi and Y.T. Hou, "On the capacity of UWB-based wireless sensor network," *Elsevier Computer Networks Journal,* vol. 52, issue 14, pp. 2797–2804, October 2008.
22. M. Win and R. Scholtz, "Ultra-wide bandwidth time-hopping spread-spectrum impulse radio for wireless multiple-access communications," *IEEE Transactions on Communications,* vol. 48, pp. 679–691, April 2000.

Chapter 9
A Parallel Routing Algorithm for Traffic Optimization

M.L. Wang, K.H. Yeung, and F. Yan

Abstract This chapter applies the general complex network theory to study a parallel routing algorithm called *Classified Traffic Routing (CTR)* for traffic optimization. The parallel routing algorithm involves two routing tables when forwarding packets. For *CTR*, performance analysis is performed which focuses on loss, delay, and energy in a scale-free network. Its performance is also compared to those of single routing algorithms. For existing two main kinds of single routing algorithms - shortest path first and congestion avoidance first algorithms, we select one representative algorithm from each kind for comparison. The result shows that good loss or delay performance (but not both) can be obtained for each representative routing algorithm, namely *Shortest Path First (SPF)* algorithm and *Optimal Routing (OR)* algorithm. The chapter then discusses a study on energy performance of these two algorithms. The results show that the two algorithms have very different performance on average energy consumption and on distribution of energy consumption among all nodes. This chapter then argues that single routing algorithm could not meet the requirements of different types of traffic while could not balance the energy consumption. In order to provide good loss performance for loss-sensitive traffic and good delay performance for delay-sensitive traffic, and in consideration of energy consumption, forwarding packets with *CTR* is a good choice. Simulation results show that *CTR* can give a much more balanced performance on loss, delay, and energy than those of *SPF* and *OR*.

9.1 Introduction

Many researchers have focused on exploring how the structure features play a role in the dynamic behaviors of networks [1]. These theoretical modeling of networks has

M.L. Wang • K.H. Yeung (✉) • F. Yan
Department of Electrical Engineering, City University of Hong Kong, Tat Chee Ave, Hong Kong
e-mail: mablewml@gmail.com; eeayeung@cityu.edu.hk; yanfan84@gmail.com

been applied to various fields, such as the social groups [2], the Internet [3–5] and the world wide web [6], metabolism [7, 8], and ecosystems [9, 10]. In all of these models, how a network handles and delivers information is one of the important issues to study. As all the mentioned networks are complex networks with scale-free property, the scale-free network model is usually used for studies.

In the studies of transport optimization, two main different kinds of approaches have been considered. One of them takes the shortest paths [11, 12], and for the other one, researchers consider the congestion problem on nodes (or links) and try to avoid congestion by not using the shortest paths. In the shortest path approach, the length of a path is the total weights of the links in the path [11–14]. This kind of approaches has the minimum packet delay but they cause many of the shortest paths going through a set of nodes, which lead to node congestion. Especially in a network with high traffic density, congestion on these nodes is easily found. The design is therefore not suitable for traffic which requires high throughput, such as TCP. In the second kind of approaches, the algorithms work on avoiding or weakening the congestion in the network, so the transportation capacity of the network will be improved [15–17]. When forwarding packets, paths with idle or not busy nodes are preferred rather than shortest paths with serious congestion. In these algorithms, queuing time will be saved and meanwhile, the number of packets dropped due to limited buffer sizes (of nodes) is decreased. However, this kind of algorithms has the disadvantage that too many routing paths are not the shortest paths and so, the average packet delay is longer. For traffic in which delay is a major concern of communication quality, this approach cannot give satisfactory performance. Therefore, routing algorithms discussed above will not give good performance when coexistence of VoIP and TCP is found [18].

Besides delay and loss performance, there is another important (but less studied) performance: the energy performance in transport optimization. The total number of Internet users now is beyond 2.09 billions [19] and keeps increasing in a rapid speed. Nowadays, the electricity usage for Internet is 0.5%–1% of the total world electricity usage [20, 21]. And if there are 2 billion people having broadband access with 1 Mb/s, then this percentage will increase to 5%. Even worse, if there are 20 billion people having broadband access with 1 Mb/s, the percentage becomes 50%. Some researchers concluded that the constraint on Internet growth will be energy consumption in the future [22]. For a long-term thinking, we must consider energy consumption when developing good routing algorithms.

In this chapter, traffic optimization based on traffic types on a scale-free network is studied. We first discuss the loss, delay and energy performance of two representative routing algorithms, namely Shortest Path First (SPF) and Optimized Routing (OR). Each of them belongs to either routing approach as mentioned above. We then argue that they cannot meet the requirements on the performance of loss, delay or energy at the same time. Compared to single routing algorithm, parallel routing algorithms therefore have their own advantages. A parallel routing algorithm called Classified Traffic routing (CTR) is therefore proposed. Simulation results show that CTR can give a more optimized performance on loss, delay and energy when compared with those of SPF and OR.

9.2 Shortest Path First and Optimized Routing

SPF can be implemented in two ways: either finding the shortest path dynamically or following a fixed routing table. In the first case, when a new packet is generated, the router searches for a shortest path among all shortest paths between its source and destination [23]; in the second one, for any pair of nodes, the shortest paths are saved in a fixed routing table [24]. When the delivering capability for each node, denoted as C_i, is the same for all i, the packet generated rate in the whole network cannot go beyond a critical point R_c. Below this critical value, the generated packets can be processed properly and the packets queueing in buffers would not be accumulated with time. However, when the network generates too many packets per time slot (beyond Rc), packets queuing in buffers will be accumulated from time to time, which means the network is overloading. This value for a heterogeneous network is much smaller than that of a homogeneous network (using SPF and under the same condition). The throughput of a regular network is also much larger than that of a scale-free network [25, 26]. The explanation is as follow: all packets are routed just with the shortest paths, so packets are more easily to be forwarded passing some "hubs" than other nodes, and congestion occurs in these "hubs". The congestion will immediately spread over the whole network.

OR [16] tries to distribute traffic to every node as uniformly as possible by uniforming the betweenness [18] of all nodes. For a given routing table, the betweenness for a specified node i is defined as: the sum of all fractional paths that pass through that node i, and $B_{\max}$ specifies the largest betweenness for any node in the network [27]. To distribute traffic uniformly to the whole network, the betweenness of all nodes would be uniformed. In order to achieve it, minimizing $B_{\max}$ is necessary so that no nodes will be too popular, and it has been shown that significant improvement in the transport capacity of a network can be achieved by minimizing the maximum betweenness of the network [16]. For a network with N nodes, on average, each time step the number of packets forwarded to node i is $\frac{RB_i}{N(N-1)}$ where R is the average packet generated rate by one node. It identifies that by reducing the maximum betweenness in the network, the throughput will be improved. In OR algorithm, the maximum betweenness is reduced. In each time step, the node with largest betweenness in the network is found, and then the lengths of all the links connected to it are increased. After that, just recalculate the betweennesses for all nodes and repeat the same steps for several times. Finally, we can obtain a routing table whose maximum betweenness for the whole network is very close to the average betweenness. The maximum betweenness is usually quite low in this case. It means that the traffic is distributed more uniformly (compared with SPF) to the whole network instead of a few "hubs". To make this argument more convincing, in Fig. 9.1, we show the utilization of nodes for a network with 400 nodes using SPF or OR. As observed, utilization of a few hub nodes are extremely busy for SPF, while majority of nodes are having low utilization (<0.2). For OR case, however, utilization of nodes are not so extreme as this case of SPF, meaning that OR successfully spread the traffic around.

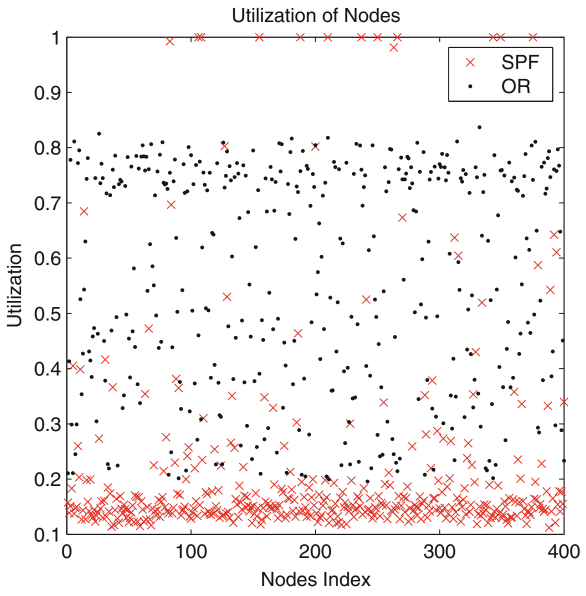


Fig. 9.1 Utilization of nodes (with index number 1–400) for a network using SPF (*red crosses*) or OR (*black dots*)

9.3 Loss, Delay, and Energy Performance of Shortest Path First and Optimized Routing

The results presented above and in the following are for an undirected scale-free network with the exponent of power-law degree distribution $\gamma = 3$. The network size is unchanged and the total number of nodes $N = 400$. The whole simulation period is divided into time slots such that in one time slot, for any individual node, at most one packet can be generated with a probability r varying between 0.05 and 0.95. The destinations for any packets are chosen randomly among the other $N - 1$ nodes. We assume all nodes having the same processing capacity and the same size of buffer (for storing waiting packets), which are 1 packet per time slot and 10 packets buffer size respectively. For all nodes, any packet received when buffer is full will be dropped. That means that the packet will be lost. When evaluating the energy performance, we consider two situations. In the first situation, energy supplied to each node is infinite, and each time when a packet is generated or forwarded by a node, the energy consumption of this node will be increased by 1 unit. If a packet is dropped, the energy consumed by it will still be counted. And then we calculate the average energy consumption to forward one packet arriving its destination. In the second situation, energy supplied to each node is finite. When any node runs out of energy, the simulation ends. This time we focus on the remaining energy of nodes when the simulation ends.

In the following, we first show how the use of routing algorithms affects the loss performance. The routing algorithms under study are SPF and OR. SPF is the basic algorithm that all packets will be routed with shortest paths, ignoring the congestion

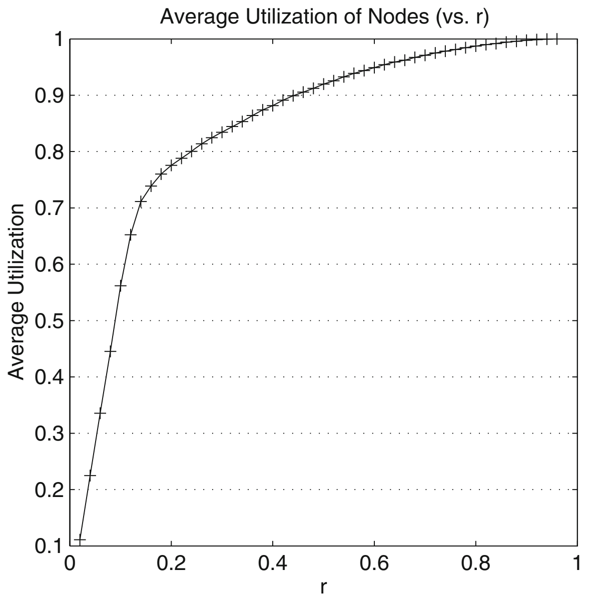


Fig. 9.2 The average utilization of nodes vs. r when OR is used

of the links and nodes. Naturally, a few nodes will be passed through by a lot of shortest paths. These nodes are called hubs. Hubs easily suffer from heavy traffic and congestion. Another algorithm, OR, is proposed to avoid node congestion [21]. With OR, the distribution of betweennesses for nodes is much more concentrated. Most of them are distributed within a narrow band, but there is a very sharp peak at the upper edge (see [21]). Although OR successfully spreads the traffic around, its loss performance has not been studied. In the following, simulation results on OR and SPF will be reported. When doing the comparison, we focus on the range with $r < 0.3$. It is because when $r = 0.3$, it is already an extremely heavy loading condition. This can be seen from the simulation results of OR on utilization of nodes in Fig. 9.2. From the figure, we observe that the network utilization increases rapidly in the range $r < 0.1$. It then tends to saturate for $r > 0.2$. At $r = 0.3$, the average utilization is really very high (> 0.8), which means the network is extremely busy at this point.

Figure 9.3 shows the average number of lost packets and the average number of nodes in dropping packets for SPF and OR with different values of r. With $r < 0.3$, we observe in Fig. 9.3 that OR performs much better than SPF in packet loss. We also observe that more nodes are involved in packet dropping. For example, when $r = 0.2$, with SPF, 56 packets will be dropped by only 14 nodes. But when OR is used, the number of dropped packets is reduced to 42, and they are dropped by 30 nodes. The results have shown that OR successfully spread the traffic to more nodes. Therefore, better throughput performance is obtained.

However, distributing traffic to more nodes means giving up the use of shortest paths. Packet delay will be increased. This is observed in Fig. 9.4. According

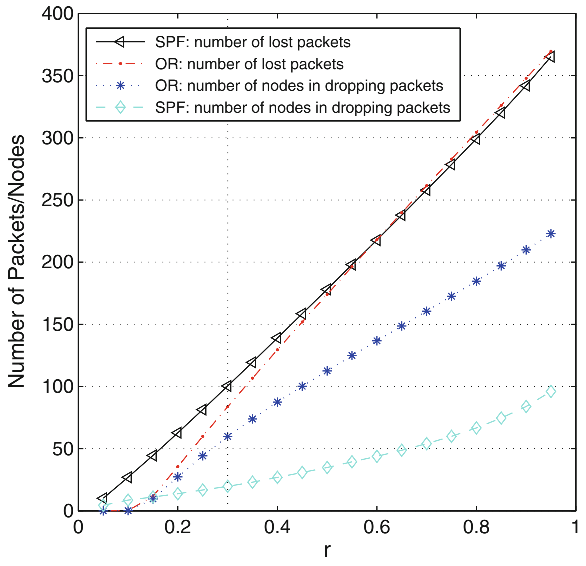


Fig. 9.3 Number of lost packets and nodes that drop packets as a function of *r*

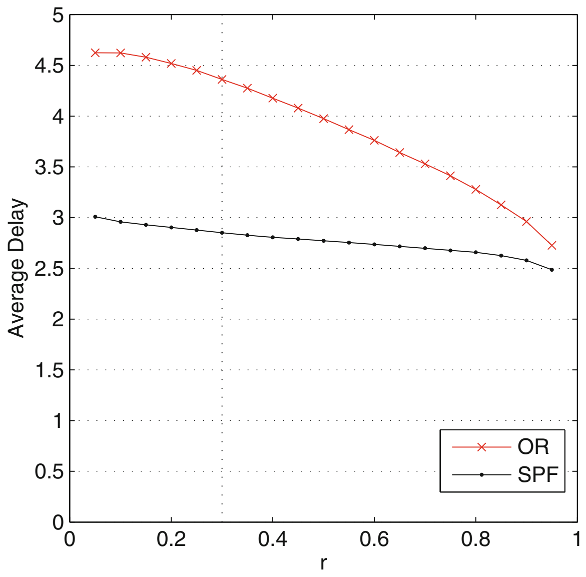


Fig. 9.4 Comparison of results for the average packet delay of two algorithms as a function of *r*

to Fig. 9.4, we can find that average packet delay of OR is much longer than that of SPF. In $r < 0.3$, the delay of OR is between 4.3 and 4.6. But for SPF, the delay varies between 2.85 and 3 in the same range of *r*. In other words, the average delay is increased to more than 1.5 times when using OR. When *r* becomes larger than 0.3, the overall average delays for both algorithms are decreased with the increase of *r*.

To find out the reason, another simulation is run to show the percentage drop of packets against the number of hops in a path. The results are shown in Fig. 9.5a,b.

Figure 9.5a,b show that no matter SPF or OR is used, packets having routing paths of more hops are dropped at higher percentages than packets that only pass through fewer hops. In other words, packets that can arrive to their destinations (not being dropped) are more probably to be those following paths with fewer hops. So the overall average delay for both algorithms is decreased with the increase of r. Moreover, when r is large, routing with pure OR has a higher probability in packets drops. The reason is that in OR, the routing paths are longer than those of SPF. We can find it from the distribution of the number of passing-by hops from Fig. 9.6. With the increasing of r, the percentage drops of packets passing-by hops increases rapidly for OR. The growth rate of loss percentage for OR is, therefore, much higher than that of SPF.

The above results show that the delay performance of OR is not so good. If we use OR to route all packets including those carrying delay sensitive data (such as VoIP), poor quality of service can be anticipated. In comparison, SPF is the better algorithm for routing time sensitive data. But for data which is not delay sensitive (e.g. TCP data), delay performance of the network is not so important when compared to the loss performance. This is because for this type of data, packet loss usually triggers retransmissions that significantly affect the overall throughput. On the other hand, this type of data can usually tolerate longer packet delay. This is due to the fact that an end-to-end estimation on the round trip time between both end nodes is made periodically.

As for the energy consumption of this two algorithms, two simulations have been run. For the first simulation on energy performance, we assume the energy supplied to each node is infinite, and each time a packet is generated or forwarded by a node, the energy consumption of this node will be increased by 1 unit. If a packet is dropped, the energy it consumed will also be accumulated to the total energy consumption. When the simulation ends, we calculate the average energy consumption for forwarding a packet to its destination successfully.

Figure 9.7 shows the average energy consumption for each forwarded packet of these two algorithms. When $r < 0.3$, OR gives a better performance than SPF. With the increase of r, the energy performance of OR gets worse rapidly. This is just the other side of the same observation as discussed a bit earlier: OR drops more packets than SPF when r is large. More packet drops also mean more energy being wasted. So when the traffic is heavy ($r > 0.3$), the performance of OR is not as good as that of SPF.

In the second simulation on energy, we assume that the energy supplied to each node is 10,000 units. Each time a packet is generated or forwarded by a node, the remaining energy of this node will be decreased by 1. When there is any node that runs out of energy, the simulation ends and we then find the remaining energy distribution of all nodes.

Although the average energy consumption of packets in OR is not as good as SPF, the energy consumption for different nodes seems to be well-distributed when simulation ends. Figure 9.8a shows the remaining energy of the nodes at different

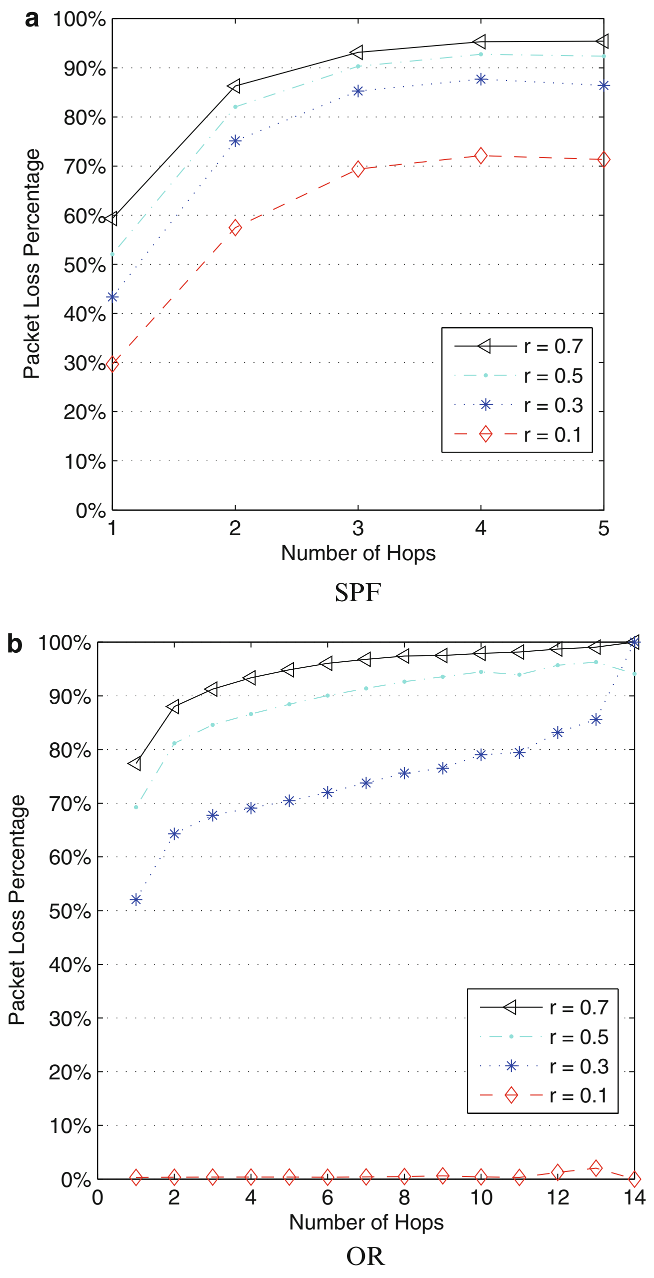
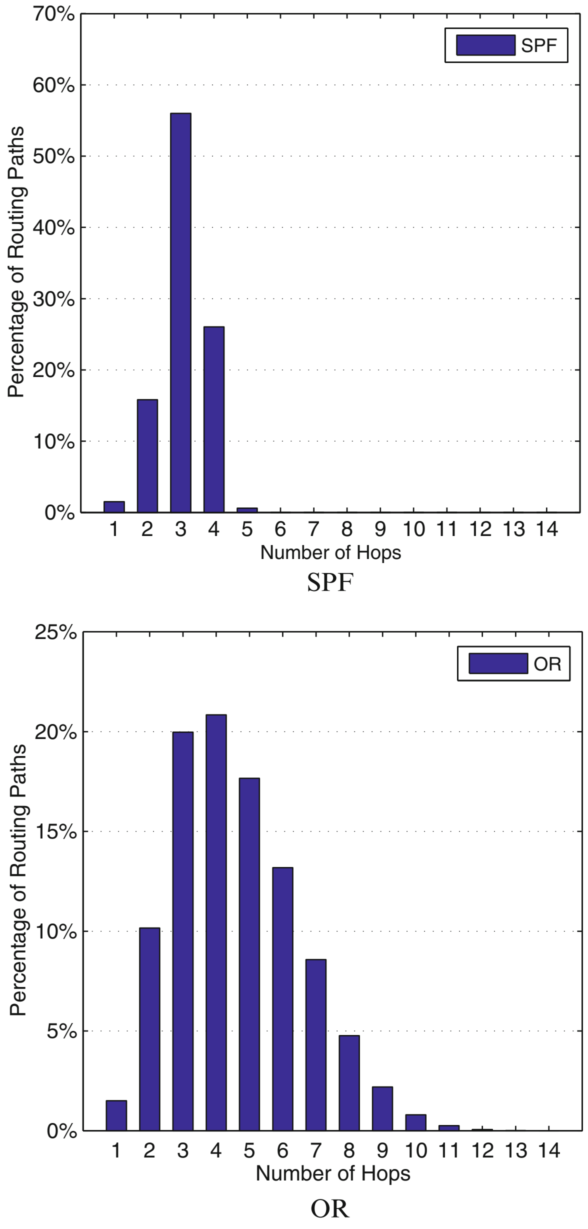


Fig. 9.5 The percentage drops of packets for SPF and OR

Fig. 9.6 The distribution of the number of passing-by hops in routing paths



time for SPF. For example, at time $t = 6{,}000$, there are 123 nodes having remaining energy of about 9,100. From the figure, we can find that for SPF, a few nodes run out of their energy very soon while most of the nodes have a lot of energy unused. But from Fig. 9.8b, the situation of OR seems to be much better.

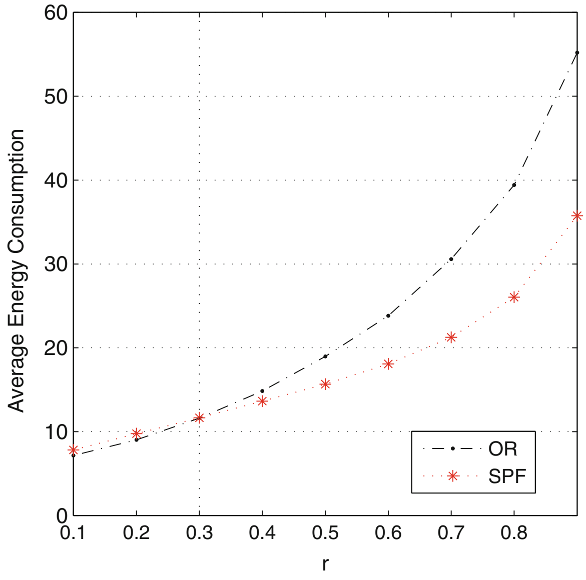


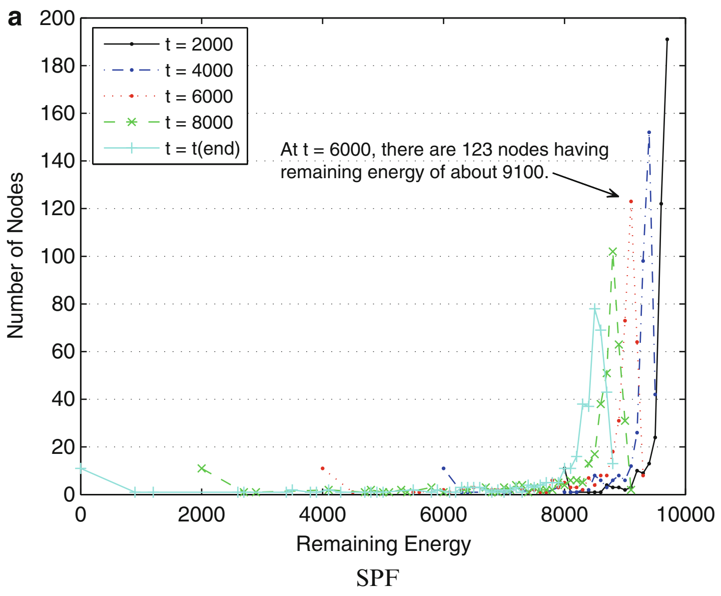
Fig. 9.7 Comparison of results for the average energy consumption for packets of two algorithms as a function of *r*

The results on energy performance shows that for SPF algorithm, it has better performance on the average energy consumption for packets. But as few nodes will run out of its energy rapidly, the network will stop working while most other nodes still have a lot of unused energy. As for OR, although the energy consumption speed of different nodes is quite uniform, the average energy consumption for packets is not so satisfying.

In conclusion, both SPF and OR cannot give satisfied performances on packet delay, packet loss and energy consumption at the same time.

9.4 Parallel Routing Algorithms

Traditionally, all routing algorithms are single routing ones. For a single routing algorithm, there is only one routing table in each router. That means for a specific router, it treats all packets equally and ignores their types or potential requirements. The situation is quite different in parallel routing. In parallel routing, each router has more than one choice on "next-hop" of the routing path. Distinguish different packet types and apply "differentiated treatment" to them is the core idea of parallel routing. The concept of parallel routing algorithm is there for many years. In computer networks, there are separate routing tables for unicast and multicast traffic. However, this chapter introduces a new concept on using two routing tables to route different types of unicast traffic. As can be seen below, parallel routing will give much more balanced performance.



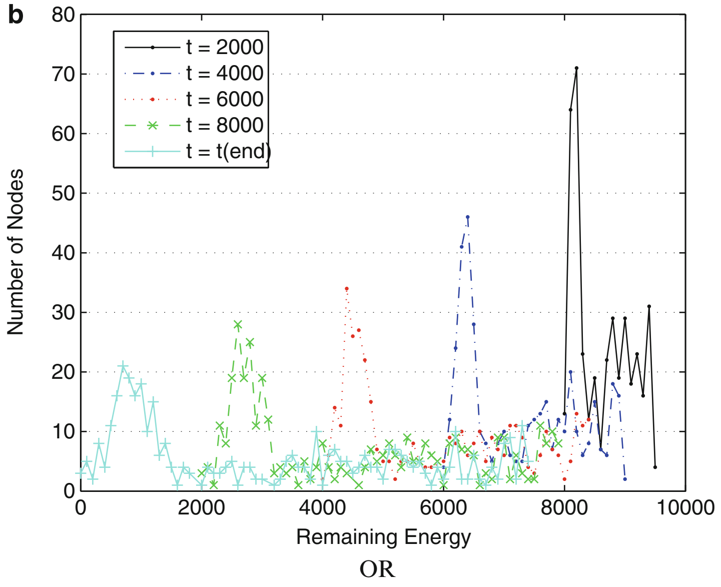


Fig. 9.8 Remaining energy distributions at different time of SPF and OR (with $r = 0.1$)

9.5 Optimization Based on Traffic Types: Classified Traffic Routing

To balance loss, delay and energy performance for different types of traffic, we propose an algorithm which keeps both SPF and OR routing tables. Based on the type of a packet, one of the routing tables will be used to route that packet. In the packet header of IP packets, there is a special field called Type of Service. By reading this field a node can know whether delay sensitive data is carried in this packet or not. Then the node can select the proper routing table to route this packet.

The proposed algorithm CTR is described below:

1. Packets are classified into two different types. Packets classified as type 1 are delay sensitive packets, and all other packets are classified as type 2.
2. In each node, two routing tables are kept– routing table 1, which is obtained by SPF and the other one, routing table 2, which is obtained by OR.
3. When routing a packet, a node will first determine the type of packet. If it is a type 1 packet, the node will forward the packet according to routing table 1. Otherwise, routing table 2 will be used to forward the packet.

To compare the performance of CTR to SPF and OR, simulation on the same network as described above was run. Different percentage mixes of type 1 and type 2 traffic were studied in our simulation. Since similar results were obtained for different percentage mixed, in the following we only present the results for a mix of 50% type 1 and 50% type 2 traffic.

Figure 9.9 shows the loss performance of CTR when compared with SPF, and OR. We can find that when $r < 0.3$, packet loss percentages for both type 1 and type 2 are less than that of SPF. It is interesting to find that the loss for type 2 traffic is even less than OR when r goes larger. To explain this, we first show the distributions of loss in the whole network for SPF, OR and CTR in Figs. 9.10, 9.11 and 9.12, respectively.

Figure 9.11 shows a uniform "busy state" in the whole network when OR is used. It is quite different from the case that when SPF is used (see Fig. 9.10). When SPF is used, there are a few "hubs", which means that too many shortest paths pass by these nodes. These "hubs" are too busy all the time and keep on dropping packets. Meanwhile, most of the other nodes are idle. For the case of CTR, as CTR type 1 packets are routed under SPF, there are also some "hubs". These "hubs" mainly drop type 1 packets (see Fig. 9.12) and a few type 2 packets which are occasionally routed to them. At the same time, CTR type 2 packets are routed under OR, which leads to "busy state" occurring on more nodes on the network.

Now we look back at Fig. 9.9 and explain why the loss for type 2 traffic is larger than that of OR when r is small (i.e. <0.14). When r is very small, using OR will lead to very few packets dropped in the whole network. However, if using CTR, although r is small, "hubs" still exist. All the packets passing though "hubs" may be dropped, including a lot of CTR type 1 and a few CTR type 2 packets. So, overall, when r is small, CTR type 2 packets are more easily dropped when compared to the

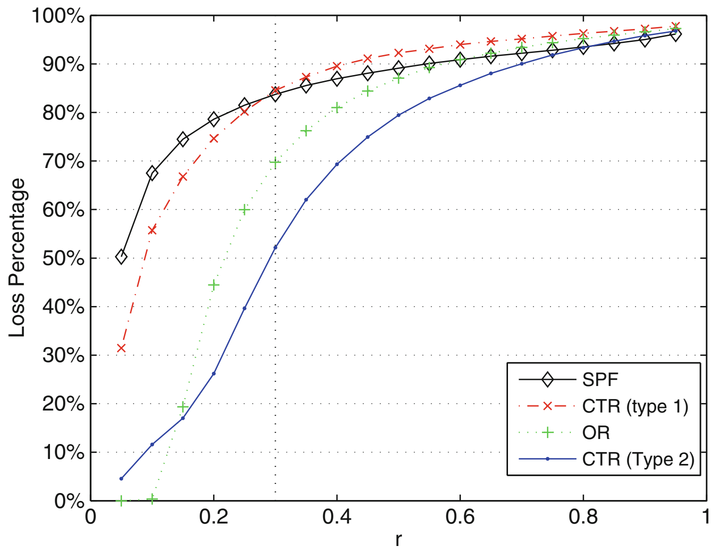
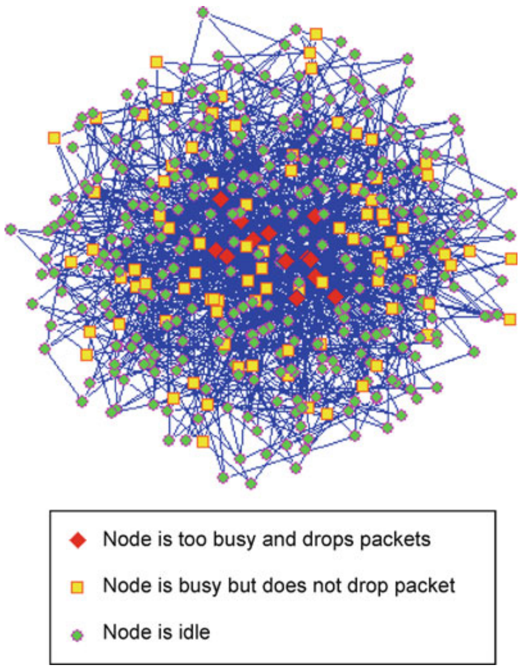


Fig. 9.9 Loss performance of CTR, SPF and OR

Fig. 9.10 Nodes states of the whole network when using SPF



Node is too busy and drops packets
Node is busy but does not drop packet
Node is idle

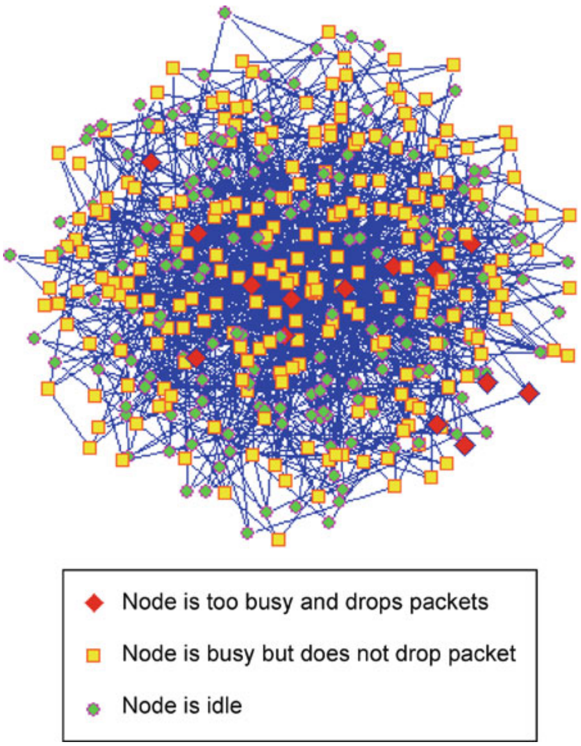

◆ Node is too busy and drops packets
□ Node is busy but does not drop packet
● Node is idle

Fig. 9.11 Nodes states of the whole network when using OR

OR case. Additional simulation results in Fig. 9.13 verify what we have stated. In the figure, the influence of CTR type 1 traffic on CTR type 2 traffic is shown. When there is no type 1 traffic (red line), there is nearly no loss in type 2 traffic for all hop lengths. However, when type 1 traffic exists (black line), many type 2 packets are dropped by the busy nodes (or, type 2 traffic going through shorter paths).

When traffic in the network increased (i.e. *r* increased), the possibility for a node to be overloaded is increased. When using OR, as the mean length of routing paths is increased, a packet has to pass through more nodes to reach its destination. If anyone among these nodes is overloaded, this packet will be dropped. So, heavier traffic means higher possibility that an arbitrary node is overloaded and higher risk for an arbitrary packet to be dropped. However, for CTR, some packets are routed under SPF. These packets are more likely to be routed to a small set of nodes centre on "hubs" than to be uniformly routed to the whole network. It leads to much less traffic in the other nodes. Although little more CTR type 2 packets will be dropped in the "hubs", in the remaining part of the network, much less packets need to be routed. As a result, less CTR type 2 packets will be dropped. This tells why the loss for type 2 traffic is even less than OR when *r* goes larger. Additional simulation results in Fig. 9.14 verify what we have stated. From the figure, it can be found that when packets need to pass by more than 3 hops, routing with OR will lead to higher risk of being dropped compared to using CTR (type 2).

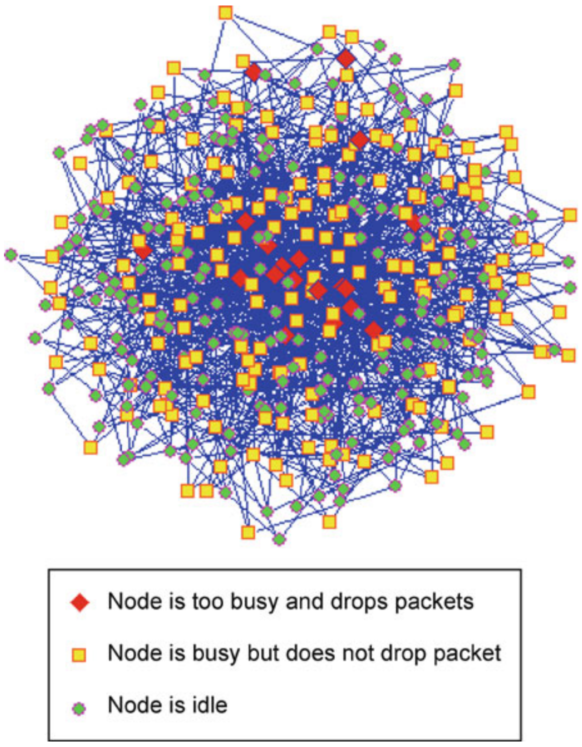

◆ Node is too busy and drops packets
▣ Node is busy but does not drop packet
● Node is idle

Fig. 9.12 Nodes states of the whole network when using CTR

The above results show that for CTR type 1 packets, the loss performance is slightly better than SPF for $r < 0.3$ while that of CTR type 2 packets is worse than OR for small r. But when r is large (not beyond 0.3), the loss performance for CTR type 2 packets is better than that of OR.

Next, we present the delay performance. Figure 9.15 shows the delay performance of CTR when compared with SPF and OR. We observe that the delay for type 1 packets is just as small as that of SPF. For CTR type 2 packets, their average delay is longer than that of OR.

To see why the average delay for CTR type 2 packets is longer than that of OR, refer to Fig. 9.15. From the figure we can find that when r is small enough so that no dropped packets occur, the average delay for CTR type 2 packets is very close to that of OR. When r goes larger, packets dropping occurs. As discussed before, when r is large, routing with pure OR has a higher probability in packet drops. Furthermore, most of the dropped packets are those having the longer routing path. That means, with OR, when r goes larger, packets with longer delay are dropped with a higher probability compared to other packets. So the delay performance (as reflected only by those packets that can reach their destinations) is better. On the contrary, in CTR, type 1 packets are routed with SPF. These type 1 packets indirectly affect the performance of type 2 packets, as they make the nodes in the hub set busy

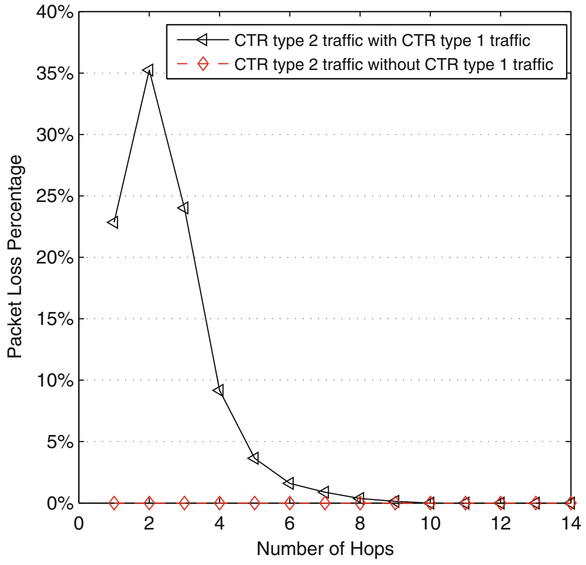


Fig. 9.13 The percentage drops for CTR type 2 packets without or with the influence of CTR type 1 packets when $r = 0.1$

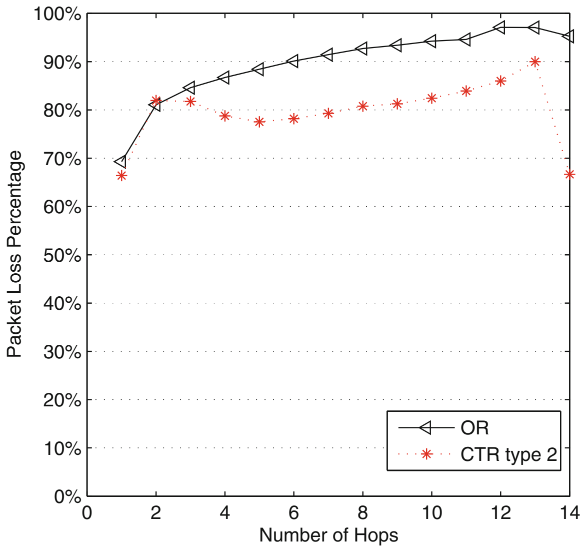


Fig. 9.14 The percentage drops of packets for CTR type 2 and OR when $r = 0.5$

all the time. CTR 2 packets will be dropped easily when passing through these nodes (i.e. nodes made busy mainly by type 1 packets). In other words, some type 2 packets with shorter delay are easier to be dropped. Since packets with longer delay will be more probable to be dropped in OR (therefore giving a "better" delay performance) while the opposite is found for CTR type 2 packets, the average delay for CTR type 2 packets is longer than the average delay of OR.

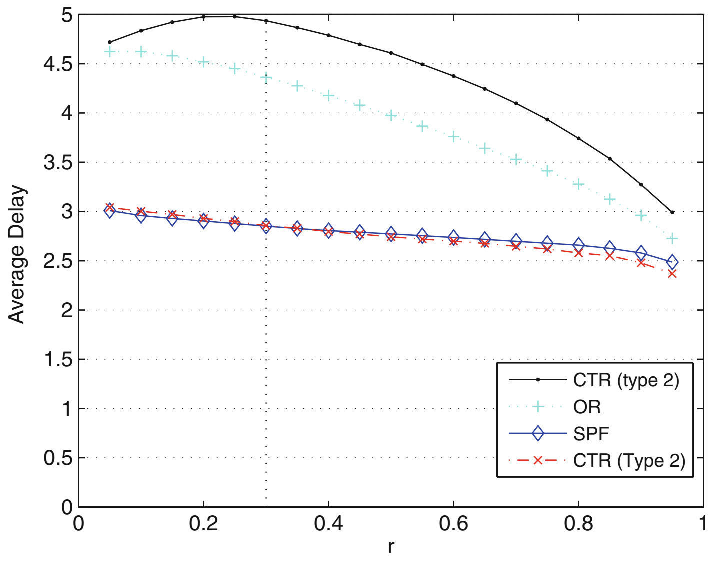


Fig. 9.15 Delay performance of CTR, SPF, and OR

According to the above results we can conclude that the delay performance for CTR type 1 packets is just as good as that of SPF, while the delay performance for CTR type 2 is slightly worse than OR.

Now we will discuss on the energy performance of CTR. Figure 9.16 shows the average energy consumption for packets of CTR with comparisons to SPF and OR. We can find that when r is small (e.g. $r < 0.3$), the average energy consumption for CTR is less than both SPF and OR. With the increasing of r, the curve for CTR is still lower than that of OR.

The average energy consumption has a strong relation with the loss performance. Generally, more loss means more waste, larger average energy consumption for a successful packet. When r is small, the overall loss performance for CTR is quite good, so the average energy consumption for packets is low. But as r increase, almost all packets will eventually be dropped, the situation of CTR becomes similar to that of OR which has been mentioned before: both CTR and OR try to forward packets by avoiding congestion and those packets may be dropped after it passing through a few hops. The energy is therefore wasted.

As for the remaining energy distribution, from Fig. 9.17 (please refer to Fig. 9.8 for comparisons), we find that when there is a node running out of its energy, the remaining energy of other nodes is much less than that of SPF, although it is more than that of OR.

This result can be easily understood that some packets are forwarded under SPF, while others are forwarded under OR. Nodes which are nearly always idle under pure SPF may be sometimes busy with the introduction of OR, so they also consume energy. The remaining energy of nodes is generally not as high as those of SPF.

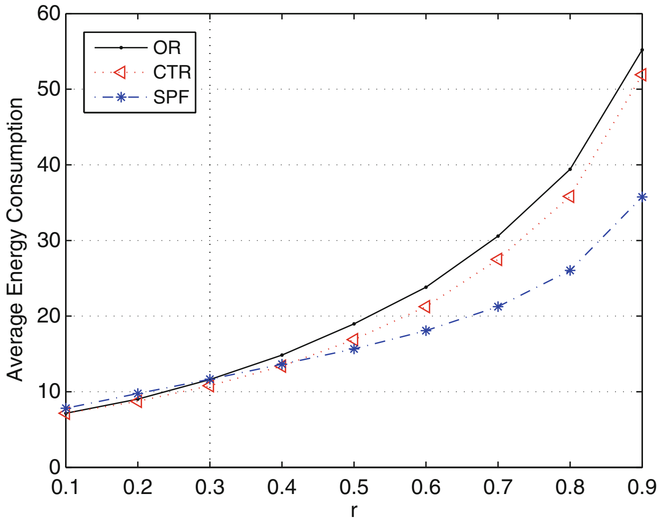


Fig. 9.16 Comparison of results for the average energy consumption for packets of three algorithms as a function of r

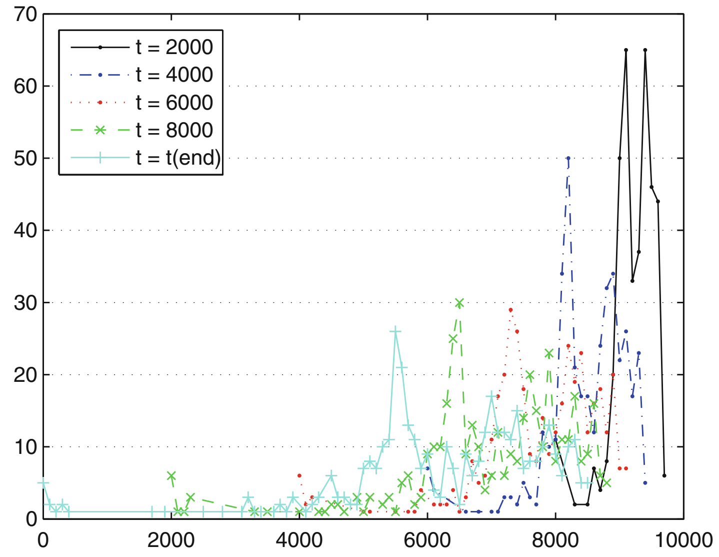


Fig. 9.17 Remaining energy distribution at different time of CTR with $r = 0.1$

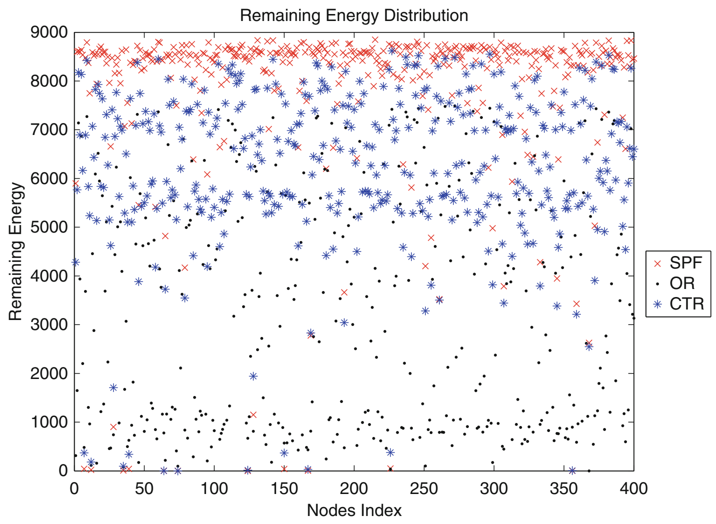


Fig. 9.18 Remaining energy distribution of SPF, OR, and CTR

Figure 9.18 shows the remaining energy distribution of SPF, OR, and CTR against the nodes' index (when there is a node running out of its energy). It can be seen that the remaining energy of nodes for CTR (the blue stars) are distributed, generally, lower than the that of SPF (the red crosses) while higher than that of OR (the black dots). We obtain the same result as mentioned before from another view – the remaining energy of other nodes is much less than that of SPF, although it is more than that of OR.

According to the above results, we can conclude that the average energy consumption of CTR is less than that of OR, and when the network is not so congested, it is even less than that of SPF. As for the remaining energy distribution, CTR's is much better than that of SPF although not so good as that of OR.

9.6 Conclusion

All the above results show that when using CTR, packets which are delay sensitive can reach their destinations as quickly as SPF. At the same time, the loss performance of CTR is better than SPF. The loss for packets which are not delay sensitive is even less than OR. CTR therefore can give a balance performance in delay and loss. As for the energy performance, when using CTR, the overall average energy consumption for each packet is quite good compared to OR, while the energy invested to nodes is well-distributed.

In conclusion, parallel routing (i.e. hold two routing tables) has its own advantages. Though the proposed CTR may not be the best parallel routing algorithm, we do hope to call for further research on parallel or even multiple routing.

Acknowledgement This work was supported by Hong Kong Government General Research Fund (grant No. CityU-123608).

References

1. R. Albert and A.L. Barabasi, Statistical mechanics of complex networks, Rev. Mod. Phys, 74, 2002, pp. 47-97.
2. F. Wu, B.A. Huberman, L.A. Adamic and J.R. Tyle, Information flow in social groups, Physica A: Statistical and Theoretical Physics, 337, 2004, pp. 327-335.
3. S.N. Dorogovtsev and J.F.F Mendes, Giant strongly connected component of directed networks, Phys. Rev. E, 64, 2001, pp. 025101
4. M. Faloutsos, P. Faloutsos, and C. Faloutsos, On power-law relationships of the Internet topology, Comput. Commun. Rev. 29, 1999, pp. 251-262
5. A. Vazquez, R. Pastor-Satorras, and A. Vespignani, Large-scale topological and dynamical properties of the Internet, Phys. Rev. E. 65, 2002, pp. 066130
6. A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajaopalan, R. Stata, A. Tomkins, and J. Wiener, Graph structure in the web, Comput. Netw. 33, 2002, pp. 309-320
7. H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.-L. Barabasi, The large-scale organization of metabolic networks, Nature, 407, 2000, pp. 651-654
8. A. Wagner and D. Fell, The small world inside large metabolic networks, Proc. R. Soc. London, Ser. B., 268, 2001, pp. 1803
9. J.A. Dunne, R.J. Williams, and N.D. Martinez, Food-web structure and network theory: The role of connectance and size, Proc. Natl. Acad. Sci. U.S.A., 99, 2002, pp. 12917
10. J. Camacho, R. Guimera, and L.A.N. Amaral, Robust Patterns in Food Web Structure, Phys. Rev. Lett., 88, 2002, pp. 228102
11. M. Ericsson, M.G.C. Resende, and P.M. Pardalos, A Genetic Algorithm for the Weight Setting Problem in OSPF Routing, J. Comb. Optim., 6, 2002, pp. 299
12. B. Fortz and M. Thorup, Optimizing OSPF/IS-IS weights in a changing World, IEEE J. Sel. Areas Commun., 20, 2002, pp. 765-767
13. V. Gabrel, A. Knippel, and M. Minoux, A comparison of heuristics for the discrete cost multicommodity network optimization problem, J. Heuristics., 9, 2003, pp. 429-445
14. D. Allen, I. Ismail, J. Kennington, and E. Olinick, An Incremental Procedure for Improving Path Assignment in a Telecommunications Network, J. Heuristics., 9, 2003, pp. 375-339
15. E. Mulyana and U. Killat, Load Balancing in IP Networks by Optimising Link Weights, European Transactions on Telecommunications. 16(3), 2005, pp. 253-261
16. B. Danila, Y. Yu, J.A. Marsh, and K.E. Bassler, Optimal transport on complex networks, Phys. Rev. E., 74, 2006, pp. 046106
17. K. Kim, D. Nicklescu and S. Hong, Coexistence of VoIP and TCP in Wireless Multihop Networksm IEEE Communications Magazine, June 2009, pp. 75-81.
18. M.E. J. Newman, Phys. Rev. E 64, 016132 (2001).
19. Miniwatts Marketing Group. World Stats. Available: http://www.internetworldstats.com/stats.htm. Last accessed 22th Sep 2011 (2011).
20. J. Baliga et al., JLT, Vol. 27, No. 13, Jul. 2009
21. R. Tucker, “A green internet”, in Proc. IEEE LEOS Annual Meeting, Newport Beach, CA, Nov. 9-13, 2008.

22. J. Baliga, K. Hinton, and R.S. Tucker, Energy consumption of the Internet, Optical Internet, 2007 and the 2007 32nd Australian Conference on Optical Fibre Technology. COIN-ACOFT 2007. Joint Int. Conf. 1–3, June 2007
23. L. Zhao, Y.-C. Lai, K. Park and N. Ye, Phys. Rev. E 71, 026125 (2005).
24. T. Zhou, G. Yan, B.-H. Wang, Z.-Q. Fu, B. Hu, C.-P. Zhu and W.-X. Wang, Dyn. Contin. Discret. Impuls. Syst. Ser. B-Appl. Algorithm 13, 463 (2006).
25. A.-L. Barabasi and R. Albert, Science 286, 509 (1999).
26. A.-L. Barabasi, R. Albert and H. Jeong, Physica A 272, 173 (1999).
27. S. Sreenivasan, R. Cohen, E. Lpez, Z. Toroczkai, and H.E. Stanley, e-print cs.NI/0604023

Chapter 10
Internet Based Service Networks

LiYing Cui, Soundar Kumara, and Réka Albert

Abstract This chapter focuses on services networks. We review the important aspects of services (flow patterns, semantic issues, Quality of Service (QoS) and user preferences), as well as service composition techniques, network metrics and models. The Concept-Service (CS) network matrix is introduced. The CS network dynamics and optimization based service composition are the original contributions of this chapter based on our years of research on this topic.

10.1 Introduction

Information Technology provides rich connectivity which is an opportunity and a challenge as this interconnectedness leads to networks of millions of nodes (e.g. YouTube/Facebook users, mobile phone owners, and Internet service providers) and of considerable heterogeneity (the nodes include software, devices, and people and broadly Internet of things). Information sharing and retrieval in internet networks drives the day-to-day business across the world. In general, the map of the Internet is considered at two different scales. At the level of Autonomous Systems (ASs), an AS is an organizational unit of a particular domain or service provider, and the edges represent physical communications that connect sub-networks or devices across these ASs. The exchange of information between devices or sub-networks is done using routers. The map of the Internet at this scale consists of nodes as routers and their communications within and across ASs as edges.

The proliferation of e-commerce and the fast paced development of information technology have fueled the use of distributed Cyber business processes. Service composition enforces the usage of automatic Internet services (also called web

L.Y. Cui • S. Kumara (✉) • R. Albert
Department of Industrial Engineering, 310 Leonhard, University Park, PA 16802, USA
e-mail: luc5@psu.edu; skumara@psu.edu; rza1@psu.edu

services) to build different kinds of service complexes, which can meet a variety of business applications. Web services and the parameters/concepts involved in them can form a network if we take services and parameters (concepts) as nodes, and treat the information flow among concepts and services as arcs.

The prospect of web service boom has inspired a number of large Information Technology companies to work on service computing. Microsoft is developing its next generation operating system that embeds a service publisher. IBM, HP, Sun and others have been cooperating on a Universal Description, Discovery and Integration (UDDI) Center. Google has set up its own registry center for web services. It must be noted that service engineering focuses on the traditional hard services carried out by physical entities, but ignores the soft services carried out automatically via Cyber space.

In this chapter, we focus on the background of using optimization and semantics in service networks. We provide preliminary analysis on how to use complex networks in service composition.

10.2 Problem Definition

We have stated in the previous section that Information Technology provides rich connectivity among millions of nodes and thus leads to the evolution of complex networks. The links of these networks are the information flow shared among the nodes. One can make the following definitions:

Definition 10.1. Service provider: Any software, machine, people, product, etc., that can provide useful function to others.

Definition 10.2. Customer: The party which needs a service from a service provider.

Definition 10.3. Service: A function that a service provider offers to a customer

Definition 10.4. Web service (Internet based service): A service provided via WWW or Internet.

Definition 10.5. Concept: A concept is a cognitive unit of meaning which is used to describe services.

Definition 10.6. Service network: A network formed where the nodes are services and concepts and the links among nodes are the information flows among the nodes.

Definition 10.7. Service composition: Service composition is a process to select a set of services and their flow sequence to perform a task which cannot be finished by any individual service.

Definition 10.8. Input: A pre-condition of a service is defined as an input to the service.

Definition 10.9. Output: A post-condition or result of a service is defined as an output of the service.

A service is a set of related activities that produce value to the customers by fulfilling a one or several function(s). Web services belong to services. However, only those services using data flow as inputs and outputs can be developed as Web Services. If we only study the business processes composed by Web Services, we call it a web service composition problem (WSC); otherwise, we call it a general service com-position problem. A web service is described by pre-conditions (inputs) and post-conditions (outputs).

The relationship among web service composition, broad service composition is the following:

$$\text{Web Service Composition} \subseteq \text{General Service Composition} \tag{10.1}$$

We will use the terms service planning and service composition interchangeably. We illustrate service planning through the following scenario: Olivia, a consultant, wants to travel from State College to Washington D.C. Saturday or Sunday of this week. She would like to return from DC next Friday. She has meetings on Monday and Tuesday. On Wednesday, she needs to present a paper at a local university. On Thursday, if weather permits, she would like to watch a baseball game. Otherwise, she would go to the Space Museum. After the museum or baseball game, she wants to have dinner at a Mexican restaurant.

She may need to accomplish the following tasks in order to implement her plan:

1. Book an airline ticket from State College to Washington D.C.
2. Book a hotel in Washington D.C.
3. Rent a car in Washington D.C.
4. Check the weather for Thursday.
5. If the weather forecast is good for Thursday, buy a ticket for the baseball game.
6. If the weather forecast is bad, check the schedule of the Space Museum.
7. Reserve a seat in a Mexican restaurant for dinner.

The task order (with antecedent and consequent states) results in a plan. The generation of this trip planning is a web service composition problem. Figure 10.1 shows the service composition process.

10.3 Components of Service Composition

10.3.1 Flow Pattern

In service composition problems, there are four types of service flow patterns, namely Sequential, Switch, Parallel, and Iterative. In Fig. 10.2a, Service 1 is the predecessor of Service 2, so these two services are sequential. In Fig. 10.2b, either Service 1 or Service 2 is adequate for the whole compound service, so the

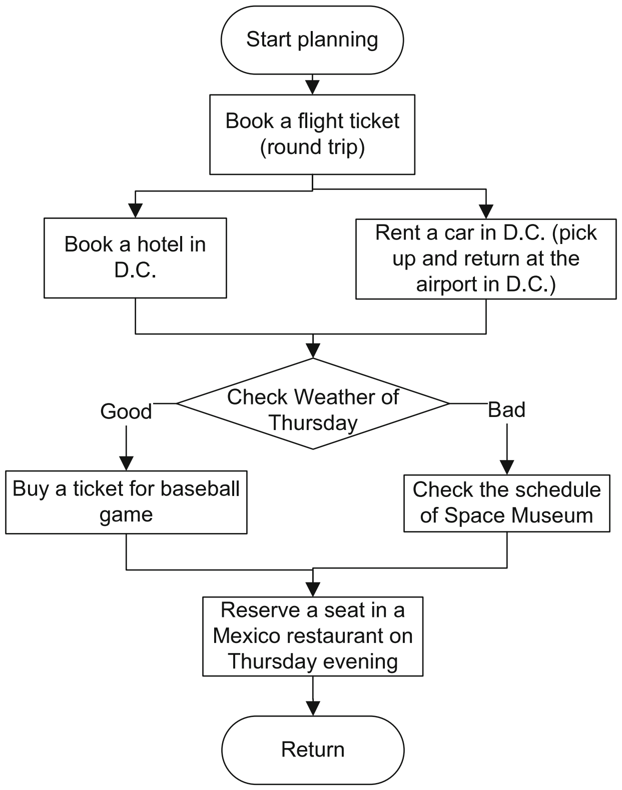

Start planning
Book a flight ticket (round trip)
Book a hotel in D.C.
Rent a car in D.C. (pick up and return at the airport in D.C.)
Check Weather of Thursday
Good
Bad
Buy a ticket for baseball game
Check the schedule of Space Museum
Reserve a seat in a Mexico restaurant on Thursday evening
Return

Fig. 10.1 An example of a web service composition

relationship between Service 1 and Service 2 is a switch (we call it "Or" Parallel in this context), with a probability for using Service 1, and probability for using Service 2. In Fig. 10.2c, both Service 1 and Service 2 are needed to provide input information for the successors, so the two services are parallel (we call it "And" Parallel in this context). In Fig. 10.2d, Services 1, 2, and 3 are in a loop, so we call this iterative flow pattern. In the example shown in Sect. 10.1, the task of booking an air ticket and the task of booking a hotel in Washington D.C. are sequential, since the number of nights needed in a hotel depends on the departure date from State College. The tasks of booking a hotel and renting a car can be executed in parallel. Buying a ticket for a baseball game and checking the schedule of the Space Museum are conditional tasks, and the flow pattern is a switch. There is no iterative flow pattern in this plan.

The data structure of web services is shown in Fig. 10.3. Web services are stored in a hierarchical form and are classified into different port types; one service may have multiple operations. Each web service operation includes a set of inputs and a set of outputs. Two web service operations can communicate with each other if

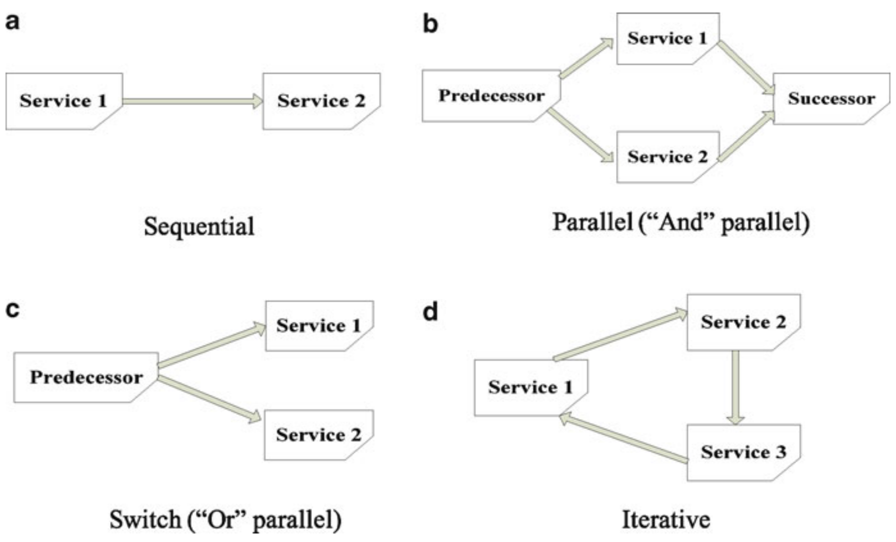


Fig. 10.2 Four basic service flow patterns

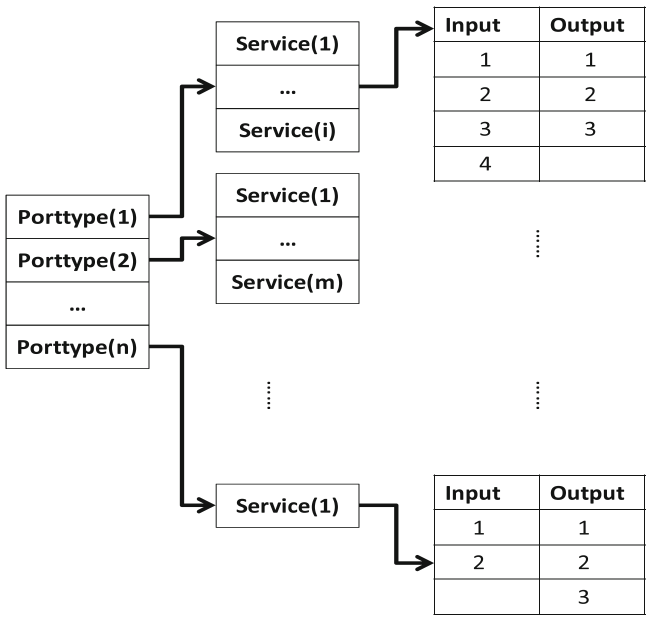


Fig. 10.3 Data structure of web service database

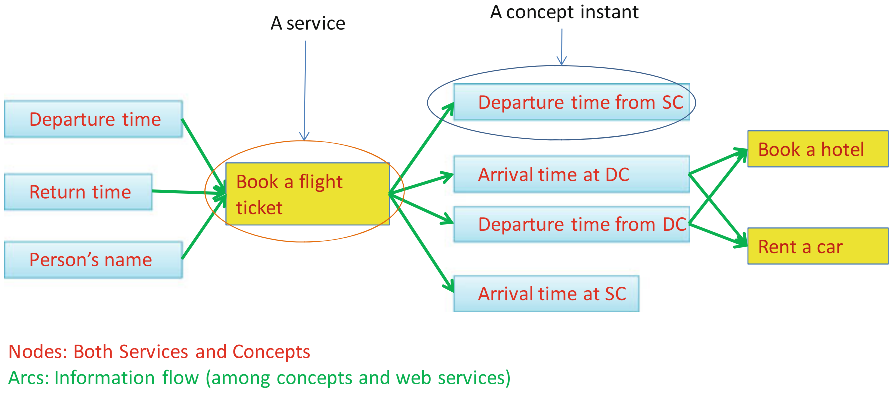


Fig. 10.4 The inner structure of web services (In this chart, parameter is concept)

one's output(s) matches the others' input(s). In order to accomplish the service composition, each web service must specify the properties of the service, such as where it is located (URL), how it is invoked (inputs), and what it does (outputs). To guarantee the successful information exchange among web services, the service providers must describe and publish the web services in a specified standard(s), such as WSDL [1], BPEL4WS [2], WS-Choreography [3], OWL-S(DAML-S) [2].

The communication between two web services is via sharing inputs or outputs between them; e.g., an output of one service can be used as an input of another. Figures 10.2, 10.3 and 10.4 show the sharing scheme.

10.3.2 Semantic Aspects

It is expected that all the input and output attributes of web services are described using standard semantic concepts. So, the web services are semantically interpreted in an ontology [36] and registered in service registries such as, for instance, UDDI [3]. A Semantic Service Matching Algorithm is used to compare the concepts derived from processing the ontology and the requirements from the users with the concepts presented in the format of published services. Next, we describe the most popular semantic matching algorithm in service composition.

Each concept has its sub-classes and super-classes. The service matching algorithm searches the Web Service Description file and the ontology file to check the semantic relationship between every pair of concepts. The algorithm derives the explicit and implicit relations between concepts are derived through reasoning. Table 10.1 illustrates four basic types of relationships between two concepts: (a) Concept 1 is the same as Concept 2; (b) Concept 1 is an ancestor of Concept 2; (c) Concept 1 is an offspring (subclass) of Concept 2; (d) Concept 1 has no relationship with Concept 2.

Table 10.1 Types of similarity match between two concepts (C_1, C_2)

Description in words	Semantic description
C_1 exactly matches C_2	$C_1 = C_2$
C_1 partially matches C_2	$C_1 \subset C_2$
C_1 over matches C_2	$C_1 \supset C_2$
C_1 fails to match C_2	No relation

A score based on the match is assigned based on these relations:

$$match(C_1, C_2) = \begin{cases} Score1 \text{ if } C_1 = C_2 \\ Score2 \text{ if } C_1 \subset C_2 \\ Score3 \text{ if } C_1 \supset C_2 \\ 0 \quad\quad \text{otherwise} \end{cases} \quad \text{Here } Score1 > Score2 > Score3 \tag{10.2}$$

Definition 10.10. Concept domination: Concept C_1 dominates concept C_2 if concept C_1 has more information than concept C_2 does.

As we know, if concept C_1 is a subclass of concept C_2, concept C_1 has more information than concept C_2 does.

Lemma 10.1. *If concept C_1 is a subclass of concept C_2, e.g. $C_1 \subset C_2$, then C_1 dominates C_2.*

When we need to check whether a service S_1 can be used instead of another service S_2, we need to compare the input and output concepts. It is preferred that service S_1 needs no more input information than service S_2 does, and generates no less output information than service S_2 does.

Definition 10.11. Service domination: Service S_1 dominates service S_2, if service S_1 needs no more input information than service S_2 does, and generates no less output information than service S_2 does.

Let the symbol *input* represent the total input concept set of service S_1 and service S_2, and the symbol *output* represent the total output concept set of service S_1 and service S_2. The following lemma holds.

Lemma 10.2. *Service S_1 dominates service S_2, if $C_i^{S_1} \supseteq C_i^{S_2}$, $i \in input$, and $C_i^{S_1} \subseteq C_i^{S_2}$, $i \in output$.*

When a query R is posed, service S can be used as a candidate to fit the request R if service S dominates the request R. The usability score of an existing web service S to query R can be defined as:

Definition 10.12. Usability score:

$$\text{Usability score}(R, S) = \sum_{i \in input} match(C_i^R, C_i^S) + \sum_{i \in output} match(C_i^S, C_i^R) \tag{10.3}$$

The above formula shows us how to calculate the usability score of a service for a query. The usability score of a service for a request consists of the match scores of all the input concepts and output concepts of the request. All the input and output matching scores between the request and the service are summed up into the usability score of a service to the requested service. All candidate services can be sorted according to their usability scores. The service with the highest usability score wins. The candidate services can also be further evaluated according to the Quality of Service (QoS). The disadvantage of the Semantic Matching Algorithm is that it is only used to match a query with existing services, instead of doing service composition. However, we can use semantic matching to identify the topologically significant concepts.

10.3.3 Quality of Services

Recently, user preferences have drawn more attention in the service oriented research field [4]. However, some traditional automated service composition approaches cannot meet the personalized user requirements due to the following reasons:

1. Common knowledge cannot characterize personalized requirements. For example, if the user wants to go to Washington D.C. from State College, taking a flight minimizes time but driving is also a viable option. In case the user is afraid of flying, or annoyed with the frequent delays airlines have or finds the cost of flying too high, flying is not the best option. In fact, the users' requirements are variable and unpredictable in different environments. Therefore, it is impossible to forecast all personalized requirements in advance.
2. Users' requirements include both hard and soft constraints. In the traditional approaches, only a user's initial state is considered as a constraint to implement planning, e.g., the user may have $5,000$ dollars budgeted for the trip. In addition, if the user wants to travel in a modality that both minimizes travel time and offers the least cost, there may be no way to satisfy both of her preferences at the same time. Thus, she has to make a compromise to select a feasible conveyance. This kind of soft-constraint is usually fuzzy and negotiable. It is important to realize that the user-centered service composition should pay much more attentions to flexible user preferences for improving users' satisfaction. In addition, the soft-constraints will also improve the success rate of service composition if a compromise feature is considered.

We consider five general QoS criteria in service selection: (1) execution price, (2) execution duration, (3) reputation, (4) reliability, and (5) availability.

A web service can contain more than one operation. The operations can be different in terms of functionality and QoS. As the service selection technique can also be applied in operation selection, we can treat each operation as an individual service. Based on Zeng et al. [6], we define the components of the QoS of services.

Table 10.2 Computing the QoS of a compound service $s(s_1, s_2, \ldots, s_n)$ (derived from [6])

Criteria	Equation
Price	$C(s) = \sum_{i=1}^{n} C(s_i)$
Duration	$D(s) = C_{\max}(s)$
Reliability	$Q(s) = \prod_{i=1}^{n} e^{R(s_i)}$
Availability	$A(s) = \prod_{i=1}^{n} e^{A(s_i)}$
Reputation	$R(s) = \sum_{i=1}^{n} \frac{1}{n} R(s_i)$

Definition 10.13. Price: Given a service s, the price $C(s)$ is the fee that a service requester has to pay for invoking service s.

Definition 10.14. Execution duration: Given a service s, the execution duration $D(s)$ measures the expected delay between the moment when a request is sent and the moment when the results are delivered. The execution duration is computed using the expression $D(s) = T_{\text{process}}(s) + T_{\text{trans}}(s)$, is the sum of the processing time and the transmission time. The transmission time is estimated based on past executions of the service, i.e., $T_{\text{trans}}(s) = \frac{\sum_{i=1}^{n} T_i(s)}{n}$, where $T_i(s)$ is one of the past transmission times, and n is the number of executions of this service observed in the past.

Definition 10.15. Reliability: The reliability $Q(s)$ of a service s is the probability that a request is successfully executed within the maximum expected time indicated in the web service description. Reliability is computed from the data of the past invocations using the expression $Q(s) = \frac{N_{\text{successful}}(s)}{N(s)}$, where $N_{\text{successful}}(s)$ is the number of times that the service s has been successfully delivered within the maximum expected time frame, and $N(s)$ is the total number of invocations of the service.

Definition 10.16. Availability: The availability $A(s)$ of a service is the probability that the service is accessible which is computed as: $A(s) = \frac{T_a(s)}{T(s)}$, where $T_a(s)$ is the total amount of time when service S is available during the total $T(s)$ amount of operation period.

Definition 10.17. Reputation: The reputation $R(s)$ of a service s is a measure of its trustworthiness. It mainly depends on the comments from users' experiences of using this service. Different users may have different opinions on the same service. The value of the reputation is defined as the average rank given to the service by n users: $\frac{\sum_{i=1}^{n} R_i}{n}$, where R_i is the users' ranking on a service's reputation, n is the number of times the service has been graded (derived from [6]).

Definition 10.18. Quality of a service: The quality a service s is defined by the following expression: $q(s) = (C(s), D(s), Q(s), A(s), R(s))$

Methods to compute the QoS criteria vary depending upon the application and we show our schema in Table 10.2.

In Table 10.2, $C(s_i)$ is the price of service s_i, $i = 1,2,...,n$. $C_{\max}(s)$ is the makes pan of the service complex. $Q(s_i)$ is the reliability of service s_i, $i = 1,2,...,n$. $A(s_i)$ is the availability of service s_i, $i = 1,2,...,n$. $R(s_i)$ is the reputation of service s_i, $i = 1,2,...,n$.

We assume that preferences upon a parameter form a totally ordered set in the domain of the parameter. In most of the cases, it is difficult to find a plan which can satisfy all the requirements. In [5], a global utility function is either implicitly or explicitly assumed to implement the trade-off between different preferences. The utility-based approaches may not be convenient for users. For example, it can be difficult for a user to weights to cost and performance. The theory of Pareto dominance can be used to select relatively better plans from feasible plans. Interested readers are referred to [7] for details.

Definition 10.19. Pareto dominance: Let A and B be the two candidate plans for a planning problem, and $(p_1, p_2, ..., p_n)$ consists of a set of preference criteria. Let $p_i(A)$ and $p_i(B)$ be the preference value of plan A and plan B for the preference criterion p_i. If $p_i(A) \geq p_i(B)$ for any i, we call plan A Pareto-dominates plan B.

10.3.4 User Preferences and Performance Metrics

User preferences play an important role in business activities, and are gaining attention in the web service research field, especially in personalized e-services. Given the explosion of web services several plans can be generated with differences in functionality, culture, QoS, and data:

1. *Functional differences*: Similar services may differ in the detail of their functions. For example, two travel services – book an air ticket, reserve a hotel room, rent a car – and – book an air ticket, reserve a hotel room – both provide the function of trip planning; however, the first service has one more sub-functionality (rent a car) compared to the second one.
2. *Cultural differences*: Similar services may refer to different cultural actions. For example, one service requires users to pay before delivery while another service charges after delivery. These two services need different preconditions, thus they should be applied to different users.
3. *QoS differences*: Similar services may deliver different QoS features. For example, a service using one algorithm may be more efficient than a service using another algorithm; however, the former may have less privacy than the latter.
4. *Data differences*: Similar services' preconditions may be different in terms of some data constraints. For example, a specific travel service can only serve those users with certain destinations and transportation options.

Definition 10.20. Satisfaction Density: $DP = \frac{N_v}{N_t}$, where N_v represents the total number of user preference vectors that are satisfied, and N_t represents the total number of preference vectors.

Definition 10.21. Satisfaction degree:

$$SD = 1 - \frac{\sum_{i=1}^{n} rd(v_{A_i}) - \sum_{i=1}^{n} rd_{\min}(v_{A_i})}{\sum_{i=1}^{n} rd_{\max}(v_{A_i})} \tag{10.4}$$

where $\sum_{i=1}^{n} rd_{\max}(v_{A_i})$ and $\sum_{i=1}^{n} rd_{\min}(v_{A_i})$, respectively, represent the maximum and minimum of relaxation degree of the candidate plans, and $\sum_{i=1}^{n} rd(v_{A_i})$ represents the relaxation degree of the selected plan.

10.4 Overview of Service Composition Techniques

The two main categories of service composition methods are (1) heuristic methods and (2) mathematical programming methods. Chan et al. [7] discussed AI planning methods under classical planning, neoclassical planning, heuristic control, and AI planning under uncertainty categories. Among the mathematical programming methods, integer programming and dynamic programming are the most popular. Within each sub-category of these composition methods, there are smaller classes of methods for service composition planning (see Table 10.3).

As for mathematical programming approaches, in 1999, Vossen et al. [8, 9] and Kautz [10] initiated the ILP (Integer Linear Programming)-based approach to AI (Artificial Intelligence)-planning problems. Zeng et al. [6] reported a multiple criteria ILP model that can be used for a general case of web service composition. Gao et al. [11] model service composition by using integer programming. In 2008, Rao et al. [12] applied linear logic theorem proving to the web service composition problem. This approach considered not only functional attributes but also non-functional ones in composing web services, which is the same objective as our integer linear programming based methodology. This approach assumes that core services are already selected by the user; however, their functionality does not completely match the user's requirement. Therefore, the approach allows partial involvement of the decision maker in the solution generation procedure.

Heuristic methods are popular in the current planners, as they can find an acceptable solution in a large decision domain in a timely manner. In this chapter, the semantic matching algorithm and semantic casual link matrices will be described in detail since they consider the semantic issue in planning, which is important in web service composition. The mathematical programming methods find the optimal plans for customers' queries. Some of the mathematical programming methods only consider the QoS issue in the domain consisting of the services with the same functions; others consider both functional and QoS issues for heterogeneous services.

In addition, miscellaneous service planning works were reported in the past decade. In 2003, Sirin et al. [13] proposed a prototype version of a semi-automatic method for web service composition. Their method provides possible web services to users at each step of the composition through matching web services based on

Table 10.3 The classification of web service composition techniques

Class	Sub-class	Detailed sub-class
Heuristic methods	Deterministic planning methods	1. State-space based planning [10, 24] 2. Plan-space based planning [17, 18] 3. Graph based planning [17] 4. Propositional satisfiability planning [7] 5. Constraint satisfaction planning [7] 6. Domain-independent heuristic control planning [7] 7. Rule based control planning [7] 8. Hierarchical task network planning [7] 9. Situation/event calculus planning [7] 10. Logic based planning [12] 11. Temporal planning [7] 12. Planning with resources [7] 13. Model checking based planning [7] 14. Case-based planning [7] 15. Agent-based planning [7] 16. Plan merging and rewriting [7] 17. Abstraction hierarchies [7] 18. Semantic matching algorithm [29] 19. Semantic casual link matrices [13]
	Planning under uncertainty	1. Case-based planning with uncertainty [16] 2. Domain analysis in time [20] 3. Planning based on Markov decision processes (Also see dynamic mathematical programming) [42] 4. Multi-agent based planning [15]
Mathematical programming	Integer programming	1. Single objective [8–11] 2. Multi objective programming [6] 3. Multi objective goal programming [28, 33]
	Markov Dynamic programming	1. Classical dynamic programming [42] 2. Nested dynamic programming [41] 3. Discounted dynamic programming [41]

functional properties as well as filtering out based on non-functional attributes. In this method, users are involved in the composition process. In 2004, Xiao et al. [14] proposed an automatic mapping framework based on XML document type definition structure. In 2005, Huang et al. [15] proposed a progressive auction based market mechanism for resource allocation that allows users to differentiate service value and ensure resource acquisition latency. Allocation rule, pricing rule, bidding strategy, etc. were explored in this work. Hwang et al. [16] proposed a dynamic web service selection model to determine a subset of web services to be invoked at runtime so as to successfully compose the required web service. Pacific et al. [17] presented an architecture and prototype implementation of performance management of cluster-based web services which can allocate server resources dynamically. Oh et al. [18, 19] present AI planning-based framework which enables automatic composition of web services, and developed a novel web service benchmarking tool. In 2007, Zhang et al. [20] proposed an architectural framework which enables technology for business service analyzers to compose and adapt heterogeneous services by pattern identification. Oh et al. [21] applied large-scale network techniques in web service analysis. Montagut et al. [22] addressed the security features on executing distributed web services. Lufei et al. [23] proposed an adaptive secure access mechanism as well as an adaptive function invocation module in mobile environments. Phan et al. [24] discussed a similarity-based SOAP multicast protocol to reduce bandwidth and latency in web services with high-volume transactions. The secure information handling in web service platform was studied by Wei et al. [25].

QoS filtering algorithm is to filter the services using the QoS criteria mentioned in Sect. 10.3. Service composition merely QoS filtering has high computational complexity when there are a large number of feasible plans, since it needs an exhaustive search before drawing a conclusion. Moreover, when there is more than one QoS criterion in the planning problem and there is no feasible plan dominating all the other plans, it is difficult to determine which plan is the best for the user. In order to alleviate this problem we develop a network mining based approach, which is described in the next section.

Graph based methods, though used for service composition, do not link the semantic aspects of concepts and services. This linkage, we believe is critical in reducing the computational complexity. This in mind, we develop network based techniques which exploit the relationship between concepts and services.

10.5 Concept-Service Network

Both services and concepts can be nodes, and the input and output information flow between services and concepts can be edges in this network. We can thus construct service networks. Albert et al. showed that the World Wide Web is a small word network (i.e., its average path length depends logarithmically on its size) [26]. Engelen showed that web service network is scale free based on the online service data he collected [27]. The motivation of the analysis in this section is to build the network of services, and use computational tools from network science to help solve the web service composition problem.

The characteristics of the web service network include:

1. The graph is dynamic: Sometimes a sub set of web services may be off line, and new web services may emerge. In addition, some web services may change their functions over time.
2. The maximum distance between two nodes may be infinity, which means that the two nodes have no relationship with each other.
3. The minimum degree (i.e. number of edges) of a service is 2; the minimum degree of a concept is 1.

4. There could be some cycles.
5. There may be some completely connected sub-graphs in the network.
6. The network is directed and asymmetric.
7. The connectivity of the graph is the most informative measure in the web service network, since it is the basis to compose individual web services into compound services.

10.5.1 Constructing the Concept-Service Network

The service network patterns are related with service flow patterns. Similar to the concept of motifs in System Biology, there are four motifs of Service flow patterns. Using the well-accepted terminologies in the web service field. We call them: Sequential (Chain), Switch, Parallel (Feed-forward loop) and Iterative (Feed-back loop) as described in Fig. 10.2.

The four flow patterns are formed depending on how web services share parameters among them. A web service has parameter definition of input flows and output flows (see Fig. 10.4). Based on the semantic information among concepts, the input output relation between a concept and a service, and the information flow among services, we can build a Concept-Service (CS) network.

The steps for building the network include: (1) indexing the services and concepts; (2) generating the relationship among concepts and services in a matrix. We use M to denote the CS network matrix, and the following steps are how to construct the matrix, M.

In matrix M, (a) if a concept is one of the inputs that a service needs, the corresponding element in M, is 1; (b) if a concept is one of the output concepts of a service, the corresponding element in M is 1.We illustrate this procedure in our preliminary work [28].

In Algorithm 1, W is the WSDL file, and G is the OWL file. (W,G) includes the total information of the WSDL file and OWL ontology. A is a container. $ElementScore$ is a temporary variable representing the utility of the relation. C is any concept in G. $ElementCtoS()$ is a sub-function to compute the utility score of the link from a concept to a service. $ElementStoC()$ is a sub-function to compute the utility score of the link from a service to a concept. $ElementCtoC()$ is a sub-function to compute the utility score of the link from a concept to a concept. $Score(C1,C2)$ is the semantic score of the link $C1$ to $C2$.

In this work, the semantic match scores among concepts are derived based on the work of Li and Horrocks [29]. The following definition is used to calculate the score in $ElementCtoC()$.

$$Score(C1,C2) = \begin{cases} 1 \text{ if } C1 = C2 \\ 1 \text{ if } C1 \subset C2 \\ 0 \text{ if } C1 \supset C2 \\ 0 \text{ Otherwise} \end{cases} \tag{10.5}$$

Algorithm 1 Network generation

```
Main function MatrixGen(W,G)
  **for** any service in W **do**
    Index it in A
  **end for**
  **for** any concept in G **do**
    Index it in A
  **end for**
  Initialize ElementScore = 0
  **for** any concept C in G and any service S in W **do**
    $ElementScore = ElementCtoS(C,S)$
  **end for**
  **for** any service S in W and any concept C in G **do**
    $ElementScore = ElementStoC(S,C)$
  **end for**
  **for** any two concepts C1 and C2 in G **do**
    $ElementScore = ElementCtoC(C1,C2)$
  **end for**
  **for** any two services S1 and S2 in W **do**
    $ElementScore = 0$
  **end for**
End

Function ElementCtoS(C, S)
  Parse W
  **if** $I$ is the input set of $S$ **and** $C \in I$ **then**
    $Score = 1$
  **end if**
End

Function ElementStoC(S, C)
  Parse W
  **if** $O$ is the output set of $S$ **and** $C \in O$ **then**
    $Score = 1$
  **end if**
End

Function ElementCtoC(C1, C2)
  $Score = Score(C1,C2)$
End
```

10.5.2 Service Composition Framework in the Concept-Service Network

The CS network we build in the previous section will play an important role in service network mining and service composition.

As shown in Fig. 10.5 (see our work in [28]), we build the network based on the WSDL and OWL ontology files. We analyze the topology of the network.

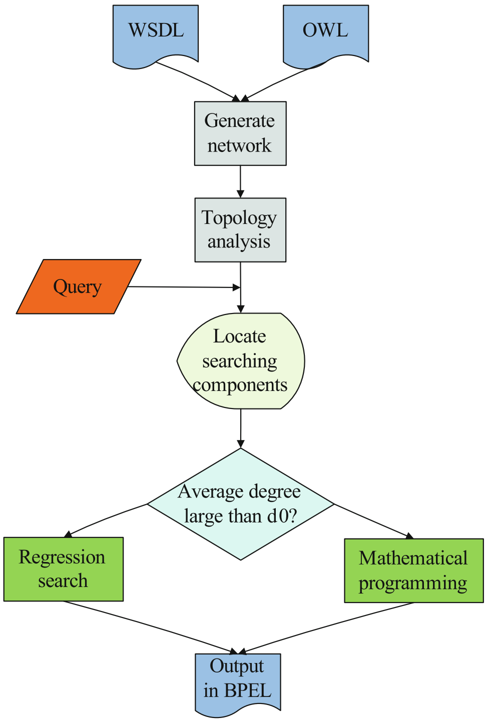


Fig. 10.5 The composition process [28]

The feasible searching region will be selected when a request arrives. Finally, one of the solvers described below can be selected according to the average degree of the feasible sub-networks. When the average degree of a component in the network is large, exhaustive search is time-consuming, so we utilize mathematical programming to find the optimal compound service; otherwise, regression search is a good technique to obtain the optimal solution quickly. It is possible that the domain searched is only one of the connected components in the large-scale network. Next, the strategy of searching for optimal solution differs based on the connectedness of the component. This helps to determine which searching strategy, regression search or mathematical programming is good for the network. The average degree of a component is: $d = \frac{1}{n}\sum_{i=1}^{n} d_i$, where n is the total number of nodes in a component and d_i is the total degree of node i. (1) If the average degree d of the connected component is small $d < d_0$, where d_0 is a threshold to be determined, we use regression search. (2) If the average degree of the connected component is large $d > d_0$, the multi-criteria mathematical integer programming in service composition is adopted. In addition, if the size of the searching sub-network is large, some heuristic method can be further adopted.

10.5.3 Concept-Service Network Analysis

Once we represent the CS network using CS network matrix M, the network analysis may include:

1. Finding the components and bridges between each pair of components in the network. This can help the system to judge the effects of network node and link failures for some queries.
2. Calculate the centrality of services and concepts. This can help the system to identify the important services and popular concepts.
3. Analyze the distance between services. We can use the information for service composition. When the distance of two services is infinity, these two services cannot be composed together.
4. Represent the network's dynamics and analyze the characteristics mentioned above. This can help find the hidden information in the network. The dynamics can include edge growth, node growth, or both. In web service composition, the edge growth dynamics is most informative, which is our focus in the next section.
5. In Fig. 10.6 [28], nodes 7, 12, 8, and 3 belong to component 1 (marked in purple); nodes 1, 2, 4, 5, 6, and 11 are in component 2 (marked in blue); and nodes 9, 10, and 13 together form component 3 (marked in yellow). In this figure, arc (4, 3) is the bridge from the component 2 to component 1. So, arc (4, 3) is important, if we need to integrate component 1 and component 2 in order to form compound services that satisfy some customers' requests.

As the specific concepts and techniques used in network analysis may not be familiar to all the readers, we review them next.

10.5.3.1 Review of Network Science

This section is organized as follows: First, we briefly review network metrics, network models and network clustering. We then introduce the service network mining techniques that we developed.

Characterization of networks

The structure of networks conveys rich information useful for inference. The last decade has seen a proliferation of topological metrics. We review some of the important ones here. The order of a network is the total number of nodes (also called vertices), and its size is the total number of links (also called edges) in a network. The degree of a node is the number of links connecting the node to its neighbors. The incoming (in) degree and outgoing (out) degree sum up to the degree of a node. The degree distribution is a 2-dimensional graph showing the frequency of nodes with different degrees in the network. The network density is the ratio between network size and the maximum possible number of links. One of the most

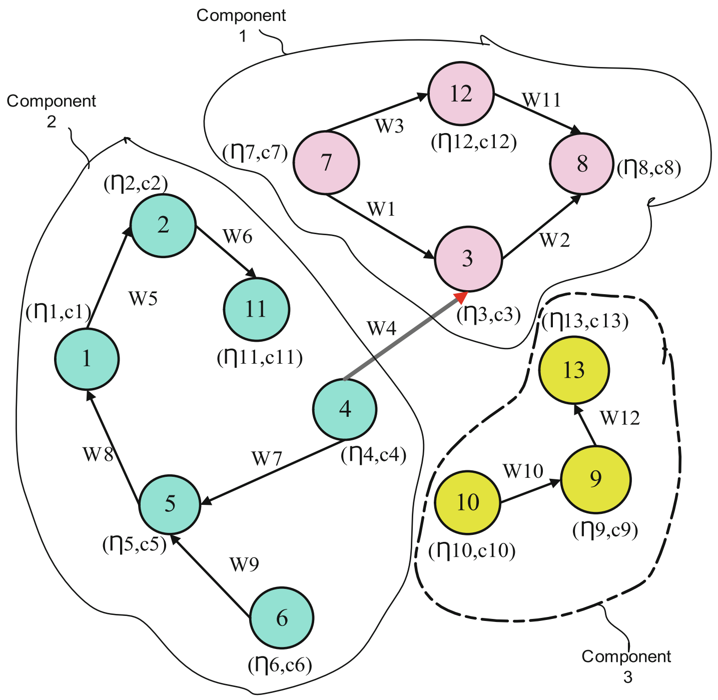


Fig. 10.6 A small network of web services [28]

important measures is the distance. The distance between two nodes is the length of the shortest path between them, i.e. the minimum number of links that one needs to follow when going from one node to the other. The shortest path can be found through Dijkstra's algorithm. The average path length of a network is the average value of distance between any pair of nodes in the network.

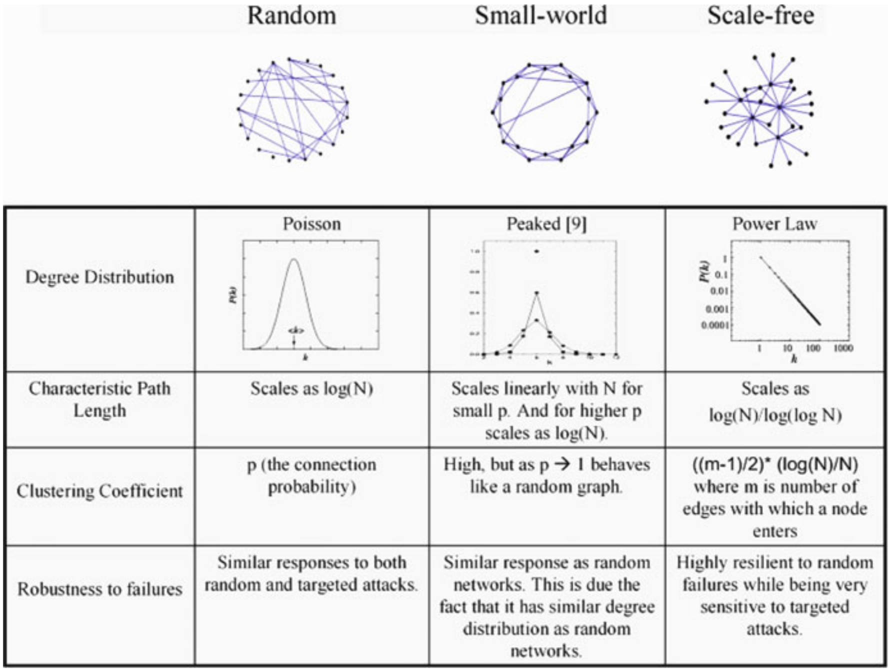
The diameter of the network is the longest distance between any pair of nodes in a network. The clustering coefficient of a node is the number of triangles centered at the node divided by the number of triples centered at the node. The clustering coefficient of a network is the arithmetic mean of the clustering coefficients of all the nodes. The betweenness centrality quantifies how much a node is between other pairs of nodes. Let us define the ratio between the clustering coefficient and the average path length as the CP ratio. The proximity ratio of a network is the CP ratio between this network and a random network. This property captures the extent of a network's small-worldness. The efficiency of a network is the communication effectiveness of a networked system (global) or of a single node (local). The mixing coefficient is the Pearson correlation between the degrees of neighboring nodes. The modularity index measures the topological similarity in the local patterns of linking. Table 10.4 summarizes the most commonly used metrics and their equations [30, 31].

Table 10.4 Metrics for networks [33]

Metric	Math equation (in undirected graph)
Order	The number of nodes n
Size	$m = \sum_i \sum_j a_{ij}$, where $a_{ij} = \begin{cases} 1 & \text{node } i \text{ and } j \text{ are linked} \\ 0 & \text{otherwise} \end{cases}$
Node degree	$d = \sum_i a_{ij}$, where $a_{ij} = \begin{cases} 1 & \text{node } i \text{ and } j \text{ are linked} \\ 0 & \text{otherwise} \end{cases}$
Density	$\delta = \frac{2m}{n(n-1)}$, where m is the number of arcs in the network and n is the number of nodes in the network
Average path length	$l = \frac{1}{n(n-1)} \sum_{i \neq j} d_{ij}$, where d_{ij} is the distance between node i and node j, and n is the number of nodes in the network
Diameter	$D = \max\{d_{ij}\}$, where d_{ij} is the distance between node i and node j, and n is the number of nodes in the network
Clustering coefficient of a node	$C_i = \frac{2t_i}{k_i(k_i-1)}$, where t_i is the number of triangles centered at node i, and k_i is the degree of node i
Clustering coefficient of a network	$C = \frac{1}{n} \sum_i C_i$, where C_i is the clustering coefficient of node i
Betweeness centrality	$B_k = \sum_{k \in p(i,j)} \frac{1}{n_{ij}}$, where n_{ij} is the number of path from node i to node j
Proximity ratio	$\mu = \frac{\frac{C}{l}}{\frac{C_{\text{rand}}}{l_{\text{rand}}}}$, where C is the clustering coefficient of the network, and l is the average path length of the network; C_{rand} is the clustering coefficient of a random network and l_{rand} is the average path length of a random graph
Efficiency	$E_{\text{global}} = \frac{1}{n(n-1) \sum_{i \neq j} \frac{1}{d_{ij}}}$, where d_{ij} is the distance between node i and node j
Mixing coefficient	$r = \frac{\sum_i (dg_i - \bar{dg})(dn_i - \bar{dn})}{\sqrt{\sum_i (dg_i - \bar{dg})^2 (dn_i - \bar{dn})^2}}$, where dg_i is the degree of node i, and dn_i is the first order mean degree of the neighbors of node i. The mixing coefficient is the standard devotion
Modularity index	$Q = \sum_i (e_{ii} - a_i)^2$, where e_{ii} is the fraction of edges in subgraph i; a_i is the fraction of edges between this subgroup and all other subgraph

Network models

There are three popular network models: (1) random networks, (2) small world networks, and (3) scale free networks. The mathematicians Erdös and Rényi analyzed *Random networks* based on a set of nodes (ER model [38]). In this network, the links are added into a network consisting of n nodes and with no links among them. The link between any pair of nodes is placed with a probability of p. The degree distribution of the network follows Poisson distribution with the average node degree of $<k>= np$. *Regular networks* include rings, lattices, trees, stars, and complete graphs. A *ring* is a connected graph in which a node is linked exactly with two other nodes. A *lattice* is a graph in which the nodes are placed on a grid and the neighbors are connected by an edge. A *tree* is a connected graph containing no cycle. A *star* graph is a tree in which every node is connected to the root.



	Random	Small-world	Scale-free
Degree Distribution	Poisson	Peaked [9]	Power Law
Characteristic Path Length	Scales as log(N)	Scales linearly with N for small p. And for higher p scales as log(N).	Scales as log(N)/log(log N)
Clustering Coefficient	p (the connection probability)	High, but as p → 1 behaves like a random graph.	((m-1)/2)* (log(N)/N) where m is number of edges with which a node enters
Robustness to failures	Similar responses to both random and targeted attacks.	Similar response as random networks. This is due the fact that it has similar degree distribution as random networks.	Highly resilient to random failures while being very sensitive to targeted attacks.

Fig. 10.7 Comparison of random network, small-world, and scale-free networks [32]

In a *full (complete)* graph, there is an edge between all pairs of nodes. One of the most important network models is the *small-world network*. The *small-world [39]* concept describes the fact in a network that regardless of size there is a relatively short path between any two nodes. The small world networks are the networks with low average shortest path length and a high clustering coefficient, which therefore are in a sense situated between regular and random networks. The average path length of this network scales in $\log_{<k>} n$. Here, $< k >$ is the average out-degree and n is the number of nodes. When pairs are selected uniformly at random, they are connected by a short path with high probability. Many real world networks exhibit this property. Random networks also have small average distances, however they are not clustered. Small-world networks usually appear in social sciences. Many systems in real world are dynamic and the order of the networks (number of nodes) grows over time. The concept of a scale-free network was introduced by Barabási and Albert (BA model [37]) in 1999. In a scale-free network, the number of nodes n is expected to change over time. In BA model, the network dynamics is described by introducing new nodes into an existing network. When a vertex is linked in, it tends to link with higher probability to a vertex that already has a large number of edges, which is the so called *preferential attachment*. A comparison of random, small-world and scale free networks is shown in Fig. 10.7. For detailed comparison of the three types of networks, please refer to [32].

Clustering

Clustering [40] is the process of organizing objects into groups whose members are similar within a group and dissimilar to the objects in other groups based on some metric. There are four categories of clustering techniques: (1) Exclusive Clustering. (2) Overlapping Clustering. (3) Hierarchical Clustering. (4) Probabilistic Clustering. In an *exclusive clustering* process, a datum belongs to a definite cluster and could not be included in any other cluster. In an *overlapping clustering* process, a datum may belong to two or more clusters with different degrees of membership. In a *hierarchical clustering* process, we begin with the condition that every datum is a cluster, and keep merging the two nearest clusters at a time, until we reach the final number of clusters needed. In a *probabilistic clustering* process, we cluster data based on a mixture of probability distributions.

We give a typical clustering method as an example in each category: K-means (Exclusive), Fuzzy C-means (Overlapping), Hierarchical clustering (Hierarchical), and Mixture of Gaussian (Probabilistic).

1. *K-means method*
   The steps of K-means clustering are: (1) Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids. (2) Assign each object to the group that has the closest centroid. (3) When all objects have been assigned, recalculate the positions of the K centroids. (4) Repeat Steps (2) and (3) until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.
2. *Fuzzy C-means method*
   Fuzzy C-means method is derived from K-means algorithm by allowing a datum to belong to multiple clusters with a membership degree.
3. *Hierarchical clustering method*
   The steps of hierarchical clustering are: (1) Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$. (2) Find the least dissimilar pair of clusters in the current clustering, say pair (r), (s), according to $d[(r),(s)] = \min d[(i),(j)]$, where the minimum is over all pairs of clusters in the current clustering. (3) Increment the sequence number: $m = m+1$. Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to $L(m) = d[(r),(s)]$. (4) Update the proximity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k) is defined in this way: $d[(k),(r,s)] = \min d[(k),(r)], d[(k),(s)]$. (5) If all objects are in one cluster, stop. Else, go to step (2).
4. *Mixture of Gaussian method*
   Mixture of Gaussian method chooses the component at random with probability $P(\omega_i)$. It samples a point $N(\mu_i, \delta^2 I)$. Suppose we have $x_1, \ldots, x_N$ and

Table 10.5 Comparison among the typical methods

Algorithm	Pros.	Cons.
K-means	• Simple unsupervised learning process	• The results produced depend on the initial values for the means, and it frequently happens that suboptimal partitions are found • The results depend on the metric used to measure distance • The results depend on the value of k
Fuzzy C-means	• Allows data to belong to more than one clusters	• Needs to decide number of clusters
Hierarchical	• Does not need to decide the number of clusters at the beginning	• Does not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects • Not easy to undo the clusters in previous steps
Gaussian	• Well-studied statistical inference techniques • Flexibility in choosing the component distribution • Obtains a density estimation for each cluster • Fuzzy classification is available	• Needs to decide the number of clusters and component distribution

$P(\omega_1),\ldots,P(\omega_K)$, we can obtain the likelihood of the sample $P(x|\omega_i,\mu_1,\ldots,\mu_K)$. Therefore, we minimize

$$P(X|\mu_1,\ldots,\mu_K)=\prod_{j=1}^{N}\sum_{i=1}^{N}P(\omega_i)P(x_j|\omega_i,\mu_1,\ldots,\mu_K) \tag{10.6}$$

We summarize the properties of the four typical clustering methods in Table 10.5.

10.5.3.2 Concept-Service Network Mining

CS network representation

First, let us review the procedure of constructing CS network we described in Sect. 10.5.1. The steps in building the CS network include: (1) Index the services and concepts. (2) Generate the relationship among concepts and services in a matrix. Let this matrix be M. We can generate the elements in matrix using the following criteria: (a) If a concept is one of the input concepts that a service needs, use value 1 as the corresponding element in. (b) If a concept is one of the output concepts of a service, use value 1 as the corresponding element in. (c) All other elements are Zero. Cui et al. describe the computer program to generate CS network matrix [28].

$$M_{(n+m)\times(n+m)} = \left[\begin{array}{ccc|ccc} M_{11} & \cdots & M_{1n} & M_{1(n+1)} & \cdots & M_{1(n+m)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nn} & M_{n(n+1)} & \cdots & M_{n(n+m)} \\ \hline M_{(n+1)1} & \cdots & M_{(n+1)n} & M_{(n+1)(n+1)} & \cdots & M_{(n+1)(n+m)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M_{(n+m)1} & \cdots & M_{(n+m)n} & M_{(n+m)(n+1)} & \cdots & M_{(n+m)(n+m)} \end{array}\right] \quad (10.7)$$

In M, $M_{ii} = 1$, $i = 1,...,n+m$ which means any service or concept can reach themselves; $M_{ij} = 0$, $i = n+1,...,n+m$, $j = n+1,...,n+m$ which means there is no direct relation between service and concept in the initial matrix.

Let $C_{n\times n}$ be the incidence matrix of concepts derived from semantics; let $I_{n\times m}$ be the incidence matrix of concepts to services by Web Service Description Language (WSDL); let $O_{m\times n}$ be the incidence matrix of services to concepts from WSDL; and let $S_{m\times m}$ be the incidence matrix of services. We initialize $M_{ij} = 0$, $i = m+1,...,m+n$, $j = m+1,...,m+n$, that is $S_{n\times n} = (0)_{n\times n}$ in M initially. Thus,

$$M_{(n+m)\times(n+m)} = \begin{bmatrix} C_{n\times n} & I_{n\times m} \\ O_{m\times n} & S_{m\times m} \end{bmatrix} \quad (10.8)$$

The CS matrix can be normalized column-wise. $M = [\frac{M_1}{M_{11}},...,\frac{M_{n+m}}{M_{(n+m)1}}]$, where $M_1,...,M_{n+m}$ are the column vectors in M. After normalizing the columns, the rows can also be normalized selectively: If the L_1 norm of any row vector in M is greater than 1, we divide each element of the row by the L_1 norm of that row; otherwise, the row is kept the same as before. Let us still denote this normalized CS matrix M. Thus, M has the following property:

Theorem 10.3. ***Properties of*** *M: (1)* $\sum_{i=1}^{m+n} M_{ij} \leq 1$, $j = 1,2,...,m+n$; *(2)* $\sum_{j=1}^{m+n} M_{ij} \leq 1$, $i = 1,2,...,m+n$.

Theorem 10.4. *If CS matrix M has properties (1) and (2), the matrices* M^i, $i = 1,2,...$ *also have properties (1) and (2).*

Definition 10.22. The potential of a Concept-Service network M is defined by $G^*_{(m+n)\times(m+n)} = \left[\begin{array}{c|c} C^*_{m\times m} & I^*_{m\times n} \\ \hline O^*_{n\times m} & S^*_{n\times n} \end{array}\right]$, where $G^* = \lim_{t\to\infty} \sum_{i=1} M^i = (I-M)^{-1} - I$.

Theorem 10.5. ***Convergence of CS matrix potential**: If M satisfies properties (1) and (2), M^i, $i = 1,2,\dots$ converges to zero, e.g.,* $\lim_{i\to\infty} M^i = 0$.

CS network dynamics

In this section, a study of a Boolean dynamic network is undertaken. A Boolean dynamic network is a network with the status of its nodes and edges changes in time, and both nodes and edges in it can only have two states: on and off (or 1 and 0). In the CS network, the procedure of calculating the CS matrix potential is the same as growing the edges via the following dynamics:

$$e_{ij}(t) = \begin{cases} 1 \text{ if } l_{ij} \le t \\ 0 \text{ otherwise} \end{cases} \tag{10.9}$$

where $e_{ij}(t)$ is the edge from node i to node j at step t, and l_{ij} is the shortest path length from node i to node j. This dynamics shows that whether an edge $e_{ij}(t)$ exists or not depends on whether the shortest path length from node i to node j is equal to or less than t or not. The above edge growth dynamics can be further studied by the following boolean dynamics:

$$N_j(t+1) = \begin{cases} N_i(t) \text{ if } e_{ij}(t) = 1 \\ N_j(t) \text{ if } e_{ij}(t) = 0 \end{cases} \tag{10.10}$$

$$e_{kj}(t+1) = \begin{cases} 1 \text{ if } N_j(t+1) = 1 \\ 0 \text{ if } N_j(t+1) = 0 \end{cases} \tag{10.11}$$

Equation (10.10) shows how to update the node status in a network. $N_j(t)$ is the status of node j at time t. $N_j(t) = 1$ means the node j is on, and $N_j(t) = 0$ means the node j is off. Equation (10.11) shows how to update the edge status in a network. $e_{ij}(t)$ is the status of the edge between node i and node j. $e_{ij} = 0$ means the edge between node i and node j does not exist at time t; $e_{ij} = 1$ means the edge between node i and node j is on at time t. Algorithm (2) shows how to calculate the potential of CS network via boolean dynamic network.

The node status in (10.10) is an intermediate step for updating edge status in (10.11). We are only interested in edge growth in CS network. An example service network with 45 concept nodes and 35 service nodes is constructed. To generate the initial network, we assume that the connectivity probability among nodes is $p = 0.01$; the connectivity probability between concepts and web services is $p = 0.04$. And there is no arc among web services. No self-loop is allowed in the network. However, there are only a few links among concept nodes in this network (see Fig. 10.8). In this network, there are some isolated concept nodes not used by any service, which means that they are not important for the network in reality.

Many network metrics can be used in analyzing network properties [30]. By applying centrality analysis we can see that node 44 is the most frequently used

Algorithm 2 Calculate CS network potential by Boolean networks

```
Main function CS-Boolean(E,V)
  for any node $k$, $k = 1,2,\dots,m+n$ do
    $N_k(1) = 1, N_j(1) = 0, j \neq k, j = 1,2,\dots,m+n$
    t=1
    repeat
      upgrade node status using Equation (10.10)
      upgrade edge status using Equation (10.11)
      $t \leftarrow t+1$
    until any node $N_j(t+1) = N_j(t), j = 1,2,\dots,m+n; j \neq k$
  end for
End
```

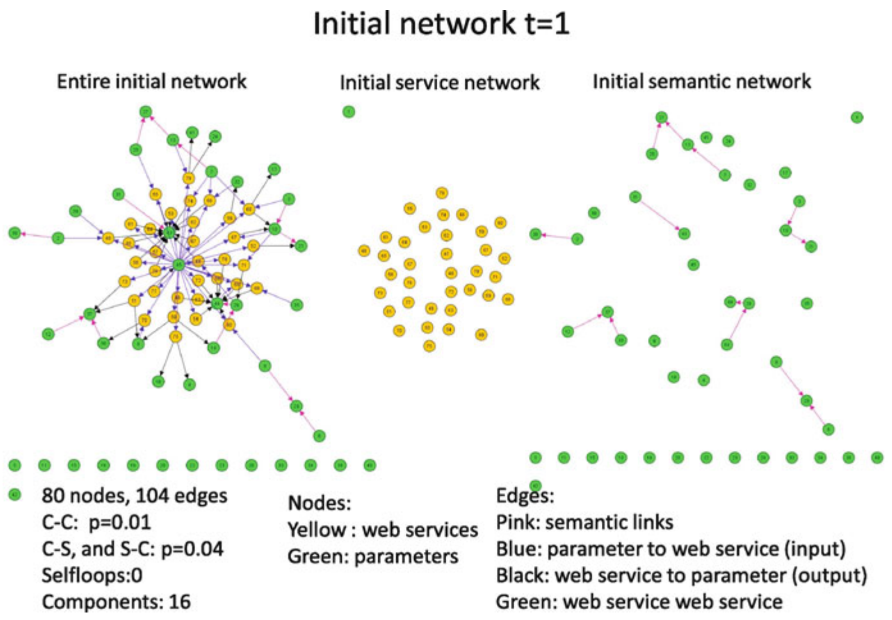


Fig. 10.8 Initial service network generation

input concept in the network and nodes 43 and 45 are the most often used output concepts in the network (Fig. 10.9).

In Fig. 10.10, the edge growth at each step of $t = 1, 2, 3, 4, 5, 6, 7$ is shown. The picture only shows the new edges added in at the current step. We notice that self-loops occur at step $t = 2$. This means that there are cycles of length 2 in the initial network. The network will not reach a steady state within a finite number of steps. However, if there are no cycles in the original network, the web service network will converge with finite rounds of propagations. For example, the experiment on the network in Fig. 10.10 it converges at step $t = 5$.

With respect to the original network in Fig. 10.11a and its degree centrality in Fig. 10.11b, we can see that the web services start to have links among them at

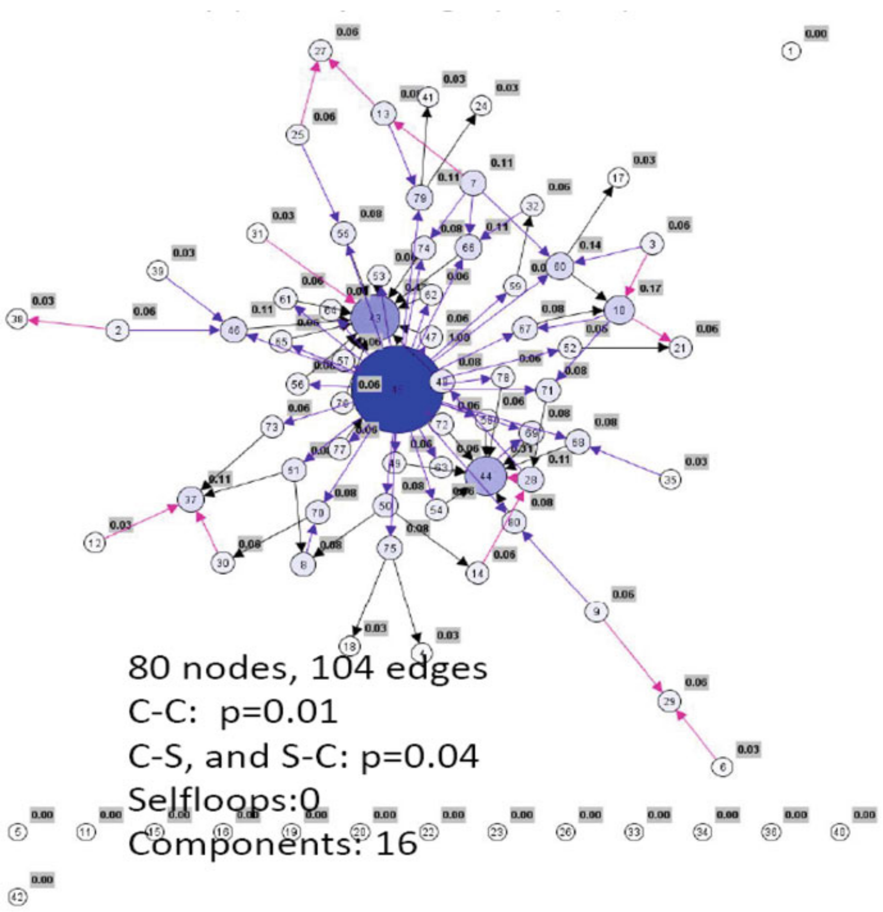


Fig. 10.9 Centrality analysis of the network

$t = 2$. We observe that more semantic links are built among the concepts after a few steps.

At step $t = 7$ as shown in Fig. 10.12, more web services get connected, and the se-mantic network is well developed. The number of edges increased from 104 to 260. The connection probability among concepts is between 0.06 and 0.23, and the connection probability among services is between 0.03 and 0.14. The overall connection probability of nodes is between 0.02 and 0.99. There are a few very popular concepts and services in the network. The number of self loops is 6. The number of components does not decrease in this case.

As we can see from Fig. 10.12, popular web services and concepts are: node 77, node 43, node 44, node 45, node 69, node 7, etc., which cannot be seen in the

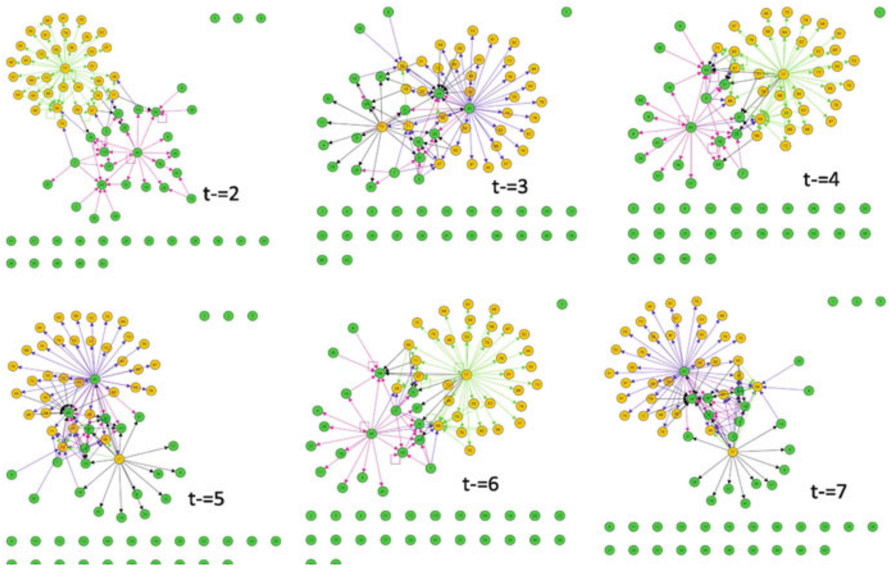


Fig. 10.10 Edge growth in the network within 7 steps

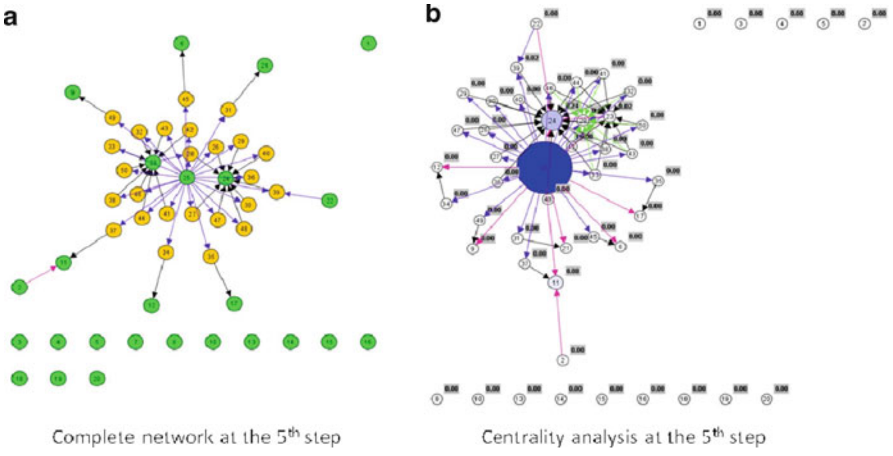

a
b
Complete network at the 5th step
Centrality analysis at the 5th step

Fig. 10.11 A sample graph reaching steady state with 5 propagations

original network, although, nodes 43, 44, and 45 are already shown to be popular. So, the network analysis can lead to unearthing such hidden information. This is useful in service computing, robustness analysis and in rescuing the query from catastrophic failures.

As we can see in Fig. 10.13 the number of nodes remains the same; however, the network is still growing in the number of edges. The sub-graph of services is

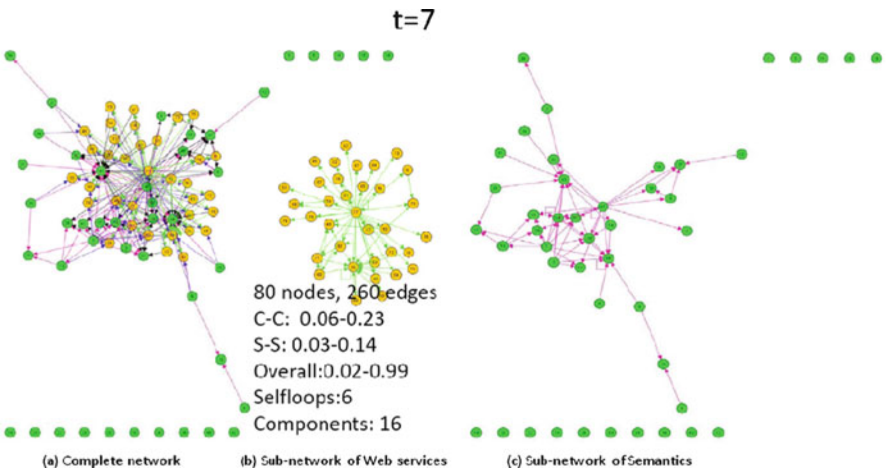


Fig. 10.12 Network at step $t = 7$

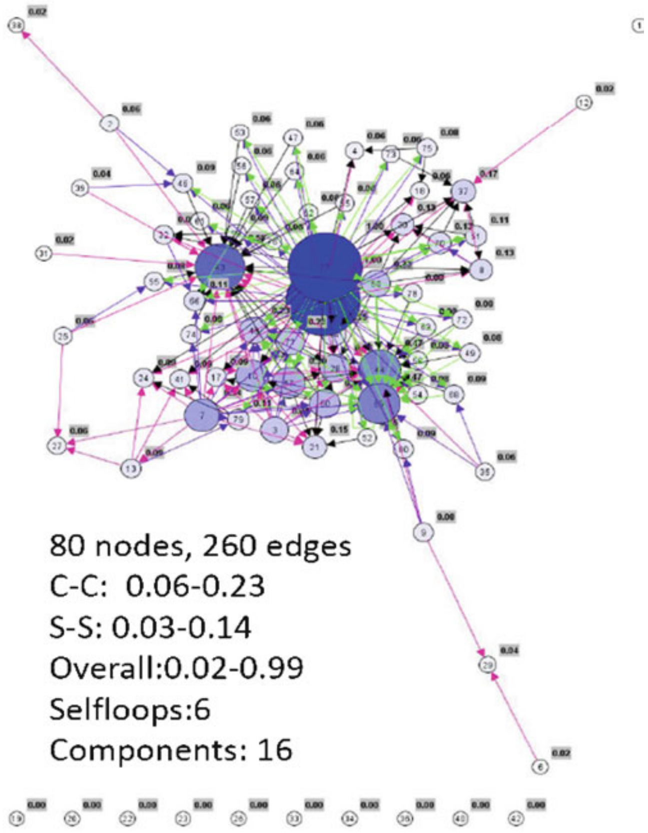


Fig. 10.13 Centrality analysis of the entire network at step $t = 7$

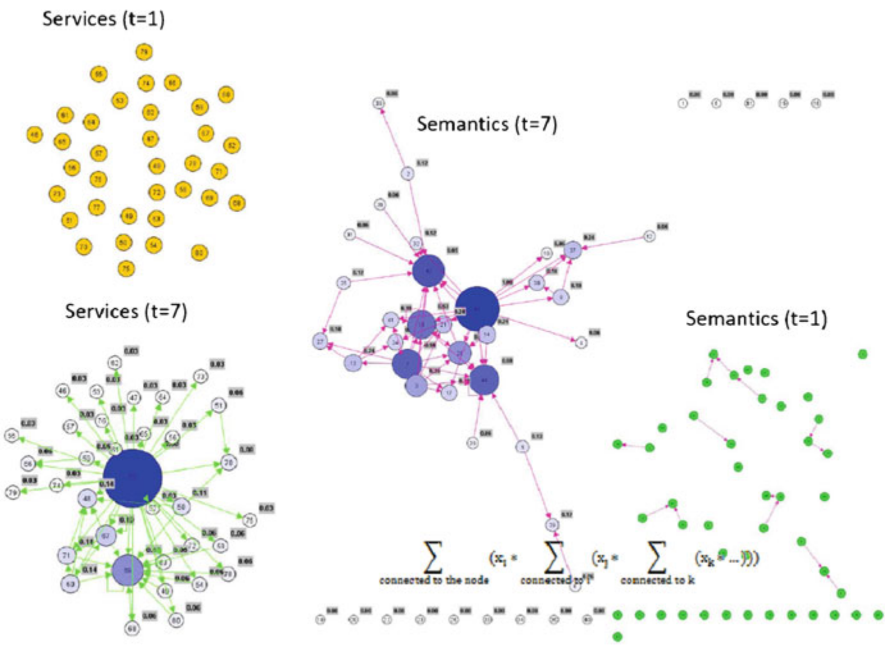


Fig. 10.14 Comparison of sub-network among services and the sub network of concepts

originally disjointed and after a few steps, the relationship is built among them. The semantic connection among concepts is sparse in the initial network, and is well developed in the network at step $t = 7$. In the meantime, new popular services and concepts are added. These predecessors of the nodes with a high out-degree have the potential to be popular when the network evolves. The potential of a node to be popular can be approximately estimated by the following equation

$$\text{potential of node} = \sum_{\text{connected to the node}} x_i \cdot \Big(\sum_{\text{connected to node } i} x_j \cdot \Big(\sum_{\text{connected to node } j} (x_k \cdot \ldots)\Big)\Big) \tag{10.12}$$

So, analyzing network evolution can help us to predict the long-run network structure conveniently. The separate service sub-graph and concept sub-graph at step 7 and step 1 are shown in Fig. 10.14. The accuracy of prediction depends on the accuracy of the Boolean rules used to describe the real network.

10.5.4 Optimization in the Concept-Service Network

We need further formulate the service composition problem as mathematical program in order to find a set of services and the relation among them, denoted as $H(F)$, so that the cost of the solution can be minimized, the reliability of the solution can be maximized, and the delivery time of the solution can be minimized:

$$\begin{aligned} &\min P(H(F)(X_0)) \\ &\max R(H(F)(X_0)) \\ &\min T(H(F)(X_0)) \\ \text{s.t.} \quad &H(F)(X_0) \geq X_1 \end{aligned} \tag{10.13}$$

where $q(X_0,X_1)$ is a given query, $P(H(F)(X_0))$ is the cost of the service composition; $R(H(F)(X_0))$ is the reliability of the service composition; $T(H(F)(X_0))$ is the delivery time of the service composition. It will be preferred if other QoS attributes can also be optimal in the solution. This mathematical optimization problem will be discussed in this section.

10.5.4.1 Multi-Criteria Programming with Pure Real Constraints

We define the entire list of variables and parameters in this section. The Quality of Services (QoS) metric is used to evaluate services. In this context, cost, execution time and reliability are used to demonstrate the model. Hence, the QoS vector for a service can be expressed as: "quality of service(s) = f(cost(s), service execution time(s), reliability(s))." Cost is the expense for purchasing a service or services; service execution time is the process time of a service or services; reliability is a measure of successfully running a service or a set of services. The entire list of variables used is shown in Table 10.6.

In general, the number of attributes in the input set I and the number of attributes in the output set O are different. However, it is reasonable to let $n = max\{h,k\}$ be the total number of attributes since most of the attributes are the inputs of some services and the outputs of other services at the same time. Generally speaking, all the attributes can be inputs in some services and outputs in other services. Thus, it can be proved that $I = O$ approximately, in large scale service networks. In order to define the reliability score for web services, we give the following definition. The reliability of a service is a function of failure rate of the service.

If the failure rate of service Z is f, the reliability score of the service is defined as: $q(f) = -\log(f)$, where $0 < f \leq 1$ and $0 \leq q(f) < +\infty$.

Here, we introduce reliability measure $q(f)$ in terms of the failure rate f. This technique is useful to convert the nonlinear objective function into linear objective function, which simplifies the problem. LP(linear programming) solvers can be used to solve the model. Next, we need to specify a depth level of composition before using this mathematical programming model. The decision about L, the depth level, is important as larger or smaller L influences the computational time and whether the optimal solution can be obtained or not.

Table 10.6 Definition of variables and parameters [33]

Variable	Definition
Z	A set of web services
I	A set of input attributes of the web services
O	A set of output attributes of the web services
m	The number of services in Z
n	The number of attributes for the services in set Z
L	The maximal number of composition levels
Z_{lj}	Web service that is currently available in the database, $Z_{lj} \in Z$; $j = 1,\ldots,m, l = 1,\ldots,L$
I_{ij}	The ith input attribute of service Z_j; $i = 1,\ldots,n$, $j = 1,\ldots,m$
O_{ij}	The ith output attribute of service Z_j; $i = 1,\ldots,n$, $j = 1,\ldots,m$
p_j	The fixed price for acquiring the service from Z_j; $j = 1,\ldots,m$
t_j	The execution time of service Z_j; $j = 1,\ldots,m$
f_j	The failure rate of service Z_j; $j = 1,\ldots,m$
q_j	The reliability of service Z_j; $j = 1,\ldots,m$
C_0	The maximum total cost that the customer is willing to pay for the services
T_0	The maximal total execution time that the customer allows to accomplish the entire process of services
Q_0	The minimal reliability that the customer allows for a service in the composition
Q_1	The minimal overall reliability that the customer allows for the entire service complex, where $Q_1 > Q_0$

Among all the variables we defined, the decision variables are Z_{ij}, the status of the jth web service in the lth level of composition, $j = 1,\ldots,m$, $l = 1,\ldots,L$.

$$Z_{ij} = \begin{cases} 1 \text{ web service } Z_j \text{ is selected in the } lth \text{ level} \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise} \end{cases} \quad \text{where } j = 1,2,\ldots,m; l = 1,2,\ldots,L. \tag{10.14}$$

The objective function is defined as follows: *Cost (criterion No. 1)*: The cost of the service composition equals to the sum of the prices of the services in the composition.

$$\min \sum_{l=1}^{L} \sum_{j=1}^{m} Z_{ij} \cdot p_j \tag{10.15}$$

Service execution time (criterion No. 2): The execution time is the total processing time for executing the entire series of services. We assume that the services at one level are executed in parallel:

The maximum execution time of the services in the 1st level is $\max_j\{t_j \cdot Z_{1j}\}$.
The maximum execution time of the services in the 2nd level is $\max_j\{t_j \cdot Z_{2j}\}$.
The maximum execution time of the services in the lth level is $\max_j\{t_j \cdot Z_{lj}\}$.
So, the total service execution time of this composition is: $\sum_{l=1}^{L} \max_j\{t_j \cdot Z_{lj}\}$.

Let η_l be the maximum service execution time of the lth level. The above total service execution time expression can be reformulated in terms of the following linear program:

$$\begin{gathered}\min \textstyle\sum_{l=1}^{L} \eta_l \\ \text{subject to} \\ \eta_l - t_j \cdot Z_{lj} \geq 0 \\ \text{where } j = 1,2,...,m; l = 1,2,...,L\end{gathered} \tag{10.16}$$

Reliability (criterion No.3): The reliability of the service composition is described by the summation of the reliability scores of all the services included in the composition.

$$\max \sum_{l=1}^{L} \sum_{j=1}^{m} Z_{ij} \cdot q_j \tag{10.17}$$

The validity of the first and the second criteria are obvious as we stated above. However, we need to validate the third criterion according to Definition 1. It can be proven that criterion No.3 is valid.

Let S be the set of services appearing in a composition, $S \subseteq Z$. Then, expression (4) equals to

$$\max \sum_{Z_j \in S} q_j \tag{10.18}$$

where $\sum_{Z_j \in S} q_j = -\sum_{Z_j \in S} (\log f_j) = -\log(\prod_{Z_j \in S} f_j)$.

Since function $y = -\log(x)$ monotonically decreases in x when $x \in (0, +\infty)$.

Now we have $\max \sum_{Z_j \in S} q_j$ equals to

$$\min \prod_{Z_j \in S} f_j \tag{10.19}$$

where $\prod_{Z_j \in S} f_j$ is the overall failure rate of the composition.

Thus, $\max \sum_{Z_j \in S} q_j$ equals to $\min \prod_{Z_j \in S} f_j$.

So, maximizing the summation of the reliability scores of the services in the composition equals to minimizing the product of the failure rates of the services in the composition. Here is how the constraints are constructed:

1. Input constraints: An input attribute of the query service should be included in the input attributes of the selected services in the composition. Thus,

$$\sum_{l=1}^{L} \sum_{j=1}^{m} I_{ij} \cdot Z_{lj} \geq I_{i0} \qquad i = 1,2,...,n \tag{10.20}$$

   This constraint can be neglected, if we allow some redundancy in the inputs provided by customers.
2. Output constraints: An output attribute of the query should be included in the output attributes of the selected services in the composition. Hence,

$$\sum_{l=1}^{L}\sum_{j=1}^{m} O_{ij}\cdot Z_{lj} \geq O_{i0} - I_{i0} \qquad i = 1,2,...,n. \tag{10.21}$$

3. The relationship of the outputs and inputs between the levels has to satisfy the following requirements.

   All the inputs of the selected services in the first level must be a subset of the initial input set given in the query.

$$\sum_{j=1}^{m} I_{ij}\cdot Z_{1j} \leq I_{i0} \qquad i = 1,2,...,n. \tag{10.22}$$

   Also, all the input sets of selected services at the *k*th level must be a subset of the union of the initial input set given in the query and the output sets of services in previous levels. The formulation is

$$\sum_{j=1}^{m} I_{ij}\cdot Z_{k+1,j} - \sum_{l=1}^{k}\sum_{j=1}^{m} O_{ij}\cdot Z_{lj} \leq I_{i0} \qquad k = 1,2,...,L-1; i = 1,2,...,n. \tag{10.23}$$

   The relation among the inputs of services in *k*th level and the outputs from the previous levels and the attributes given in the query needs to satisfy equations (10.4)–(10.11).

4. Goal constraint on the total cost: The customer hopes that the total cost should not exceed C_0.

$$\sum_{l=1}^{L}\sum_{j=1}^{m} Z_{lj}\cdot p_j \leq C_0 \tag{10.24}$$

5. Constraint on the service execution time: The customer hopes that the total service execution time should not exceed T_0. Since some services can be executed in parallel, we take the longest execution time as the execution time of the set of services executed in parallel. The execution time of the composition, e.g. total service execution time is the sum of the service execution times of *L* levels. Thus,

$$\eta_l \geq Z_{lj}t_j \qquad j = 1,2,...,m; l = 1,2,...,L. \tag{10.25}$$

$$\sum_{l=1}^{L} \eta_l \leq T_0 \tag{10.26}$$

6. Constraint on reliability: The reliability of each service has to be equal to or better than a certain specified level, i.e.,

$$Z_{lj}\cdot(q_j - Q_0) \geq 0 \qquad j = 1,2,...,m; l = 1,2,...,L. \tag{10.27}$$

7. Constraint on total reliability of service composition: The total reliability of the service composition should be equal to or greater than a certain level Q_1.

$$\sum_{l=1}^{L} Z_{lj} \cdot q_j \geq Q_1 \qquad j = 1,2,...,m. \tag{10.28}$$

8. Non negative and binary constraints:

$$Z_{lj} \geq 0 \tag{10.29}$$

$$Z_{lj} \in \{0,1\} \qquad \text{where } j = 1,2,...,m; l = 1,2,...,L. \tag{10.30}$$

$$\eta_l \geq 0 \tag{10.31}$$

The input–output constraints can handle both sequential and parallel flow patterns in service network.

We formulate three multi-criteria scenarios, wherein the customers require: (1) Optimal solution, (2) Optimal solution under a possible compromise of the QoS, and (3) An acceptable solution with both functional and nonfunctional attributes considered. Based on the formulations in previous sections, the following Multi-Criteria Programming (MCP) model with pure real constraints can be defined:

$$\min Z_1 = \sum_{l=1}^{L} \sum_{j=1}^{m} Z_{lj} \cdot p_j \tag{10.32}$$

$$\min Z_2 = \sum_{l=1}^{L} \eta_l \tag{10.33}$$

$$\min Z_3 = \sum_{l=1}^{L} \sum_{j=1}^{m} Z_{lj} \cdot q_j \tag{10.34}$$

subject to the constraints (10.5)–(10.15), e.g. subject to

$$\sum_{l=1}^{L} \sum_{j=1}^{m} Z_{lj} \cdot p_j \leq C_0 \tag{10.35}$$

$$\sum_{l=1}^{L} \eta_l \leq T_0 \tag{10.36}$$

$$\sum_{l=1}^{L} \sum_{j=1}^{m} Z_{lj} \cdot q_j \geq Q_1 \tag{10.37}$$

$$\sum_{l=1}^{L} \sum_{j=1}^{m} I_{ij} \cdot Z_{lj} \geq I_{i0} \quad i = 1,...,n \tag{10.38}$$

$$\sum_{l=1}^{L}\sum_{j=1}^{m} O_{ij}\cdot Z_{lj} \geq O_{i0} - I_{i0} \quad i = 1,\dots,n \tag{10.39}$$

$$\sum_{j=1}^{m} I_{ij}\cdot Z_{1j} \leq I_{i0} \quad i = 1,\dots,n \tag{10.40}$$

$$\sum_{j=1}^{m} I_{ij}\cdot Z_{k+1,j} - \sum_{l=1}^{k}\sum_{j=1}^{m} O_{ij}\cdot Z_{lj} \leq I_{i0} \quad i = 1,\dots,n; k = 1,\dots,L-1 \tag{10.41}$$

$$Z_{lj}\cdot(q_j - Q_0) \geq 0 \quad j = 1,\dots,m; l = 1,\dots,L \tag{10.42}$$

$$\eta_j - t_j\cdot Z_{lj} \geq 0 \quad j = 1,\dots,m; l = 1,\dots,L \tag{10.43}$$

$$Z_{lj} \geq 0 \quad j = 1,\dots,m; l = 1,\dots,L \tag{10.44}$$

$$Z_{lj} \in \{0,1\} \quad j = 1,\dots,m; l = 1,\dots,L \tag{10.45}$$

$$\eta_l \geq 0 \quad l = 1,\dots,L \tag{10.46}$$

This MCP model can be solved either by the preemptive method or by the non-preemptive method (weighted average method). If the customer of the query gives the priority of the objectives in order. For instance, if $\min Z_1$ has the highest priority (denote it P_1), $\min Z_2$ has the second highest priority (denote it P_2), and $\min Z_3$ has the least priority (denote it P_3), the model can be solved by solving three mathematical programming model sequentially (see [34]). This is called preemptive method. We call the preemptive model with pure real constraints as *Model 1*.

If the customer of the query gives the weights to the objectives. For instance, if the weights of Z_1, Z_2 and Z_3 are W_1, W_2 and W_3, respectively, the above model can be solved by solving the mathematical programming model with objective

$$\min W_1\cdot Z_1 + W_2\cdot Z_2 - W_3\cdot Z_3. \tag{10.47}$$

We call the non-preemptive model with pure real constraints as *Model 2*.

The advantage of MCP models (Model 1 and Model 2) is that they find the service composition of the query which satisfies both the functional requirements (starting from the inputs given in the query, it finds the composition to give the outputs requested in the query), and the nonfunctional requirements (also called QoS requirements: for example cost, reliability and execution time). The disadvantage of this model is that it won't give a compromise solution if there is not a composition satisfying both functional and nonfunctional requirements. In general, the compromise solution is informative and useful to the customer, so we revise some of the constraints into goal constraints in [33]. This distinguishes our work from the existing literature. Interested readers are referred to [33] for other optimization models in CS networks.

10.5.4.2 An Application in Manufacturing Process Integration

The standardization of XML (Extensible Markup Language), SOAP (Simple Object Access Protocol) and UDDI (Universal Description, Discovery, and Integration) enables information exchange across platforms in varieties of areas. Web service integration engine provides the manufacturing industry the ability to horizontally and vertically integrate data across a wide range-machines plants, vendors, and enterprise domains. Manufacturing Execution System (MES) and Enterprize Resource Planning (ERP) are able to exchange data of distributed processes through the Internet. This whole system comprises of a wide-area distributed systems which are typically connected to the Internet or the Intranet.

In the case study, we considered the manufacturer user has an information system, which has a client machine for a business domain. There are separate service providers to collect and manipulate information via the Internet. An application from the business domain is sent to a service broker, which is running on the Internet, and reads list of service providers from the service registry center. According to the information the manufacturer knows, and needs, a proper service can be found, if it exists. Thus, the manufacturer and the service provider can exchange data between them using SOAP communication protocol. If no single service can meet the manufacturer's needs alone, a series of web services need to be composed through a service composition algorithm.

Scenario: In the near future we can envision that the online services are all standard modules in WSDL (Web Services Description Language), and they can communicate with each other via SOAP (Service Oriented Architecture Protocol). In this scenario, an automobile manufacturer is designing a new car for the future and decides to use online web services as one of the options.

We illustrate our approach through a simple example with a service set of 5 services:

Z, the set of web services, including 5 online web services, i.e., $m = 5$.

$$\begin{array}{l}
Service_1: \text{“sketch designing”} \\
\begin{cases} inputs: & \text{style, functions, price} \\ outputs: & \text{2D model, tolerance, material info} \end{cases} \\
Service_2: \text{“three dimension modeling”} \\
\begin{cases} inputs: & \text{2D model, tolerance, material info} \\ outputs: & \text{shapes, vertices, structure} \end{cases} \\
Service_3: \text{“surface and volume meshing”} \\
\begin{cases} inputs: & \text{shapes, vertices, structure} \\ outputs: & \text{nodes, elements, assigned material} \end{cases} \\
Service_4: \text{“simulation and analysis”} \\
\begin{cases} inputs: & \text{nodes, elements, assigned material} \\ outputs: & \text{safety, mileage, speed} \end{cases} \\
Service_5: \text{“simulation and analysis”} \\
\begin{cases} inputs: & \text{safety, mileage, speed} \\ outputs: & \text{price, structure, speed} \end{cases}
\end{array} \tag{10.48}$$

The total set of the attributes are: [style, functions, price, 2D model, tolerance, material info, shapes, structure, vertex, nodes, elements, assigned material, safety, mileage, speed].

Using the MCP model discussed, we can build an optimization model. Given that the maximum depth of the levels is 5, the preemptive goal programming model finds a solution that carries out service composition in 4 levels, and the solution is $Z_{11} = Z_{22} = Z_{33} = Z_{44} = 1$, and the others are zero.

At the first level, service Z_1 is selected to execute using the input of initial request I_0; at the second level, service Z_2 is selected execute using the output from Z_1; at the third level, service Z_3 is selected to execute using the output from Z_1, Z_2; at the fourth level, service Z_4 is selected to execute using the outputs from Z_1, Z_2, Z_3.

10.6 Applications of Service Composition

The services currently available online are all standard modules in WSDL (Web Services Description Language), and they can exchange information interactively via SOAP (Service Oriented Architecture Protocol). Utilizing the online services is usually cheaper than hiring a number of physical consultants. Web Service Composition, among other domains, can be applied to (1) health care, (2) production design, and (3) enterprise application integration, etc. Figure 10.15 shows the infrastructure of the service network including users, registry, service providers and service brokers.

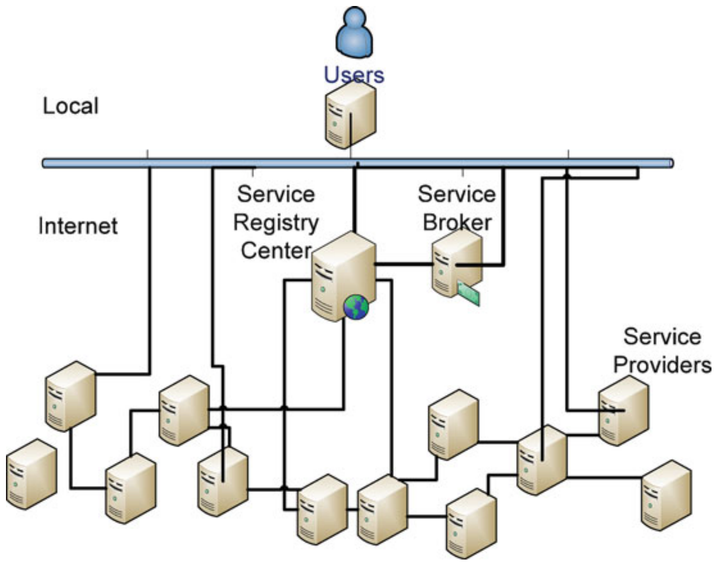


Fig. 10.15 The topology of users, service providers, service brokers, and service registry center

10.6.1 Resource Allocation in Health Care

In recent times, we have been experiencing extreme deficiencies of medical resources, such as medical personnel, medical devices, blood or organ donors for organ transplantation. Such a deficiency can be reduced by collaboration among medical institutions and national health organizations, which might be involved in the issues related to home land security, disaster recovery, and emergency management and thus, the efficiency is important. Efficient collaborations in such a geographically distributed environment need a support from Information Technology (IT) through the Internet. Web services can provide such collaborations for health care services.

10.6.2 Product Design in Manufacturing

One of the recent trends in product design is modularization of parts. Modularization also contributes to defining parts in a standardized, machine-readable way. Especially, modularization enables us to define the features of each part as input, output, function and geometric information. Defining a set of inputs, outputs, and other features makes it possible for a manufacturing company to identify the required parts. In order to automate such production design processes, manufacturers can utilize the Web Service Composition algorithms to match the parts from suppliers. Moon et al. used a service based platform to facilitate global product design [35].

10.6.3 Business Integration

Globalization is a significant feature in the manufacturing industry. Services are offered by a variety of companies depending on their expertise, such as production design companies, manufacturing companies, marketing companies, etc. These enterprises can collaborate with each other through web services integration and automation. These online business processes can be executed automatically in a scheduled order. One service could use another's output as an input, and all of these services can work together and compose the production solutions for a business. For instance, automobile manufacturers are able to utilize online services to help designing their new car models for the coming years (see Sect. 10.5.4.2 for a brief discussion). As the number of Web Services increases and varieties of emerging service requests appear in current, competitive, and complex business environments, automatic web service composition algorithms become an essential feature of commercial web services.

In general web services as we have described can be modeled as networks (CS). From the networks view it is possible that the functionalities of the nodes

Table 10.7 Applications of networks

Engineered system	Scale	Network model
Supply chain	Large, often global	Not known
Service	Large, often global	Scale free
Mobile	Large, global	Not known
Crowd sourcing	Large, global	Not known
Internet	Large scale, global	Scale free

in a network can be modeled as services and service composition algorithms can be used on them. Man-made networks are the consequences of proliferation of information technology and globalization. Due to IT connectivity and reach, the world is becoming a networked one. How can we use the knowledge that is gained so far in network science in these new application areas of engineered systems? The last decade has seen many firsts in these areas with respect to applying network science principles. In reality, there are many man-made or engineered networks which are worth of the readers' interest in pursuing these lines of research. Several man-made networks are shown in Table 10.7.

10.7 Conclusions and Future Work

In this chapter, Internet service networks and a set of current planning techniques of Internet services are discussed. A novel service network mining technique is explored in Sect. 10.5. This technique can be used to find clusters in service networks and identify popular and critical services and hot concepts online. Network mining can be used to discover hidden information before service composition to reduce the overall composition. The real-time re-planning in relation to faulty behaviors of web service composition needs to be fully studied in future work. A suitable control and recovery mechanism should be included in any service composition engine. Techniques to avoid information congestion and to guarantee the security of service delivery will also be in high demand in the future.

References

1. W3C, *Web Services Description Language (WSDL) Version 2.0*, W3C Working Draft, http://www.w3.org/TR/wsdl20, 2003.
2. IBM, http://www-128.ibm.com/developerworks/webservices.
3. UDDI, http://www.uddi.org/pubs/DataStructure-V1.00-Published-20020628.pdf.
4. Y. Li, J. P. Huai, H. Sun et al., *Pass: An approach to personalized automated service composition*, pp. 283-290, 2008.
5. B. Raman, S. Agarwal, Y. Chen et al., *The SAHARA model for service composition across multiple providers*, Pervasive Computing, pp. 585-597, 2002.

6. L. Zeng, B. Benatallah, A. H. H. Ngu et al., *QoS-aware middleware for web services composition*, IEEE Transactions on Software Engineering, pp. 311-327, 2004.
7. K. S. M. Chan, J. Bishop, and L. Baresi, *Survey and comparison of planning techniques for web services composition*, University of Pretoria2007. c ISMED, vol. 209, 2007.
8. T. Vossen, M. Ball, A. Lotem et al., *On the use of integer programming models in AI planning*, in Proc. IJCAI'1999, pp.304-309, 1999.
9. T. Vossen, M. Ball, A. Lotem et al., *Applying integer programming to AI planning*, The Knowledge Engineering Review, vol. 15, no. 1, pp. 85-100, 2000.
10. H. Kautz, and J. P. Walser, *State-space planning by integer optimization*, AAAI '99/IAAI '99 Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence American Association for Artificial Intelligence Menlo Park, CA, USA, pp. 526-533, 1999.
11. A. Gao, D. Yang, S. Tang et al., *Web service composition using integer programming-based models*, IEEE International Conference on e-Business Engineering, 2005. pp. 603-609, 2005.
12. J. Rao, and P. Kngas, *Application of linear logic to web service composition*,In Proceedings of ICWS'2003. pp.1 3, 2003
13. E. Sirin, J. Hendler, and B. Parsia, *Semi-automatic composition of web services using semantic descriptions*, In Proceedings of WSMAI'2003, pp. 17-24, 2003.
14. L. Xiao, and L. Zhang, *Automatic mapping from XML documents to ontologies*, in Proc. CIT'2004, pp. 321-325, 2004.
15. S. Huang, H. Chen, and L. J. Zhang, *Progressive auction based resource allocation in service-oriented architecture*,in Proc. IEEE SCC'2005, pp.85-92, 2005.
16. S. Y. Hwang, E. P. Lim, C. H. Lee et al., *Dynamic web service selection for reliable web service composition*, IEEE transactions on services computing, pp. 104-116, 2008.
17. G. Pacifici, M. Spreitzer, A. N. Tantawi et al., *Performance management for cluster-based web services*, IEEE Journal on Selected Areas in Communications, vol. 23, no. 12, pp. 2333-2343, 2005.
18. S. C. Oh, H. Kil, D. Lee et al., *Wsben: A web services discovery and composition benchmark*, in Proc. ICWS'2006, pp. 239-248, 2006.
19. S. C. Oh, D. Lee, and S. R. T. Kumara, *WSPR: An Effective and Scalable Web Service Composition Algorithm*, International Journal of Web Services Research, Vol. 4, No. 1, pp. 1-22, 2007.
20. L. J. Zhang, S. Cheng, Y. M. Chee et al., *Pattern recognition based adaptive categorization technique and solution for services selection*, apscc, pp. 535-543, 2007.
21. S. C. Oh, D. Lee, and S. R. T. Kumara, *Effective web service composition in diverse and large-scale service networks*, IEEE transactions on services computing, pp. 15-32, 2008.
22. F. Montagut, and R. Molva, *Bridging security and fault management within distributed workflow management systems*, IEEE transactions on services computing, pp. 33-48, 2008.
23. H. Lufei, W. Shi, and V. Chaudhary, *Adaptive Secure Access to Remote Services in Mobile Environments*, IEEE transactions on services computing, pp. 49-61, 2008.
24. K. A. Phan, Z. Tari, and P. Bertok, *Similarity-Based SOAP Multicast Protocol to Reduce Bandwith and Latency in Web Services*, IEEE transactions on services computing, pp. 88-103, 2008.
25. J. Wei, L. Singaravelu, and C. Pu, *A secure information flow architecture for web service platforms*, IEEE transactions on services computing, pp. 75-87, 2008.
26. R. Albert, H. Jeong, and A. L. Barabsi, *The diameter of the world wide web*, Arxiv preprint cond-mat/9907038, 1999.
27. R. V. Engelen, *Are web services scale free?*, http://www.cs.fsu.edu/~engelen/powerlaw.html, 2005.
28. L. Y. Cui, S. Kumara, J. J. W. Yoo et al., *Large-Scale Network Decomposition and Mathematical Programming Based Web Service Composition*, IEEE Conference on Commerce and Enterprise Computing'2009, pp. 511-514, 2009.

29. L. Li, and I. Horrocks, *A software framework for matchmaking based on semantic web technology*, International Journal of Electronic Commerce, vol. 8, no. 4, pp. 39-60, 2004.
30. L. Y. Cui, S. Kumara, and R. Albert, *Complex networks: an engineering view*, Circuits and Systems Magazine, IEEE, vol. 10, no. 3, pp. 10-25, 2010.
31. R. Baggio, N. Scott, and C. cooper, *Network science: A review focused on tourism*, http://cdsweb.cern.ch/record/1245639, 2010.
32. H. P. Thadakamalla, U. N. Raghavan, and S. Kumara, *Survivability of multiagent-based supply networks: A topological perspective*, IEEE Intelligent Systems, pp. 24-31, 2004.
33. L. Y. Cui, S. R. T. Kumara, and D. Lee, *Scenario Analysis of Web Service Composition based on Multi-criteria Mathematical Goal Programming*, Service Science, vol. 3, no. 3, 2011.
34. J. L. Arthur, and A. Ravindran, *PAGP, a partitioning algorithm for (linear) goal programming problems*, ACM Transactions on Mathematical Software (TOMS), vol. 6, no. 3, pp. 378-386, 1980.
35. S. K. Moon, T. W. Simposon, L. Y. Cui et al., *A Service based Platform Design Method for Customized Products*, Proceedings of CIRP Integrated Production and Service Systems conference'2010, pp. 3-10, 2010.
36. D. L. McGuinness, F. Van Harmelen et al., *OWL web ontology language overview*, W3C recommendation, http://www.w3.org/TR/owl-features/,2004.
37. A. L. Barabsi and R. Albert, *Emergence of Scaling in Random Networks*, Science, vol.286 no. 5439, pp. 509-512, 1999.
38. P. Erdöys and A. Rényi, *Random Graphs*, Publ. Math. Inst. Hung. Acad.Sci. vol. 5, No.17, 1960.
39. D. J. Watts and S. H. Strogatz1, *Collective Dynamics of "Small-world" Networks*, Nature, 393, pp. 440-442, 1998
40. R. Xu and D. Wunsch, *Survey of Clustering Algorithms* Neural Networks, IEEE Transactions on, vol. 16, no.3, pp. 645-678, 2005.
41. L. Y. Cui, S. Kumara and T. Yao, *Service Composition using Dynamic Programming and Concept Service (CS) Network*, Proceedings of IERC 2011, May Reno, Nevada, USA., 2011.
42. A. Gao, D. Yang et al.,*Web Service Composition Using Markov Decision Processes*. Advances in Web-Age Information Management, Springer Berlin, vol.3739/2005, pp. 308-319, 2005.

Part III
Online Social Networks and Security

Chapter 11
On Detection of Community Structure in Dynamic Social Networks

Nam P. Nguyen, Ying Xuan, and My T. Thai

Abstract Community structure is a very special and interesting property of social networks. Knowledge of network community structure not only provides us key insights into developing more social-aware strategies for social network problems, but also promises a wide range of applications enabled by mobile networking, such as routings in Mobile Ad Hoc Networks (MANETs) and worm containments in cellular networks. Unfortunately, understanding this structure is very challenging, especially in dynamic social networks where social activities and interactions tend to come and go rapidly. Can we quickly and efficiently identify the network community structure? Can we adaptively update this structure based on its history instead of recomputing from scratch?

In this chapter, we present two methods for detecting community structures on social networks. First, we introduce *Q*uick *C*ommunity *A*daptation (*QCA*), an adaptive modularity-based method for identifying and tracing the discrete community structure of dynamic social networks. This approach has not only the power of quickly and efficiently updating the network structure by only using the identified structures, but also the ability of tracing the evolution of its communities over time. Next, we present *DOCA*, an quick method for revealing the overlapping network communities that can be implemented in a decentralized manner. To illustrate the effectiveness of our methods, we extensively test *QCA* and *DOCA* on not only synthesized but also on real-world dynamic social networks including ENRON email network, arXiv e-print citation network and Facebook network. Finally, we demonstrate the bright applicability of our methods via two realistic applications on routing strategies in MANETs and worm containment on online social networks.

N.P. Nguyen (✉) • Y. Xuan • M.T. Thai
CISE Department, University of Florida, Gainesville, Florida, USA
e-mail: nanguyen@cise.ufl.edu; yxuan@cise.ufl.edu; mythai@cise.ufl.edu

11.1 Introduction

Many social networks exhibit the property of containing community structure [1,2], i.e. they naturally divide into groups of vertices with denser connections inside each group and fewer connections crossing groups, where vertices and connections represent network users and their social interactions, respectively. Members in each community of a social network usually share common interests such as photography, movies, music or discussion topics and thus, they tend to interact more frequently with each other than with members outside of their community. Community detection in a network is the gathering of network vertices into groups in such a way that nodes in each group are densely connected inside and sparser outside.

It is noteworthy to differentiate between community detection and graph clustering. These two problems share the same objective of partitioning network nodes into groups; however, the number of clusters is predefined or given as part of the input in graph clustering whereas the number of communities is typically unknown in community detection. Detecting communities in a network provides us meaningful insights its internal structure as well as its organization principles. Furthermore, knowing the structure of network communities could also provide us more helpful points of view to some uncovered parts of the network, thus helps in preventing potential networking diseases such as virus or worm propagation. Studies on community detection on static networks can be found in an excellent survey [3] as well as in the work of [4–7] and references therein.

Real-world social networks, however, are not always static. In fact, most of social networks in reality (such as Facebook, Bebo, and Twitter) evolve and witness an expand in size and space as their users increase, thus lend themselves to the field of dynamic networks. A dynamic network is a special type of evolving complex networks in which changes are frequently introduced over time. In the sense of an online social network, such as Facebook, Twitter, or Flickr, changes are usually introduced by users joining in or withdrawing from one or more groups or communities, by friends and friends connecting together, or by new people making friend with each other. Any of these events seems to have a little effect to a local structure of the network on one hand; the dynamics of the network over a long period of time, on the other hand, may lead to a significant transformation of the network community structure, thus drives a natural need of reidentification. However, the rapidly and unpredictably changing topology of a dynamic social network makes it an extremely complicated yet challenging problem.

Although one can possibly run any of the static community detection methods, which are widely available [4–6, 8], to find the new community structure whenever the network is updated, he may encounter some disadvantages that cannot be neglected (1) the long running time of a specific static method on large networks (2) the trap of local optima and (3) the almost same reaction to a small change to some local part of the network. A better, much efficient, and less time consuming way to accomplish this expensive task is to adaptively update the network communities

$$\begin{tikzcd}
\mathcal{G}: & G_t \arrow[r, "\Delta G_t"] \arrow[d] & G_{t+1} \arrow[d] \\
\mathcal{C}: & C(G_t) \arrow[r, dashed, "\mathcal{A}"] & C(G_{t+1})
\end{tikzcd}$$

Fig. 11.1 The network evolves from time t to $t+1$ under the change ΔG_t. The adaptive algorithm $\mathscr{A}$ quickly finds the updated structure $\mathscr{C}(G_{t+1})$ based on the previous structure $\mathscr{C}(G_t)$ together with the changes ΔG_t

from the previous known structures, which helps to avoid the hassle of recomputing from scratch. This adaptive approach is the main focus of our study in this chapter. In Fig. 11.1, we briefly generalize the idea of dynamic network community structure adaptation.

Detecting community structure in a dynamic social network is of considerable usages. To give a sense of its effects, consider the routing problem in communication network where nodes and links present people and mobile communications, respectively. Due to nodes mobility and unstable links properties of the network, designing an efficient routing scheme is extremely challenging. However, since people have a natural tendency to form groups of communication, there exist groups of nodes which are densely connected inside than outside in the underlying MANET as a reflection, and therefore, forms community structure in that MANET. An effective routing algorithm, as soon as it discovers the network community structure, can directly route or forward messages to nodes in the same (or to the related) community as the destination. By doing this way, we can avoid unnecessary messages forwarding through nodes in different communities, thus can lower the number of duplicate messages as well as reduce the overhead information, which are essential in MANETs.

In this context, we study the dynamics of communities in social network and prove theoretical results regarding its various behaviors over time. We then propose *QCA*, a fast and adaptive algorithm for efficiently identifying the community structure of a dynamic social network. Our approach takes into account the previously discovered network structure and processes on network changes only, thus significantly reduces computational cost and processing time. In addition, we present *DOCA*, a quick detection method that can nicely identify overlapping network communities with high quality. We extensively evaluate our algorithms on various real world dynamic social networks including Enron email network, ArXiv citation network, and Facebook network. Experimental results show that our method not only achieves competitive modularities but also high quality community structures in a timely manner. As applications, we employ *QCA* as a community identification core in routing strategies in MANETs and worm containment methods in online social networks. Simulation results show that *QCA* outperforms the current available methods and confirm the bright applicability of our proposed method in not only in social networks but also in mobile computing.

The rest of this chapter is organized as follow: Sect. 11.2 presents the preliminaries and problem definition. Section 11.3 gives a complete description of our algorithms and their theoretical analysis. Section 11.4 shows experimental results of our approach on various real world datasets. In Sects. 11.5 and 11.6, we present two practical application of our approaches in MANETs and on OSNs, respectively. In Sect. 11.7, we present *DOCA*, our method for identifying overlapping network communities. In Sect. 11.8, we discuss about the related work and finally conclude our work in Sect. 11.9.

11.2 Problem Formulation

In this section, we present the notations, objective function as well as the dynamic graph model representing a social network that we will use throughout the paper.

(Notation) Let $G = (V,E)$ be an undirected unweighted graph with N nodes and M links representing a social network. Let $\mathscr{C} = \{C_1, C_2, .., C_k\}$ denote a collection of disjoint communities, where $C_i \in \mathscr{C}$ is a community of G. For each vertex u, denote by d_u, $C(u)$ and $NC(u)$ its degree, the community containing u and the set of its adjacent communities. Furthermore, for any $S \subseteq V$, let m_S, d_S and e^u_S be the number of links inside S, the total degree of vertices in S and the number of connections from u to S, respectively. The pairs of terms *community* and *module*; *node* and *vertex* as well as *edge* and *link* and are used interchangeably.

(Dynamic social network) Let $G^s = (V^s, E^s)$ be a time dependent network snapshot recorded at time s. Denote by ΔV^s and ΔE^s the sets of vertices and links to be introduced (or removed) at time s and let $\Delta G^s = (\Delta V^s, \Delta E^s)$ denote the change in term of the whole network. The next network snapshot G^{s+1} is the current one together with changes, i.e. $G^{s+1} = G^s \cup \Delta G^s$. A *dynamic network* $\mathscr{G}$ is a sequence of network snapshots evolving over time: $\mathscr{G} = (G^0, G^1, .., G^s, ..)$.

(Objective function) In order to quantify the goodness of a network community structure, we take into account the most widely accepted measure called *modularity* $\mathscr{Q}$, which is defined in [6] as

$$\mathscr{Q} = \sum_{C \in \mathscr{C}} \frac{m_C}{M} - \frac{\mathrm{d}_C^2}{4M^2}.$$

Basically, $\mathscr{Q}$ is the fraction of all links within communities subtracts the expected value of the same quantity in a graph whose nodes have the same degrees but links are distributed randomly, and the higher modularity $\mathscr{Q}$, the better network community structure. Therefore, our objective is to find a community assignment for each vertex in the network such that $\mathscr{Q}$ is maximized. Modularity, just like other quality measurements for community identifications, has some certain disadvantages such as its non-locality and scaling behavior [7] or resolution limit [9]. However, it is still very well considered due to its robustness and usefulness that closely agree with intuition on a wide range of real world networks.

Problem Definition: Given a dynamic social network $\mathscr{G} = (G^0, G^1, .., G^s, ..)$ where G^0 is the original network and G^1, G^2,.., G^s,.. are the network snapshots obtained through ΔG^1, ΔG^2,.., ΔG^s,.. we need to devise an adaptive algorithm to efficiently detect and identify the network community structure at any time point utilizing the information from the previous snapshots as well as tracing the evolution of the network community structure.

11.3 Method Description

Let us first discuss about how changes to the evolving network topology affect the structure of its communities. Assume that $G = (V, E)$ is the current network and $\mathscr{C} = \{C_1, C_2, .., C_k\}$ is its corresponding community structure. We use the term *intra-community links* to denote edges whose two endpoints belong to the same community and the term *inter-community links* to denote those with endpoints connecting different communities. For each community C of G, the number of connections linking C with other communities are much fewer than the number of connections within C itself, i.e. nodes in C are densely connected inside than outside. Intuitively, adding intra-community links inside or removing inter-community links between communities of G will strengthen those communities and make the structure of G more clear. Vice versa, removing intra-community links and inserting inter-community links will loosen the structure of G. However, when two communities have less distraction caused by each other, adding or removing links makes them more attractive to each other and thus, leaves a possibility that they will be combined to form a new community. The community updating process, as a result, is extremely challenging since any insignificant change in the network topology can possibly lead to an unexpected transformation of its community structure. We will discuss in detail various possible behaviors of a dynamic network community structure in Sect. 11.3.1.

In order to reflect changes introduced to a social network, its underlying graph is constantly updated by either inserting or removing a node or a set of nodes, or by either introducing or deleting an edge or a set of edges. In fact, the introduction or removal a set of nodes (or edges) can be decomposed as a sequence of node (or edge) insertions (or removals), in which a single node (or a single edge) is introduced (or removed) at a time. This observation helps us to treat network changes as a collection of *simple events* where a simple event can be one of *newNode, removeNode, newEdge, removeEdge* whose details are as follow:

- *NewNode* $(V + u)$: A new node u with its associated edges are introduced. u could come with no or more than one new edge(s).
- *RemoveNode* $(V - u)$: A node u and its adjacent edges are removed from the network.
- *NewEdge* $(E + e)$: A new edge e connecting two existing nodes is introduced.
- *RemoveEdge* $(E - e)$: An existing edge e in the network is removed.

11.3.1 Algorithms

Our approach first requires an initial community structure $\mathscr{C}_0$, which we refer to as a basic structure, in order to process further. Since the input model is restricted as an undirected unweighted network, this initial community structure can be obtained by performing any of the available static community detection methods [4, 5, 8]. To obtain a good basic structure, we choose the method proposed by Blondel *et al* in [5] which produces a network community structure of high modularity in a reasonable amount of time [3].

11.3.1.1 New Node

Let us consider the first case when a new node u and its associated connections are introduced. Note that u may come with no adjacent edges or with many of them connecting one or more communities. If u has no adjacent edges, we create a new community containing only u and leave the other communities as well as the overall modularity $\mathscr{Q}$ intact. The interesting case happens, as it always does, when u comes with edges connecting one ore more existing communities. In this latter situation, we need to determine which community u should join in in order to maximize the gained modularity. There are several local methods introduced for this task, for instance the algorithms of [4, 8]. Our method is inspired by a physical approach proposed in [10], in which each node is influenced by two forces: F_{in}^{C} (to keep u stays inside community C) and F_{out}^{C} (the force a community C makes in order to bring u to C) defined as follow:

$$F_{\text{in}}^{C}(u) = e_{C}^{u} - \frac{\mathrm{d}_u(\mathrm{d}_C - \mathrm{d}_u)}{2M},$$

$$F_{\text{out}}^{S}(u) = \max_{S \in NC(u)} \left\{ e_{S}^{u} - \frac{\mathrm{d}_u \mathrm{d}_{\text{outS}}}{2M} \right\},$$

where d_{outS} is of opposite meaning of d_S. Taking into account the above two forces, a node u can actively determines its best community membership by computing those forces and either lets itself join in the community having the highest $F_{\text{out}}(u)$ (if $F_{\text{out}}(u) > F_{\text{in}}^{C(u)}(u)$) or stays put in the current community otherwise. By Theorem 11.1, we bridge the connection between those forces and the objective function, i.e. joining the new node in the community with the highest outer force will maximize the local gained modularity. This is the central idea for handling the first case and the algorithm is presented in Algorithm 1.

Theorem 11.1. *Suppose C is the community that gives maximum $F_{\text{out}}^{C}(u)$ when a new node u with degree p is introduced to G, then joining u in C gives the maximal modularity contribution.*

Algorithm 1 New_Node

```
Input: New node $u$ with associated links; Current structure $\mathscr{C}_t$.
Output: An updated structure $\mathscr{C}_{t+1}$
1: Create a new community of only $u$;
2: for $v \in N(u)$ do
3:     Let $v$ determine its best community;
4: end for
5: for $C \in NC(u)$ do
6:     Find $F^C_{\mathrm{out}}(u)$;
7: end for
8: Let $C_u \leftarrow \arg\max_C \{F^C_{\mathrm{out}}(u)\}$;
9: Update $\mathscr{C}_{t+1} : \mathscr{C}_{t+1} \leftarrow \big(\mathscr{C}_t \backslash C_u\big) \cup \big(C_u \cup u\big)$;
```

Proof. Let D be a community of G and $D \neq C$, we show that joining u in D contributes less modularity than joining u in C. The overall modularity $\mathscr{Q}$ when u joins in C is

$$\mathscr{Q} = \frac{m_C + e^u_C}{M+p} - \frac{(\mathrm{d}_C + e^u_C + p)^2}{4(M+p)^2} + \frac{m_D}{M+p} - \frac{(\mathrm{d}_D + e^u_D)^2}{4(M+p)^2}, +A$$

where A is the summation of other modularity contributions. Similarly, joining u to D gives

$$\mathscr{Q}' = \frac{m_C}{M+p} - \frac{(\mathrm{d}_C + e^u_C)^2}{4(M+p)^2} + \frac{m_D + e^u_D}{M+p} - \frac{(\mathrm{d}_D + e^u_D + p)^2}{4(M+p)^2} + A$$

and

$$\mathscr{Q} - \mathscr{Q}' = \frac{1}{M+p}\left(e^u_C - e^u_D + \frac{p(\mathrm{d}_D - \mathrm{d}_C + e^u_D - e^u_C)}{2(M+p)}\right).$$

Now, since C is the community that gives the maximum $F^C_{\mathrm{out}}(u)$, we obtain

$$e^u_C - \frac{p(\mathrm{d}_C + e^u_C)}{2(M+p)} > e^u_D - \frac{p(\mathrm{d}_D + e^u_D)}{2(M+p)},$$

which implies

$$e^u_C - e^u_D + \frac{p(\mathrm{d}_D - \mathrm{d}_C + e^u_D - e^u_C)}{2(M+p)} > 0.$$

Hence, $\mathscr{Q} - \mathscr{Q}' > 0$ and thus the conclusion follows.

11.3.1.2 New Edge

In case that a new edge $e = (u,v)$ connecting two existing vertices u,v is introduced, we divide it further into two smaller cases: e is an intra-community link (totally inside a community C) or an inter-community link (connects two communities

Algorithm 2 New_Edge

```
Input: Edge $\{u,v\}$ to be added; Current structure $\mathscr{C}_t$.
Output: An updated structure $\mathscr{C}_{t+1}$.
 1: if ($u$ and $v$ are new vertices) then
 2:     $\mathscr{C}_{t+1} \leftarrow \mathscr{C}_t \cup \{u,v\}$;
 3: else if $C(u) \neq C(v)$ then
 4:     if $\Delta q_{u,C(u),C(v)} < 0$ and $\Delta q_{v,C(u),C(v)} < 0$ then
 5:         return $\mathscr{C}_{t+1} \equiv \mathscr{C}_t$;
 6:     else
 7:         $w = \arg\max\{\Delta q_{u,C(u),C(v)}, \Delta q_{v,C(u),C(v)}\}$;
 8:         Move $w$ to the new community;
 9:         for $t \in N(w)$ do
10:             Let $t$ determine its best community;
11:         end for
12:         Update $C_{t+1}$;
13:     end if
14: end if
```

$C(u)$ and $C(v)$). If e is inside a community C, its presence will strengthen the inner structure of C according to Lemma 11.1. Furthermore, by Theorem 11.2, we know that adding e should not split the current community C into smaller modules. Therefore, we leave the current network structure intact in this case.

The interesting situation happens when e is a link connecting communities $C(u)$ and $C(v)$ since the presence of e could possibly make u (or v) leave its current modules and join in the new community. Additionally, if u (or v) decides to change its membership status, it can eventually advertise its new community to all its neighbors and some of them might want to change their memberships as a consequence. By Lemma 11.2, we show that if u (or v) should ever change its cluster assignment then $C(v)$ (or $C(u)$) is the best new community for it. But how can we efficiently and quickly decide whether u (or v) should change its membership or not in order to form a better community structure when e is added? To this end, we provide a criterion to test for membership changing of u and v in Theorem 11.3. Here, if both $\Delta q_{u,C,D}$ and $\Delta q_{v,C,D}$ fail to satisfy the criteria, we can safely preserve the current network community structure and keep going (Corollary 11.1). Otherwise, we move u (or v) to its new community and consequently let its neighbors determine their best modules to join in, using local search and swapping to maximize gained modularity. Figure 11.2a describes the procedure for this latter case. The algorithm is described in Algorithm 2.

Lemma 11.1. *For any community $C \in \mathscr{C}$, if $\mathrm{d}_C \leq M-1$ then adding an edge within C will increase its modularity contribution.*

Proof. The portion $\mathscr{Q}_C$ that community C contributes to the overall modularity $\mathscr{Q}$ is: $\mathscr{Q}_C = \frac{m_C}{M} - \frac{\mathrm{d}_C^2}{4M^2}$. When a new edge coming in, the new modularity $\mathscr{Q}'_C$ is $\mathscr{Q}'_C = \frac{m_C+1}{M+1} - \frac{(\mathrm{d}_C+2)^2}{4(M+1)^2}$. Now, taking the difference between the two expressions $\mathscr{Q}'_C$ and $\mathscr{Q}_C$ gives

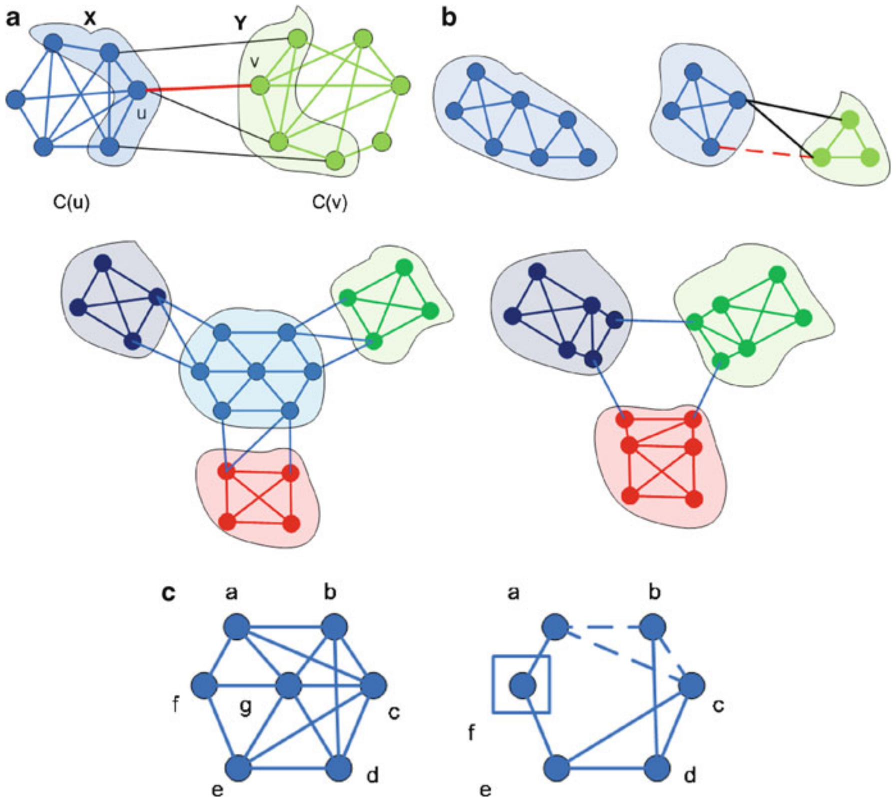


Fig. 11.2 (**a**): When an edge (u,v) joining $C(u)$ and $C(v)$ is introduced. Tests on membership are performed on sets X and Y. (**b**): (*Left*) The original community (*Right*) After an edge (in dotted line) is removed, the community is broken into two smaller communities. (**c**): (*Left*) The original network with four communities (*Right*) After the highest degree node is removed, the leftover nodes join in different modules, forming a new network with three communities. (**d**): (*Left*) The original community (*Right*) When the central node g is removed, a 3-clique is placed at a to discover b, c, d and e. f assigned singleton afterwards

$$
\begin{aligned}
\Delta \mathcal{Q}_C &= \mathcal{Q}'_C - \mathcal{Q}_C \\
&= \frac{4M^3 - 4m_C M^2 - 4\mathrm{d}_C M^2 - 4m_C M + 2\mathrm{d}_C^2 M + \mathrm{d}_C^2}{4(M+1)^2 M^2} \\
&\geq \frac{4M^3 - 6\mathrm{d}_C M^2 - 2\mathrm{d}_C M + 2\mathrm{d}_C^2 M + \mathrm{d}_C^2}{4(M+1)^2 M^2} \qquad (\text{since } m_C \leq \tfrac{\mathrm{d}_C}{2}) \\
&\geq \frac{(2M^2 - 2\mathrm{d}_C M - \mathrm{d}_C)(2M - \mathrm{d}_C)}{4(M+1)^2 M^2} \geq 0.
\end{aligned}
$$

The last inequality holds since $\mathrm{d}_C \leq M-1$ implies $2M^2 - 2\mathrm{d}_C M - \mathrm{d}_C \geq 0$.

Theorem 11.2. *If C is a community in the current snapshot of G, then adding any intra-community link to C will not split it into smaller modules.*

Proof. Assume the contradiction, i.e, C should be divided into smaller modules when an edge is added into it. Let $X_1, X_2, .., X_k$ be disjoint subsets of C representing these modules. Denoted by d_i and e_{ij} the total degree of vertices inside X_i and the number of links going from X_i to X_j, respectedly. Assume that, W.L.O.G, when an edge is added inside C, it is added to X_1. We will show that

$$\left\lceil \frac{\sum_{i\neq j} \mathrm{d}_i \mathrm{d}_j}{2M} \right\rceil < \sum_{i\neq j} e_{ij} < \left\lceil \frac{\sum_{i\neq j} \mathrm{d}_i \mathrm{d}_j}{2M} \right\rceil + 1,$$

which can not happen since $\sum_{i\neq j} e_{ij}$ is an integer.

Recall that $\mathscr{Q}_C = \frac{m_C}{M} - \frac{\mathrm{d}_C^2}{4M^2}$ and $\mathscr{Q}_{X_i} = \frac{m_i}{M} - \frac{\mathrm{d}_i^2}{4M^2}$. Prior to adding an edge to C, we have $\mathscr{Q}_C > \sum_{i=1}^k \mathscr{Q}_{X_i}$, or equivalently,

$$\frac{m_C}{M} - \frac{\mathrm{d}_C^2}{4M^2} > \sum_{i=1}^k \left(\frac{m_i}{M} - \frac{\mathrm{d}_i^2}{4M^2} \right).$$

Since $X_1, X_2, .., X_k$ are disjoint subsets of C, it follows that $\mathrm{d}_C = \sum_{i=1}^k \mathrm{d}_i$ and $m_C = \sum_{i=1}^k m_i + \sum_{i\neq j} e_{ij}$ (where m_i is the number of links inside X_i). Thus, the above inequality equals to

$$\frac{m_C}{M} - \sum_{i=1}^k \frac{m_i}{M} > \frac{\mathrm{d}_C^2}{4M^2} - \sum_{i=1}^k \frac{\mathrm{d}_i^2}{4M^2}$$

or

$$\sum_{i<j} e_{ij} > \left\lceil \frac{\sum_{i\neq j} \mathrm{d}_i \mathrm{d}_j}{2M} \right\rceil.$$

Now, assume that the new edge is added to X_1 and C is split into $X_1, X_2, .., X_k$ which implies that dividing C into k smaller communities will increase the overall modularity, i.e, $\mathscr{Q}'_C < \sum_{i=1}^k \mathscr{Q}'_{X_i}$

$$\Leftrightarrow \quad \frac{\sum_{i=1}^k m_i + \sum_{i<j} e_{ij} + 1}{M+1} - \frac{\left(\sum_{i=1}^k \mathrm{d}_i + 2\right)^2}{4(M+1^2)}$$
$$< \frac{m_1 + 1}{M+1} - \frac{(\mathrm{d}_1 + 2)^2}{4(M+1)^2} + \sum_{i=2}^k \left(\frac{m_i}{M+1} - \frac{\mathrm{d}_i^2}{4(M+1)^2} \right)$$
$$\Leftrightarrow \quad \frac{\sum_{i=1}^k m_i + \sum_{i<j} e_{ij} + 1}{M+1} - \frac{\left(\sum_{i=1}^k \mathrm{d}_i + 2\right)^2}{4(M+1^2)}$$
$$< \frac{\sum_{i=1}^k m_i + 1}{M+1} - \frac{(\mathrm{d}_1 + 2)^2}{4(M+1)^2} - \sum_{i=2}^k \frac{\mathrm{d}_i^2}{4(M+1)^2}$$
$$\Leftrightarrow \quad \sum_{i<j} e_{ij} < \frac{\sum_{i=1}^k \mathrm{d}_i - 2\mathrm{d}_1 + \sum_{i<j} \mathrm{d}_i \mathrm{d}_j}{2(M+1)}.$$

Moreover, since it is obvious that $\sum_{i=1}^{k} \mathrm{d}_i - 2\mathrm{d}_1 < 2M$, we have

$$\frac{\sum_{i=1}^{k} \mathrm{d}_i - 2\mathrm{d}_1 + \sum_{i<j} \mathrm{d}_i \mathrm{d}_j}{2(M+1)} < \left\lceil \frac{\sum_{i<j} \mathrm{d}_i \mathrm{d}_j}{2M} \right\rceil + 1$$

and thus the conclusion follows.

Lemma 11.2. *When a new edge (u,v) connecting communities $C(u)$ and $C(v)$ is introduced, $C(v)$ (or $C(u)$) is the best candidate for u (or v) if it should ever change its membership.*

Proof. Let $C \equiv C(u)$ and $D \equiv C(v)$. Recall the outer force that a community S applies to vertex u is $F_{\text{out}}^{S}(u) = e_u^S - \frac{\mathrm{d}_u \mathrm{d}_{\text{outS}}}{2M}$. We will show that the presence of edge (u,v) will strengthen $F_{\text{out}}^{D}(u)$ while weaken the other outer forces $F_{\text{out}}^{S}(u)$, i.e, we show that $F_{\text{out}}^{D}(u)$ increases while $F_{\text{out}}^{S}(u)$ decreases for all $S \notin \{C, D\}$.

$$\begin{aligned}
&F_{\text{out}}^{D}(u)_{\text{new}} - F_{\text{out}}^{D}(u)_{\text{old}} \\
&= \left(e_u^D + 1 - \frac{(\mathrm{d}_u+1)(\mathrm{d}_{\text{outD}}+1)}{2(M+1)}\right) - \left(e_u^D - \frac{\mathrm{d}_u \mathrm{d}_{\text{outD}}}{2M}\right) \\
&= \frac{2M + \mathrm{d}_u \mathrm{d}_{\text{outD}}}{2M} - \frac{\mathrm{d}_u \mathrm{d}_{\text{outD}} + \mathrm{d}_{\text{outD}} + \mathrm{d}_u + 1}{2(M+1)} \\
&\geq \frac{2M + \mathrm{d}_u \mathrm{d}_{\text{outD}}}{2(M+1)} - \frac{\mathrm{d}_u \mathrm{d}_{\text{outD}} + \mathrm{d}_{\text{outD}} + \mathrm{d}_u + 1}{2(M+1)} > 0
\end{aligned}$$

and thus $F_{\text{out}}^{D}(u)$ is strengthen when (u,v) is introduced. Furthermore, for any community $S \in \mathscr{C}$ and $S \notin \{C, D\}$,

$$\begin{aligned}
F_{\text{out}}^{S}(u)_{\text{new}} - F_{\text{out}}^{S}(u)_{\text{old}} &= \left(e_u^S - \frac{(\mathrm{d}_u+1)\mathrm{d}_{\text{outS}}}{2(M+1)}\right) - \left(e_u^S - \frac{\mathrm{d}_u \mathrm{d}_{\text{outS}}}{2M}\right) \\
&= \mathrm{d}_{\text{outS}} \left(\frac{\mathrm{d}_u}{2M} - \frac{\mathrm{d}_u+1}{2(M+1)}\right) < 0,
\end{aligned}$$

which implies $F_{\text{out}}^{S}(u)$ is weaken when (u,v) is connected. Hence, the conclusion follows.

Theorem 11.3. *Assume that a new edge (u,v) is added to the network. Let $C \equiv C(u)$ and $D \equiv C(v)$. If $\Delta q_{u,C,D} \equiv 4(M+1)(e_D^u + 1 - e_C^u) + e_C^u(2\mathrm{d}_D - 2\mathrm{d}_u - e_C^u) - 2(\mathrm{d}_u + 1)$ $(\mathrm{d}_u + 1 + \mathrm{d}_D - \mathrm{d}_C) > 0$ then joining u to D will increase the overall modularity.*

Proof. Node u should leave its current community C and join in D if $\mathscr{Q}_{D+u} + \mathscr{Q}_{C-u} > \mathscr{Q}_C + \mathscr{Q}_D$, or equivalently,

Algorithm 3 Node_Removal

```
Input: Node $u \in C$ to be removed; Current structure $\mathscr{C}_t$.
Output: An updated structure $\mathscr{C}_{t+1}$.
 1: $i \leftarrow 1$;
 2: while $N(u) \neq \emptyset$ do
 3:     $S_i = \{$Nodes found by a 3-clique percolation on $v \in N(u)\}$;
 4:     if $S_i == \emptyset$ then
 5:         $S_i \leftarrow \{v\}$;
 6:     end if
 7:     $N(u) \leftarrow N(u) \backslash S_i$;
 8:     $i \leftarrow i+1$;
 9: end while
10: Let each $S_i$ and singleton in $N(u)$ consider its best communities;
11: Update $\mathscr{C}_t$;
```

$$\frac{m_D+e_D+1}{M+1}-\frac{(\mathrm{d}_D+\mathrm{d}_u+2)^2}{4(M+1)^2}+\frac{m_C-e_C}{M+1}-\frac{(\mathrm{d}_C-\mathrm{d}_u-e_C)^2}{4(M+1)^2}>\frac{m_D}{M+1}$$
$$-\frac{(\mathrm{d}_D+1)^2}{4(M+1)^2}+\frac{m_C}{M+1}-\frac{(\mathrm{d}_C+1)^2}{4(M+1)^2}\Leftrightarrow 4(M+1)(e_D+1-e_C)$$
$$+e_C(2\mathrm{d}_D-2\mathrm{d}_u-e_C)-2(\mathrm{d}_u+1)(\mathrm{d}_u+1+\mathrm{d}_D-\mathrm{d}_C)>0.$$

Corollary 11.1. *If the condition in Theorem 11.3 is not satisfied, then neither u nor its neighbors should be moved to D.*

Proof. The proof follows from Theorem 11.3.

11.3.1.3 Node Removal

When an existing node u of a community C is removed at time t, all of its adjacent edges are removed as a result. This case is challenging in the sense that the resulting community is very complicated: it can be either unchanged or broken into smaller pieces and could probably be merged with other communities. To give a sense of it, let's consider two extreme cases when a single degree node and a node with highest degree in a community is removed. If a single degree node is removed, it leaves the resulted community unchanged (Lemma 11.4). However, when a highest degree vertex is removed, the current community might be disconnected and broken in to smaller pieces which then are merged to other communities as depicted in Fig. 11.2c. Therefore, identifying the leftover structure of C is a crucial part once a vertex in C is removed.

To quickly and efficiently handle this task, we utilize the clique percolation method presented in [2]. In particular, when a vertex u is removed from C, we place a 3-clique at one of its neighbor and let the clique percolates until no vertices in C are discovered (Fig. 11.2d). We then let the remaining communities of C choose their best communities. The algorithm is presented in Algorithm 3.

Algorithm 4 Edge_Removal

```
Input: Edge $(u,v)$ to be removed; Current structure $\mathscr{C}_t$.
Output: An updated clustering $\mathscr{C}_{t+1}$.
 1: if $(u,v)$ is a single edge then
 2:     $\mathscr{C}_{t+1} = (\mathscr{C}_t \backslash \{u,v\}) \cup \{u\} \cup \{v\}$;
 3: else if Either $u$ (or $v$) is of degree one then
 4:     $\mathscr{C}_{t+1} = (\mathscr{C}_t \backslash C(u)) \cup \{u\} \cup \{C(u) \backslash u\}$;
 5: else if $C(u) \neq C(v)$ then
 6:     $\mathscr{C}_{t+1} = \mathscr{C}_t$;
 7: else
 8:     % Now $(u,v)$ is inside a community $C$ %
 9:     $L$ = {Maximal "quasi-cliques" in $C$};
10:     Let the singletons in $C \backslash L$ consider their best communities;
11: end if
12: Update $\mathscr{C}_{t+1}$;
```

11.3.1.4 Edge Removal

In the last case when an edge $e = (u,v)$ is about to be removed, we divide further into four subcases: (1) e is a single edge connecting only u and v where u and v are of single degree, (2) either u or v has degree one, (3) e is an inter-community link connecting $C(u)$ and $C(v)$, and (4) e is an inter-community link. If e is an single edge, it is clear that removing e will result in the same community structure plus two singletons of u and v themselves. The same reaction applies to the second subcase when either u or v has single degree due to Lemma 11.4, thus results in the prior network structure plus u (or v). When e is an inter-community link, the removal of e will strengthen the current network communities (Lemma 11.3) and hence, we just make no change to the community structure.

The last but most complicated case happens when an intra-community link is deleted. As depicted in Fig. 11.2b, removing this kind of edge often leaves the community unchanged if the community itself is densely connected; however, the target module will be divided if it contains substructures which are less attractive or loosely connected to each other. Therefore, the problem of identifying the structure of the remaining modules becomes very complicated. Theorem 11.4 provides us a convenient tool to test for community bi-division when an intra-community link is removed from the host community C. However, it requires an intensive look for all subsets of C, which may be time consuming. Note that prior to the removal of (u,v), the community C hosting this link should contain dense connections within itself and thus, the removal of (u,v) should leave some sort of "quasi-clique" structure inside C. Therefore, we find all maximal "quasi-cliques" within the current community and have them (as well as leftover singletons) determine their best communities to join in. The detailed procedure is described in Algorithm 4.

Lemma 11.3. *If C_1 and C_2 are two communities of G, then the removal of an inter-community link connecting them will strengthen modularity contributions of both C_1 and C_2.*

Proof. Let $\mathcal{Q}_1$ and $\mathcal{Q}'_1$ be the modularities of C_1 before and after the removal of that link. We show that $\mathcal{Q}'_1 > \mathcal{Q}_1$ (and similarly, $\mathcal{Q}'_2 > \mathcal{Q}_2$) and thus, C_1 and C_2 contribute higher modularities to the network. Now,

$$\mathcal{Q}'_1 - \mathcal{Q}_1 = \left(\frac{m_1}{M-1} - \frac{(\mathrm{d}_1-1)^2}{4(M-1)^2}\right) - \left(\frac{m_1}{M} - \frac{\mathrm{d}_1^2}{4M^2}\right)$$
$$= m_1\left(\frac{1}{M-1} - \frac{1}{M}\right) + \frac{1}{4}\left(\frac{\mathrm{d}_1}{M} - \frac{\mathrm{d}_1-1}{M-1}\right)\left(\frac{\mathrm{d}_1}{M} + \frac{\mathrm{d}_1-1}{M-1}\right).$$

Since all terms are all positive, $\mathcal{Q}'_1 - \mathcal{Q}_1 > 0$. The same technique applies to show that $\mathcal{Q}'_2 > \mathcal{Q}_2$.

Lemma 11.4. *The removal of (u,v) inside a community C where only u or v is of degree one will not separate C.*

Proof. Assume the contradiction, i.e. after the removal of (u,v) where $\mathrm{d}_u = 1$, C is broken into smaller communities X_1, X_2,..., X_k which contribute higher modularity: $\mathcal{Q}_{X_1} + ... + \mathcal{Q}_{X_k} > \mathcal{Q}_C$. W.L.O.G, suppose u connected to X_1 prior to its removal. It follows that $\mathcal{Q}_{X_1+u} > \mathcal{Q}_{X_1}$ and thus $\mathcal{Q}_{X_1+u} + ... + \mathcal{Q}_{X_k} > \mathcal{Q}_C$, which raises a contradiction since C is originally a community.

Lemma 11.5. *(Separation of a community) Let $C_1 \subseteq C$ and $C_2 = C \backslash C_1$ be two disjoint subsets of C. $(\mathscr{C} \backslash C) \cup \{C_1, C_2\}$ is a community structure with higher modularity when an edge crossing C_1 and C_2 is removed, i.e, C should be separated into C_1 and C_2, if and only if $e_{12} < \frac{\mathrm{d}_1\mathrm{d}_2 - \mathrm{d}_C + 1}{2(M-1)} + 1$.*

Proof. Let $\mathcal{Q}'_1$, $\mathcal{Q}'_2$ and $\mathcal{Q}'_C$ denote the modularity contribution of C_1, C_2, and C after an edge crossing (X_1, X_2) has been removed. Now,

$$e_{12} < \frac{\mathrm{d}_1\mathrm{d}_2 - \mathrm{d}_C + 1}{2(M-1)} + 1 \Leftrightarrow \frac{2\mathrm{d}_1\mathrm{d}_2 - 2\mathrm{d}_C + 2}{4(M-1)^2} > \frac{e_{12}-1}{M-1}$$
$$\Leftrightarrow \frac{(\mathrm{d}_1+\mathrm{d}_2-2)^2}{4(M-1)^2} - \frac{(\mathrm{d}_1-1)^2}{4(M-1)^2} - \frac{(\mathrm{d}_2-1)^2}{4(M-1)^2}$$
$$> \frac{m_1+m_2+e_{12}-1}{M-1} - \frac{m_1-1}{M-1} - \frac{m_2-1}{M-1}$$
$$\Leftrightarrow \frac{m_1-1}{M-1} - \frac{(\mathrm{d}_1-1)^2}{4(M-1)^2} + \frac{m_2-1}{M} - \frac{(\mathrm{d}_2-1)^2}{4(M-1)^2}$$
$$> \frac{m_1+m_2+e_{12}-1}{M-1} - \frac{(\mathrm{d}_1+\mathrm{d}_2-2)^2}{4(M-1)^2}$$
$$\Leftrightarrow \mathcal{Q}'_1 + \mathcal{Q}'_2 > \mathcal{Q}'_C.$$

Thus, the conclusion follows.

Theorem 11.4. *(Testing of module separation) For any community C, let α and β be the lowest and the second highest degree of vertices in C, respectively. Assume that an edge e is removed from C. If there do not exist subsets $C_1 \subseteq C$ and $C_2 \equiv C \backslash C_1$ such that*

1. *e is crossing C_1 and C_2*
2. $\dfrac{\min\{\alpha(\mathrm{d}_C - \alpha), \beta(\mathrm{d}_C - \beta)\}}{2M} < e_{12} < \dfrac{(\mathrm{d}_C - 2)^2}{8(M-1)} + 1,$

then any bi-division of C will not benefit the overall $\mathcal{Q}$.

Proof. From Lemma 11.5, it follows that in order to really benefit the overall modularity we must have

$$\frac{\mathrm{d}_1\mathrm{d}_2}{2M} < e_{12} < \frac{\mathrm{d}_1\mathrm{d}_2 + 1}{2(M-1)} + 1.$$

Since $\mathrm{d}_1 + \mathrm{d}_2 = \mathrm{d}_C$, it follows that

$$e_{12} < \frac{\mathrm{d}_1\mathrm{d}_2 - d + 1}{2(M-1)} + 1 \leq \frac{\frac{(\mathrm{d}_1+\mathrm{d}_2)^2}{4} - d + 1}{2(M-1)} + 1 \leq \frac{\frac{\mathrm{d}_C^2}{4} - \mathrm{d}_C + 1}{2(M-1)} + 1 = \frac{(\mathrm{d}_C - 2)^2}{8(M-1)} + 1.$$

For a lower bound of the LHS inequality, we rewrite $\mathrm{d}_1\mathrm{d}_2$ as $\mathrm{d}_1\mathrm{d}_2 = \mathrm{d}_1(d - \mathrm{d}_1) = \mathrm{d}_1 d - \mathrm{d}_1^2$ and find the non-zero minimum value on the range $\mathrm{d}_1 \in [\alpha, \beta]$. By taking the derivative and solving for critical point, $\mathrm{d}_1 d - \mathrm{d}_1^2$ is minimized either at $\mathrm{d}_1 = \alpha$ or $\mathrm{d}_1 = \beta$. Therefore,

$$\frac{\min\{\alpha(d-\alpha), \beta(d-\beta)\}}{2M} \leq \frac{\mathrm{d}_1\mathrm{d}_2}{2M} < e_{12} \leq \frac{(\mathrm{d}_C - 2)^2}{8(M-1)} + 1.$$

Finally, our main algorithm *QCA* for quickly updating community structures of dynamic social networks is presented in Algorithm 5.

11.4 Experimental Results

In this section, we present the experimental results of our *QCA* algorithm on identifying and updating the network community structures of dynamic social networks. To illustrate the strength and effectiveness of our approach, we choose three popular real-world social networks including *ENRON* email network [11], *arXiv* e-print citation network (provided by the KDD cup 2003 [12]), and Facebook online social network [13]. The static method we are comparing to is the one proposed by Blondel *et al* [5], which we refer to as Blondel method or static method. In addition to the static method, we further compare *QCA* to a recent dynamic

Algorithm 5 Quick Community Adaptation (*QCA*)

```
Input: $G \equiv G_0 = (V_0, E_0)$, $\mathscr{E} = \{\mathscr{E}_1, \mathscr{E}_2, .., \mathscr{E}_s\}$ a collection of simple events
Output: Community structure $\mathscr{C}_t$ of $G^t$ at time $t$.
 1: Use [5] to find an initial community clustering $\mathscr{C}_0$ of $G_0$;
 2: for ($t \leftarrow 1$ to s) do
 3:     $\mathscr{C}_t \leftarrow \mathscr{C}_{t-1}$;
 4:     if $\mathscr{E}_t = newNode(u)$ then
 5:         New_Node($\mathscr{C}_t, u$);
 6:     else if $\mathscr{E}_t = newEdge((u,v))$ then
 7:         New_Edge($\mathscr{C}_t, (u,v)$);
 8:     else if $\mathscr{E}_t = removeNode(u)$ then
 9:         Remove_Node($\mathscr{C}_t, u$);
10:     else
11:         Remove_Edge($\mathscr{C}_t, (u,v)$);
12:     end if
13: end for
```

adaptive method called *MIEN* proposed in [14]. Basically, *MIEN* tries to compress and decompress network modules into nodes in order to adapt with the changes and uses fast modularity method [4] to keep the network structure updated. In particular, we will show in the experiments the following quantities (1) modularity values (2) the quality of the identified network communities through NMI scores and (3) the processing time of our *QCA* in comparison with other methods. The above networks expose to contain clear community structure due to their high modularities, which is the main reason for them to be chosen.

In order to quantify the quality of the identified community structure, i.e. the similarity between the identified communities and the ground truth, we adopt a well known measure in Information Theory called *Normalized Mutual Information (NMI)*. NMI has been proven to be reliable and is currently used in testing community detection algorithms [3]. Basically, $NMI(U,V)$ equals 1 if structures U and V are identical and equals 0 if they are totally separated. Due to space limit, the readers are encouraged to read [3] for NMI formulas.

For each network, time information is extracted in different ways and a portion of the network data (usually the first network snapshot) is collected to form the basic network community structure. Our *QCA* algorithm takes into account that basic community structure and runs only on the network changes whereas the static method has to be performed on the whole network snapshot up to each time point.

11.4.1 ENRON Email Network

The *Enron* email network contains email messages data from about 150 users, mostly senior management of *Enron* Inc., from January 1999 to July 2002 [11]. Each email address is represented by an unique identification number in the dataset and each link corresponds to a message sent between the sender and the receiver.

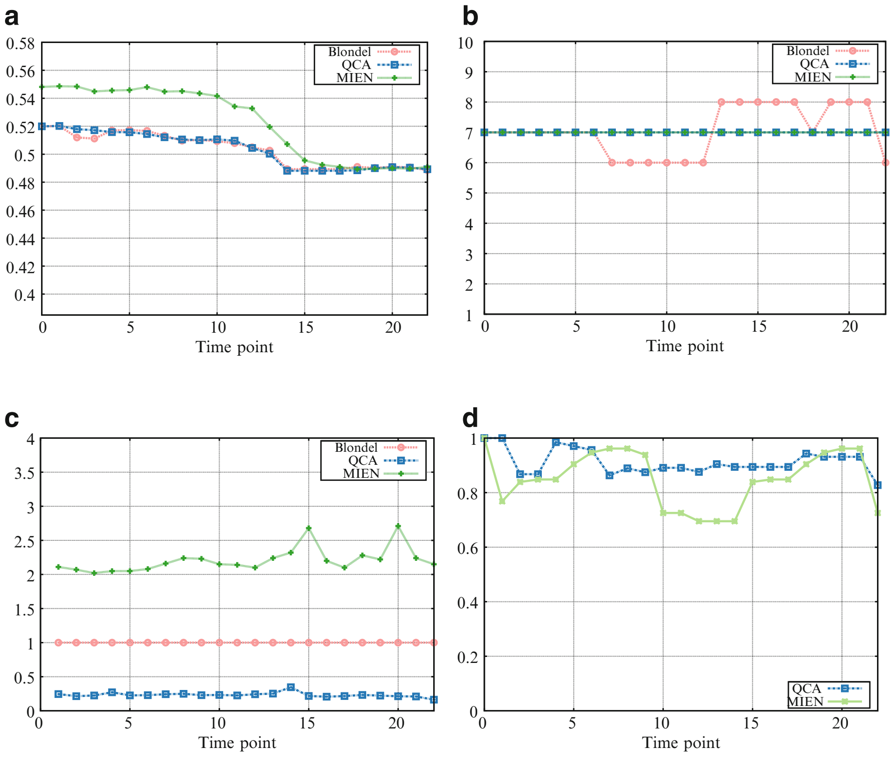


Fig. 11.3 Simulation results on Enron Email Network dataset. (**a**) Modularity (**b**) Number of Communities (**c**) Running Time(s) (**d**) NMI

After a data refinement process, we choose 50% of total links to form a basic community structure of the network with 7 major communities, and simulate the network evolution via a series of 21 growing snapshots in which roughly 10^3 links are added at a time.

We first evaluate the modularity values computed by *QCA*, *MIEN*, and Blondel method. As shown in Fig. 11.3a, our *QCA* algorithm archives competitively higher modularities than the static method but a little bit less than *MIEN* method while maintaining the nearly same number of communities of the other two (Fig. 11.3b). In particular, the modularity values produced by *QCA* very well approximate those found by static method with lesser variation. There are reasons for that. Recall that our *QCA* algorithm takes into account the basic community structures detected by the static method (at the first snapshot) and processes on network changes only. Knowing the basic network community structure is a great advantage of our *QCA* algorithm: it can avoid the hassle of searching and computing from scratch to update the network with changes. In fact, *QCA* uses the basic structure for finding and quickly updating the local optimal communities to adapt with changes introduced during the network evolution.

The running time of *QCA* and the static method in this small network are relatively close: the static method requires 1 s to complete each of its tasks while our *QCA* does not even ask for one (Fig. 11.3c). In this dataset, *MIEN* requires a little more time (1.5 s in average) to complete the task. Time and computational cost are significantly reduced in *QCA* since our algorithms only take into account the network changes while the static method has to work on the whole network every time.

Due to the lack of the proper information about real communities in *Enron* Inc. (and in other datasets), we use community structure identified by the static method as a reference to the ground truth. The quantity $NMI(QCA, Blondel)$ (or $NMI(QCA)$ in short) indicates how community labels assigned by *QCA* similar to those of the ground truth computed at every timepoint. A NMI value of 1 means two assignments are identical and 0 implies the opposite. As one can see in Fig. 11.3d, both the NMI scores of *MIEN* method and ours are very high and relatively close to 1, indicating that in this Enron email network, both our *QCA* and *MIEN* algorithm are able to identify high quality community structure with high modularity; however, only our method significantly reduces the processing time and computational requirement.

11.4.2 arXiv E-Print Citation Network

The arXiv e-print citation network [12], which was initialized in 1991, has become an essential mean of assessing research results in various areas including physics and computer sciences. At the time the dataset was collected, the network already contained more than 225 K fulltext articles and was growing of over 40 K submissions each year, ranging from January 1996 to May 2003.

In our experiments, citation links of the first two years 1996 and 1997 were taken into account to form the basic community structure of our *QCA* method. In order to simulate the network evolution, a total of 30 time dependent snapshots of the arXiv citation network are created on a two-month regular basis in the duration between January 1998 and January 2003.

We compare modularity results obtained by *QCA* algorithm at each network snapshot to Blondel method as well as to *MIEN* method. It reveals from Fig. 11.4a that the modularities returned by *QCA* are very close to those obtained by the static method with much more stabler and are far higher than those of *MIEN*. In particular, the modularity values produced by *QCA* algorithm cover from 94% up to 100% that of Blondel method and from 6% to 10% higher than *MIEN*. Moreover, our method reveals a much better network community structure since it discovers many more communities than both the static method and *MIEN* as the network evolves (Fig. 11.4b). This can be explained based on the resolution limit of modularity [9]: the static method might disregard some small communities and tend to combines them in order to maximize the overall network modularity.

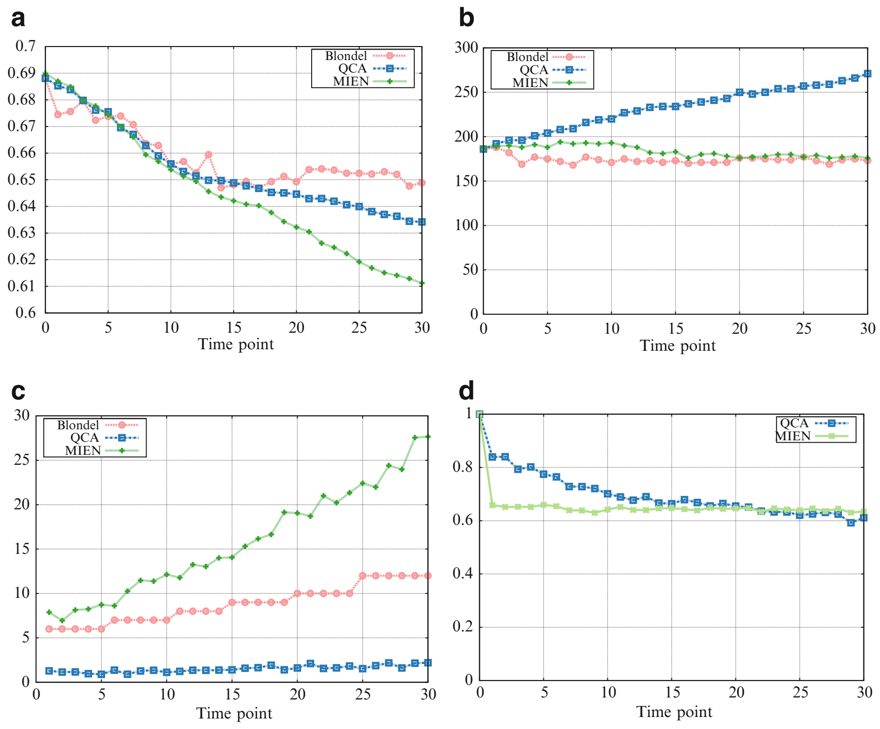


Fig. 11.4 Simulation results on ArXiv e-Print citation network. (**a**) Modularity (**b**) Number of Communities (**c**) Running Time(s) (**d**) NMI

Second observation on running time shows that *QCA* outperforms the static method as well as its competitor *MIEN*: *QCA* takes at most 2 s to complete updating the network structure while Blondel method requires more than triple that amount of time (Fig. 11.4d) and *MIEN* asks for more than 5x time. In addition, higher NMI scores of *QCA* than *MIEN* methods (Fig. 11.4) implies network communities identified by our approach are not only of high similarity to the ground truth but also more precise than that detected by *MIEN*, meanwhile the computational cost and the running time are significantly reduced.

11.4.3 Facebook Social Network

This data set contains friendship information (i.e. who is friend with whom and wall posts) among New Orleans regional network on Facebook, spanning from Sep 2006 to Jan 2009 [13]. To collect the information, the authors created several Facebook accounts, joined each to the regional network, started crawling from a single user and visited all friends in a breath-first-search fashion. The data set contains more

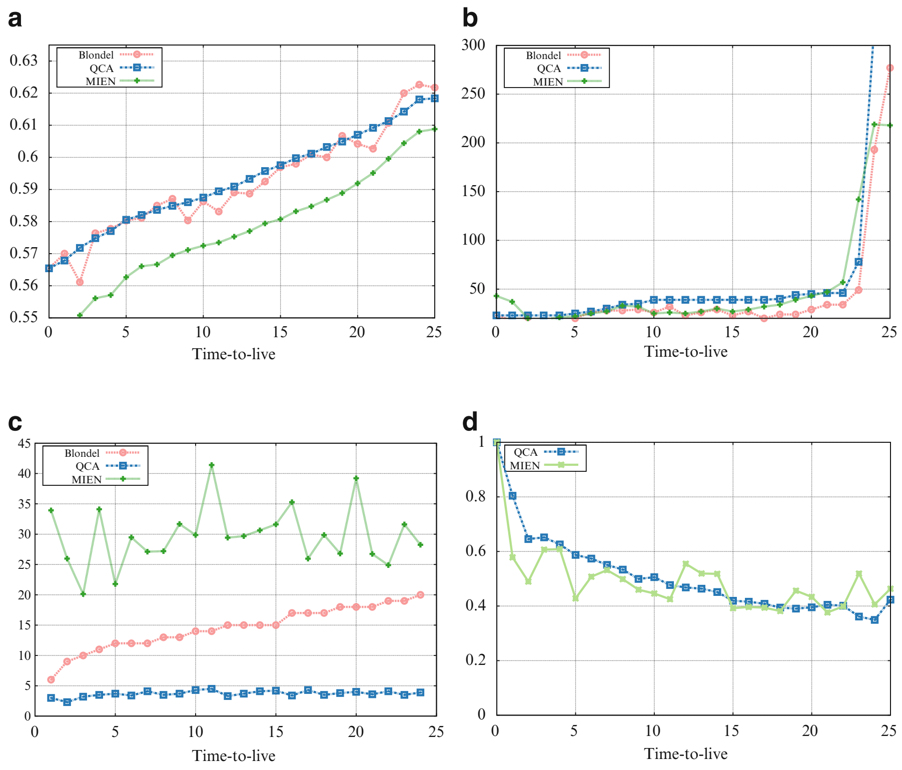


Fig. 11.5 Simulation results on facebook social network. (**a**) Modularity (**b**) Number of Communities (**c**) Running Time(s) (**d**) NMI

than 60 K nodes (users) connected by more than 1.5 million friendship links with an average node degree of 23.5. In our experiments, the nodes and links from Sep 2006 to Dec 2006 are used to form the basic community structure of the network and each network snapshot is recored after every month during Jan 2007 to Jan 2009 for a total of 25 network snapshots.

Evaluation depicted in Fig. 11.5a reveals that *QCA* algorithm achieves competitive modularities in comparison with the static method, and again far better than those obtained by *MIEN*. In the general trend, the line representing *QCA* results closely approximates that of the static method with much more stable. Moreover, the two final modularity values at the end of the experiment are relatively the same, which means that our adaptive method performs competitively with the static method running on the whole network.

Figure 11.11.5c describes the running time of the three methods on the Facebook data set. As one can see from this figure, *QCA* takes at least 3 s and at the most 4.5 s to successfully compute and update every network snapshot whereas the static method, again, requires more than triple processing time. *MIEN* method really suffers on this large scale network when requiring more than 10x amount of *QCA* running time. This result confirms the effectiveness of our adaptive method when applied to real-world social networks where a centralized algorithm may not be able to detect a good network community structure in a timely manner.

However, there is a limitation of *QCA* algorithm we observe on this large network and want to point out here: As the the duration of network evolution lasts longer over time (i.e. the number of network snapshots increases), our method tends to divide the network into smaller communities to maximize the local modularity, thus results in an increasing number of communities and a decreasing of NMI scores. Figure 11.5b and 11.5d describes this observation. For instance, at snapshot #12 (a year after Dec 2006), the NMI score is approximately $1/2$ and gets decaying after this timepoint. It implies a refreshment of network community structure is required at this time, after a long enough duration. This is reasonable since activities on an online social network, especially on Facebook social network, tend to come and go rapidly and local adaptive procedures are not enough to reflect the whole network topology over a long period of time.

11.5 Application: Social-Aware Routing in MANETs

In this section, we present an application where the detection of network community structures plays an important role in routing strategies in MANETs. A MANET is a dynamic wireless network with or without the underlying infrastructure, in which each node can move freely in any direction and organize itself in an arbitrary manner. Due to nodes mobility and unstable links nature of a MANET, designing an efficient routing scheme has become one of the most important and challenging problems on MANETs.

Recent researches have shown that MANETs exhibit the properties of social networks [15–17] and social-aware algorithms for network routing are of great potential. This is due to the fact that people have a natural tendency to form groups or communities in communication networks, where individuals inside each community communicate with each other more frequent than with people outside. This social property is nicely reflected to the underlying MANETs by the existence of groups of nodes where each group is densely connected inside than outside. This resembles the idea of *community structure* in MANETs.

Multiple routing strategies [16]–[18] based on the discovery of network community structures have provided significant enhancement over traditional methods. However, the community detection methods utilized in those strategies are not applicable for dynamic MANETs since they have to recompute the network structure whenever changes to the network topology are introduced, which results in significant computational costs and processing time. Therefore, employing an adaptive community structure detection algorithm as a core will provide speedup as well as robustness to routing strategies in MANETs.

We evaluate the following five routing strategies (1) *WAIT*: the source node waits and keeps sending or forwarding the messages until it meets the destination node (2) *MCP*: A node keeps forwarding the messages until they reach the maximum number of hops, (3) *LABEL*: A node forwards or sends the messages to all members in the destination community [15], (4) *QCA*: A Label version utilizing *QCA* as the dynamic community detection method, and lastly (5) *MIEN* A social-aware routing strategy on MANETs [14].

Even thought the *WAIT* and *MCP* algorithms are very simple and straightforward to understand, they provide us helpful information about the lower and upper bounds for message delivery ratio, time redundancy as well as message redundancy. *LABEL* forwarding strategy works as follow: it first finds the community structure of the underlying MANET, assigns each community with the same label and then exclusively forwards messages to destinations, or to next-hop nodes having the same labels as the destinations. *MIEN* forwarding method utilizes MIEN algorithm as a subroutine. *QCA* routing strategy, instead of using a static community detection method, utilizes *QCA* algorithm for adaptively updating the network community structure and then uses the newly updated structure to inform the routing strategy for forwarding messages.

We choose Reality Mining data set [19] provided by the MIT Media Lab to test our proposed algorithm. The Reality Mining data set contains communication, proximity, location, and activity information from 100 students at MIT over the course of the 2004–2005 academic year. In particular, the data set includes call logs, Bluetooth devices in proximity, cell tower IDs, application usage, and phone status (such as charging and idle) of the participated students of over 350,000 hours. In this chapter, we take into account the Bluetooth information to form the underlying MANET and evaluate the performance of the above five routing strategies.

For each routing method, we evaluate the followings (1) Delivery ratio: The portion of successfully delivered over the total number of messages (2) Average delivery time: Average time for a message to be delivered. (3) Average number of duplicated messages for each sent message. In particular, a total of 1,000 messages are created and uniformly distributed during the experiment duration and each message can not exist longer than a threshold *time-to-live*. The experimental results are shown in Figs. 11.6a, 11.6b, and 11.6c.

Figure 11.6a describes the delivery ratio as a function of *time-to-live*. As revealed by this figure, *QCA* achieves much better delivery ratio than *MIEN* as well as *LABEL* and far better than *WAIT*. This means that *QCA* routing strategy successfully delivers many more messages from the source nodes to the destinations than the others. Moreover, as *time-to-live* increases, the delivery ratio of *QCA* tends to approximate the ratio of *MCP*, the strategy with highest delivery ratio.

Comparison on delivery time shows that *QCA* requires less time and gets messages delivered successfully faster than *LABEL*, as depicted in Fig. 11.6c. It even requires less delivery time in comparison with the social-aware method *MIEN*. This can be explained as the static community structures in *LABEL* can possibly get message forwarded to a wrong community when the destinations eventually change their communities during the experiment. Both *QCA* and *MIEN*, on the

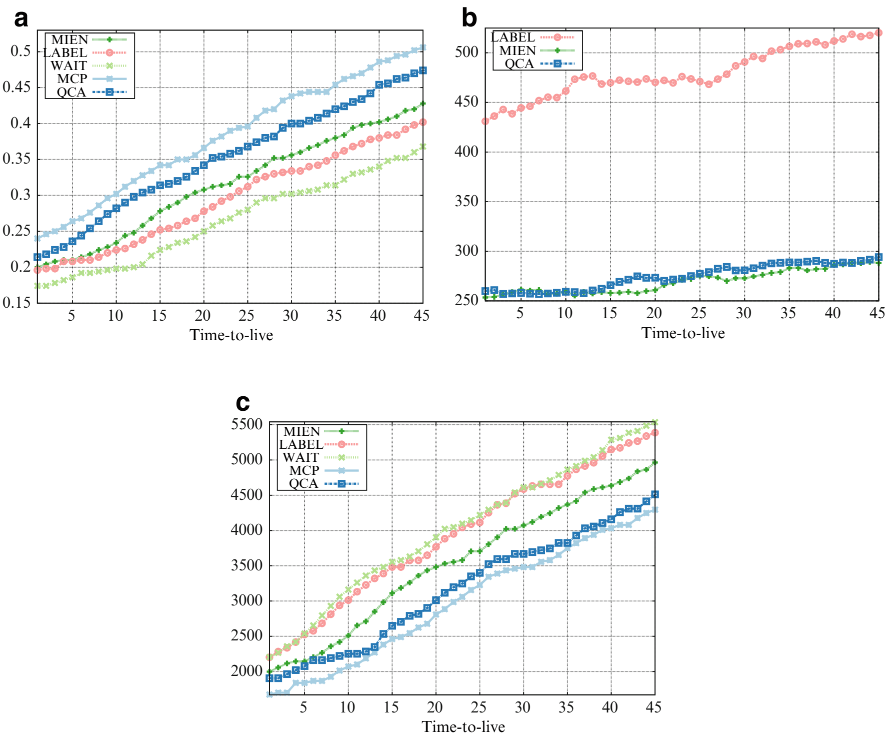


Fig. 11.6 Experimental results on the reality mining data set. (**a**) Delivery Ratio (**b**) Average Duplicate Message (**c**) Average Delivery Time

other hand, captures and updates the community structures on-the-fly as changes occur, thus achieves better results. Since *MIEN* needs to compress and decompress the network communities whenever network evolves, it may disregard the existence of newly formed communities and thus, may requires extra time to forward the messages.

The numbers of duplicate messages presented in Fig. 11.6b indicate that both *QCA* and *MIEN* achieves the best results. The numbers of duplicated messages of *MCP* method are weight higher than those of the others and are not plotted. In fact, the results of *QCA* and *MIEN* are relatively close and tend to approximate each other as *time-to-live* increases.

In conclusion, *QCA* is the best social-aware routing algorithm among five routing strategies since its delivery ratio, delivery time and redundancy outperform those of the other methods and are only below *MCP* while the number of duplicate messages are much lower. *QCA* also shows a significant improvement over the naive *LABEL* method which uses a static community detection method and thus, confirms the applicability of our adaptive algorithm to routing strategies in MANETs.

11.6 Application: Worm Containment on Social Networks

In this section, we brighten the applicability of QCA method via another practical application in Worm containment on online social networks. Since their introduction, popular social network sites such as Facebook, Twitter, Bebo, and MySpace have attracted millions of users worldwide, many of whom have integrated those sites into their everyday lives. On the bright side, online social networks are ideal places for people to keep in touch with friends and colleagues, to share their common interests, to hold discussions in forums or just simply to socialize online. However, on the other side, social networks are also fertile grounds for the rapid propagation of malicious softwares (such as viruses or worms) and false information.

Facebook, one of the most famous online social networks, experienced a wide propagation of a trojan worm named "Koobface" in late 2008. Koobface made its way not only through Facebook but also Bebo, MySpace, and Friendster social networks [20, 21]. Once a user is infected, this worm scans through the current user's profile and sends out fake messages or wall posts to everyone in the user's friend list with titles or comments appeal to people's curiosity. If one of the user's friends, attracted by the comments without a shadow of doubt, clicks on the link and installs the fake "flash player", he will be infected. Koobface's life will then cycle on this newly infected machine. Since people are able to access social network sites via cell phones nowadays, worm's targets are now not only computers but also mobile devices.

The problem of worm containment becomes more and more complicated on a dynamic social network as this kind of network evolves and changes rapidly over time. The dynamics of social networks thus gives worms more chances to spread out faster and wider as they could flexibly switch between new and existing users in order to propagate. Therefore, the problem of containing worm propagation on social networks is extremely challenging in the sense that a good solution at the previous time step might not be sufficient or effective at the next time step. Although one can recompute a new solution at each time the network changes, doing so would result in heavy computational costs and time consuming as well as worms spreading out wider during the recomputing process. A better solution should quickly and adaptively update the current worm containing strategy based on changes in network topology, thus could avoid the hassle of recomputing from scratch whenever the network evolves.

There are many proposed methods dealing with worm containment on computer networks by either using a multi-resolution approach to enhance the power of threshold-based detection [22], or using a simplification of the Threshold Random Walk scan detector [23], or by measuring the velocity of the number of new connections and infected hosts [24], or using fast and efficient worm signature generation [25, 26]. There are also several method proposed for cellular and mobile networks [27–29]. However, all of these above approaches fail to take into account the community structure as well as the dynamics of social networks, thus might not be appropriate for our problem. A recent work [30] proposed a social-based patching scheme for worm containment on cellular networks. However, this method encounters the following limitations on a real social networks (1) its clustered partitioning does not necessarily reflect the natural network community structure, (2) it requires the number of clusters k (which is generally unknown for social networks) must be specified beforehand, and (3) it exposes weaknesses when dealing with dynamics of the network.

To overcome these limitations, our approach first utilizes QCA to identify the network community structure and then adaptively keeps this structure fresh and updated as the network evolves. Once the network communities are detected, our patch distribution procedure will select the most influential users from different communities in order to sending patches. These users, as soon as they receive patches, will apply them to first disinfect the worm and then redistribute them to all friends in their communities. These actions will contain worm propagation to only some communities of the social network and prevent it from spreading out to a larger population. To this end, a quick and precise community detection method will definitely help the network administrator to select a more sufficient set of critical users to send patches, thus lowers down the number of sent patches as well as overhead information over the social network.

We next describe our patch distribution procedure. This procedure takes into account the community structure identified from the previous step and selects a set of influential users from each community of the network in order to distribute patches. *Influential users* of a community are ones having the most relationships or connections to other communities. In the point of an attacker view, these influential users are potentially vulnerable since they not only interact actively within their communities but also with people outside, thus, they can easily fool (or be fooled by) people both inside and outside of their communities. On the other point of view, these users are the best candidates for the network defender to distribute patches since they can easily announce and forward patches to other members and non-members.

We present here a quick greedy algorithm for selecting the set of most influential users in each community. In particular, for each community of G, the algorithm starts with picking the user whose number of social connections to outside communities is the highest and temporarily remove this user from the considering community. This process repeats until there is no connection crossing among communities of G. This set of influential users is the candidate for the network defender for distributing patches. The distribution procedure is presented in Algorithm 6. Note that we can directly apply the algorithm for Vertex Cover problem to the patch distribution to get a 2-approximation algorithm; however, doing so would result in heavy computation and much time consuming, especially on very large social networks.

Algorithm 6 Patch Distribution Procedure

```
Input: A community structure $\mathscr{S} = \{S_1, S_2, .., S_p\}$
Output: Set of influential users.
1: $IS = \emptyset$;
2: for $S_i \in S$ do
3:     while $\max_{v \in S_i}\{d^{S_i}_{\text{out}}(v)\} > 0$ do
4:         Let $a \leftarrow \arg\max_{v \in S_i}\{d^{S_i}_{\text{out}}(v)\}$;
5:         $IS = IS \cup a$;
6:         Temporary remove $a$ from $S_i$;
7:     end while
8: end for
9: Send patches to users in $IS$;
```

Results

We present the experimental results of our method on the Facebook network dataset [13] and compare the results with the social based method (Zhu's method [30]) via a weighted version of our community update algorithms. One notable feature of this dataset is time information (stamped at every moment the information was recorded) represents the dynamics of the network, which nicely suits to our adaptive method.

The worm propagation model in our experiments mimics the behavior of the famous "Koobface" worm which once spread out widely on Facebook. In our model, worms are able to explore their victim's friend list and then send out fake messages containing malicious links for propagating. The probability of a victim's friend activating the worm is proportional to communication frequency between the victim and his friends. The time taken for worms to spread out from one user to another is inversely proportional to the communication frequency between this user and his particular friend. Finally, when a worm has successfully infected a user's computer, it will start propagating as soon as this computer connects to a specific social network (Facebook in this case).

When the fraction of infected users (the number of infected users over the number of all users) reaches a threshold α, the detection system raises an alarm and patches will automatically be sent to most influential users selected by Algorithm 6. Once an influential user receives a patch, he will first apply the patch to disinfect the worm and then will have an option to forward this patch to all friends in his community.

Each experiment on this dataset is seeded with 0.02% of users to be infected by worms and worm propagation is simulated through the duration of 2 days. In each experiment, we compare infection rates of the social-based method of Zhu's and ours. The infection rate is computed as a fraction of the remaining infected users over the overal infected ones. The number of clusters k in Zhu's method is set to be 150 (in a static network), 200, and 250 (in dynamic networks), respectively. For each value of k, the alarmming threshold α is set to be 2%, 10%, and 20%, respectively. Each experiment is repeated 1,000 times for consistency.

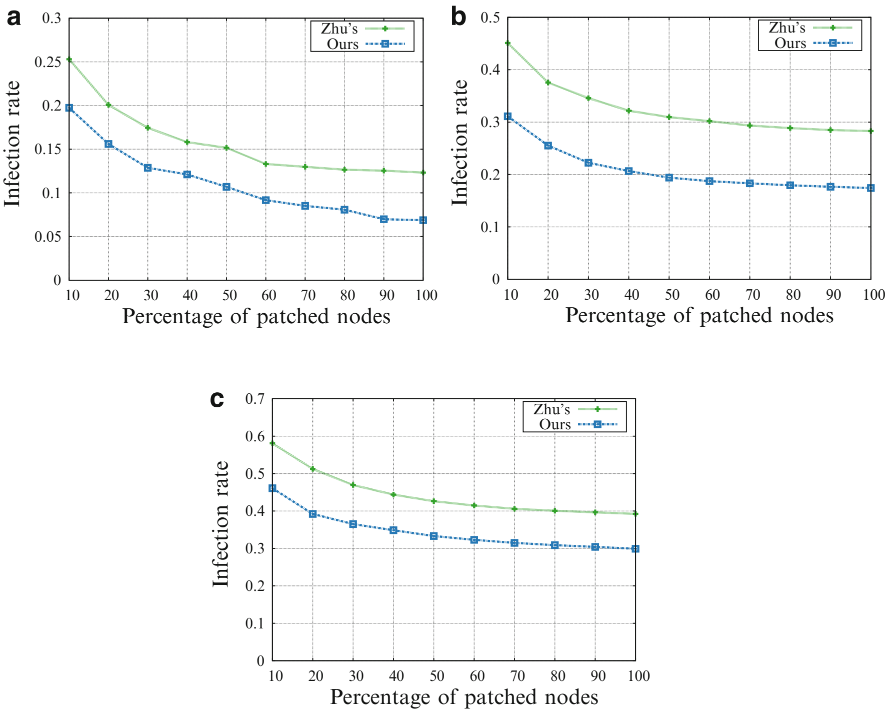


Fig. 11.7 Infection rates on static network with $k = 150$ clusters. (**a**) $\alpha = 2\%$ (**b**) $\alpha = 10\%$ (**c**) $\alpha = 20\%$

Figures 11.7, 11.8 and 11.9 show the results of our experiments for three different values of k and α. We first observe that the longer we wait (the higher the alarm threshold is), the higher number of users we need to send patches to in order to achieve the expected infection rate. For example, with $k = 150$ clusters and an expected infection rate of 0.3, we need to send patches to less than 10% number of users when $\alpha = 2\%$, to more than 15% number of users when $\alpha = 10\%$ and to nearly 90% of total influential users when $\alpha = 20\%$.

Second observation reveals that our approach achieves better infection rates than the social-based method of Zhu in a static version of the social network as depicted in Fig. 11.7. In particular, the infection rates obtained in our method are from 5% to 10% better than those of Zhu's. When the network evolves as new users join in and new social relationships are introduced, we resize the number of cluster k and recompute the infection rates of the social based method with the number of cluster $k = 200$ and $k = 250$, and the alarm threshold $\alpha = 2\%, 10\%$ and 20%, respectively. As depicted in Figs. 11.8 and 11.9, our method, with the power of quickly and adaptively update the network community structure, achieves better infection rates than Zhu's method meanwhile the computational costs and running time is significantly reduced. As discussed, detecting and updating the network

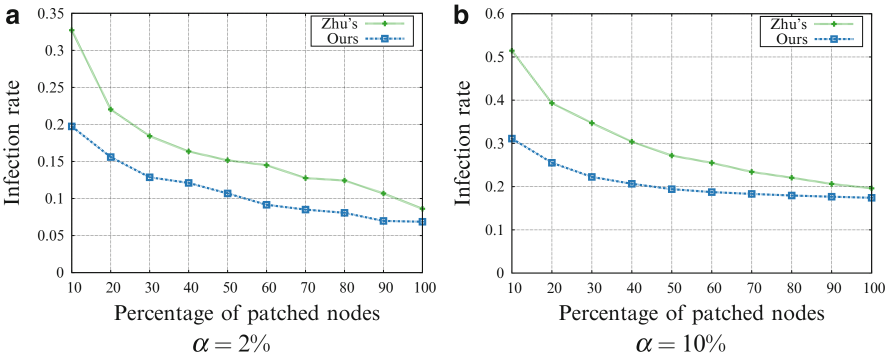


Fig. 11.8 Infection rates on dynamic network with $k = 200$ clusters. (**a**) $\alpha = 2\%$ (**b**) $\alpha = 10\%$

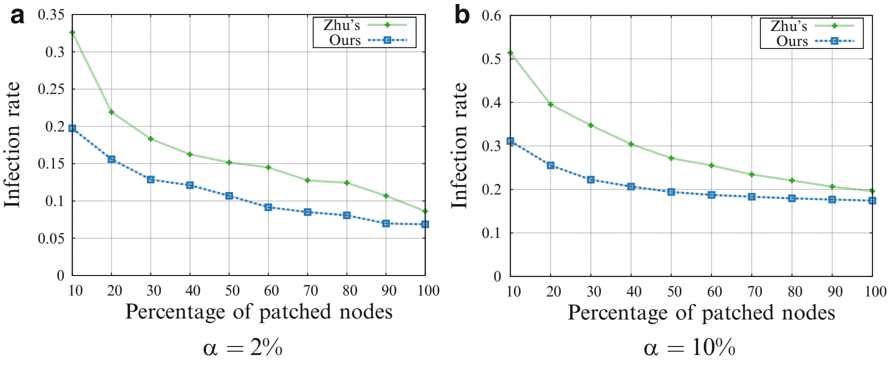


Fig. 11.9 Infection rates on dynamic network with $k = 250$ clusters. (**a**) $\alpha = 2\%$ (**b**) $\alpha = 10\%$

community is the crucial part of a social based patching scheme: a good and up-to-date network community structure will provide the network defender a tighter set of vulnerable users and thus, will help to achieve lower infection rates. Our adaptive algorithm, instead of recomputing the network structure every time changes are introduced, quickly and adaptively updates the network communities on-the-fly. Thanks to this frequently updated community structure, our patch distribution procedure is able to select a better set of influential users, and thus helps in reducing the number of infected users once patches are sent.

We further look more into the behavior of Zhu's method when the number of clusters k varies. We compute and compare the infection rates on Facebook dataset for various k ranging from 1 K to 2.5 K with our approach. We first hope that the more predefined clusters, the better infection rates clustered partitioning method will achieve. However, the experimental results depicted in Fig. 11.10 reveal the opposite. In particular, with a fixed alarming threshold $\alpha = 10\%$ and 60%

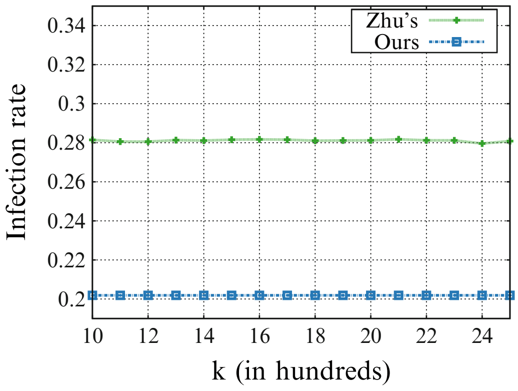


Fig. 11.10 Infection rates at $\alpha = 10\%$ using 60% of the patches nodes

patched nodes, the infection rates achived by Zhu's method do not decrease but ranging near 28% while ours are far better (20%) with much less computational time.

Finally, a comparison on running time on the two approaches shows that time taken for Clustered Partitioning procedure is much more than our community updating procedure and thus, may prevents this method to complete in a timely manner. In particular, our approach takes only 3 s for obtaining the basic community structure and at most 30 s to complete all the tasks whereas [30] requires more than 5 minutes performing Clustered Partitioning to divide the communication network into modules and selecting the vertex separators. In that delay, the worm propagation may spread out to a larger population and thus, the solution may not be effective. These above experimental results confirm the robustness and efficiency of our approach on social networks.

11.7 Detection of Overlapping Communities

In this section, we describe *DOCA*, a connection-based algorithm to quickly and efficiently discover the overlapping network community structure. In a big picture, *DOCA* works toward the classification of network nodes into different groups by first detecting all possible densely connected parts of the network, and then combining those who share a significant substructure, i.e. those who highly overlap with each other, hence revealing the overlapped network community structure.

There are two important points we want to stress here. Firstly, *DOCA* requires β, the overlapping threshold on how much substructure two communities can share, as an input parameter and that is all it needs. Secondly, *DOCA* fundamentally differs from the one recently suggested in [31] in the way it allows $|C_i \cap C_j| \geq 2$ for any subset C_i, C_j of V, and consequently allows network communities to overlap not only at a single vertex but also as a substructure of the whole community.

11.7.1 Density Function

In order to quantify the goodness of an identified community, we use the popular density function Ψ defined as follow: $\Psi(C) = \frac{|C^{\text{in}}|}{\binom{|C|}{2}}$ [32] where C is a subset of V. The more C approaches a clique of its size, the higher its density value $\Psi(C)$. In order to set up a threshold on the number of connections that suffices for a set of nodes C to be a community, we propose a function $\tau(C)$ defined as follows:

$$\tau(C) = \frac{\sigma(C)}{\binom{|C|}{2}} \text{ where } \sigma(C) = \binom{|C|}{2}^{1-\frac{1}{|C|}}.$$

A subgraph induced by C is a local community iff $\Psi(C) \geq \tau(C)$ or equivalently, $|C^{\text{in}}| \geq \sigma(C)$. Several functions with the same purpose have been introduced in the literature, for instance, in the work of [33, 34] and it is worth noting down the main differences between these functions and ours. First and foremost, our function processes on the candidate group only and does not require any user-input parameter or predefined threshold. Secondly, by Proposition 11.1, $\sigma(C)$ is an increasing function and closely approaches C's full connections, i.e. the number of edges in a clique of size $|C|$. That makes $\sigma(C)$ and $\tau(C)$ relaxation versions of the traditional density function, yet powerful ones as we shall show in the experiments.

Proposition 11.1. *$f(n) = n^{1-\frac{1}{n}}$ is strictly increasing for $n \geq 3$ and $\lim_{n\to\infty} f(n) = n$.*

11.7.2 Objective Function

Our objective is to find a community assignment for the set of nodes V which maximizes the overall internal density function $\Psi(\mathscr{C})$ since the higher the internal density of a community is, the clearer its structure would be. Unlike the case of disjoint community structure, in which the number of connections crossing communities should be less than those inside them, our objective does not take into account the number of out-going links from each community. To understand the reason, let's consider a simple example pictured in Fig. 11.11, in which the concept of a community is violated in both weak and strong senses. In the overlapping community structure point of view, it is clear that every clique in this figure should form a community of its own, and each community overlaps the central clique at exactly one node. However, in the disjoint community structure point of view, any vertex at the central clique has n internal and $2n$ external connections, which violates the concept of a community in the strong sense.

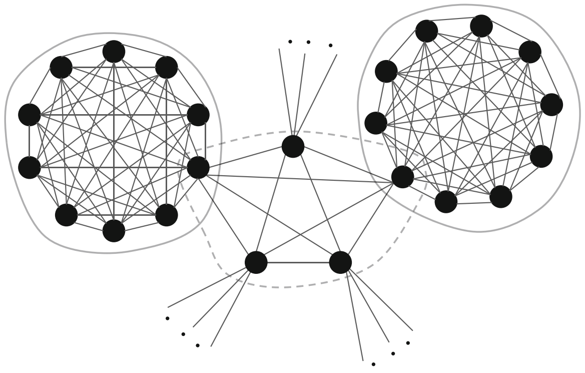


Fig. 11.11 Overlapped vs. non-overlapped community structure. The central clique violates the general concept of community in both strong and weak senses

Furthermore, the internal connectivity of the central clique is also dominated by its external density, which implies the concept of a community in weak sense is also violated.

11.7.3 Locating Local Communities

Local communities are connected parts of the network whose internal densities are greater than a certain level. In our *DOCA* algorithm, this level is automatically determined based on the the size of each corresponding part and the function $\tau(\cdot)$. In particular, a local community is defined based on a connection (u,v) when the number of internal connections within the subgraph induced by $C \equiv N(u) \cap N(v)$ is greater than $\sigma(C)$, or in other words, when C's internal density is greater than $\tau(C)$ (Fig. 11.12a). However, one problem may eventually arise during the detection of these local communities: the containment of sub-communities in an actual bigger one. Intuitively, one would like to detect a bigger community unified by smaller ones if the bigger community is itself densely connected. In order to filter this unfortunate situation, we therefore impose

$$\Psi\left(\bigcup_{i=1}^{s} C_i\right) < \tau\left(\bigcup_{i=1}^{s} C_i\right) \quad \forall s \in [1, |\mathscr{C}|].$$

In addition, we allow this locating procedure to skip over tiny communities of size less than four. This condition is carried out from the Proposition 11.1. It makes sense in terms of mobile or social networks where a group of mobile devices or a social community usually has size larger than three, and intuitively agrees with the finding of [35, 36]. Those tiny communities will then be identified later in *DOCA*.

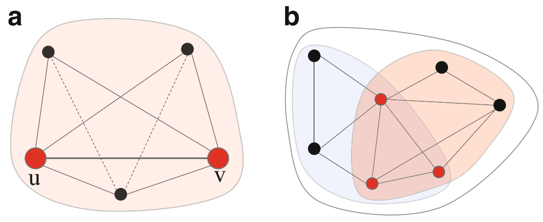


Fig. 11.12 (**a**) A local community C defined by a link $\{u,v\}$. Here, $\Psi(C) = 0.9 > \tau(C) = 0.725$ (**b**) Combining two local communities sharing a significant substructure. Here, $\beta = 0.8$ and OS score is $3/5 + 3/7 = 1.027$

Algorithm 7 Locating local communities

Input: $G = (V, E)$
Output: A collection of raw communities $\mathscr{C}_r$.

```
1: for $(u,v) \in E$ do
2:     if $Com(u) \cap Com(v) = \emptyset$ then
3:         Let $C = N(u) \cap N(v)$;
4:         if $|C^{\text{in}}| \geq \sigma(C)$ and $|C| \geq 4$ then
5:             Define $C$ a local community;
6:             $\mathscr{C}_r = \mathscr{C}_r \cup \{C\}$;
7:         end if
8:     end if
9: end for
```

Lemma 11.6. *The time complexity of Algorithm 7 is $O(M)$.*

Proof. Each time an edge (u,v) is examined, we have to find the intersection of $N(u)$ and $N(v)$, which result in time complexity of $|N(u)| + |N(v)| = \mathrm{d}_u + \mathrm{d}_v$. Moreover, when u and v are in the same community, $\{u,v\}$ will not be taken in consideration. Therefore, the total time complexity is $\sum_{u \in V} \mathrm{d}_u = 2M$.

Lemma 11.7. *Algorithm 7 detects all raw communities C's of size $|C| \geq 4$ and $\Psi(C) \geq \tau(4) \approx 0.83$.*

Proof. By its greedy nature, Algorithm 7 will examine every edge $(u,v) \in E$ (except for those who already found to be in the same community) and will detect any local community C having $|C| \geq 4$ and $\Psi(C) \geq \tau(C) \geq \tau(4) \approx 0.83$. Since each edge is visited at least once, Algorithm 7 will make sure each local community is visited at least once, and thus, the conclusion follows.

Theorem 11.5. *The local community structure $\mathscr{C}_r$ detected by Algorithm 7 satisfies $\Psi(\mathscr{C}_r) \geq 0.83 \times \Psi(OPT)$ where OPT is the optimal community assignment that maximizes the overall internal density function.*

Proof. This Theorem follows from Lemma 11.7 and the condition that no real local community is a substructure of another local community. This also implies that Algorithm 7 is an 0.83-approximation algorithm for finding local densely connected communities.

Algorithm 8 Combining local communities

```
Input: Raw community structure $\mathscr{C}_r$
Output: A refined community structure $\mathscr{C}_f$.
1: $\mathscr{C}_f \leftarrow \mathscr{C}_r$;
2: for $C_i$ and $C_j$ in $\mathscr{C}_r$ and $!Done$ do
3:     if $OS(C_i, C_j) > \beta$ then
4:         $C \leftarrow$ Combine $C_i$ and $C_j$;
5:         /*Update the current structure*/
6:         $\mathscr{C}_f = (\mathscr{C}_f \backslash C_i \backslash C_j) \cup C$;
7:         $Done \leftarrow False$;
8:     end if
9: end for
```

11.7.4 Combining Overlapping Communities

As soon as Algorithm 7 finishes, the raw network community structure is now pictured as a collection of (possibly overlapped) dense parts of the network together with outliers. As some of those dense parts can possibly share significant common substructures, we need to combine them if they are really highly overlapped. In order to do so, we introduce the *overlapping score* of two communities defined as follows:

$$OS(C_i, C_j) = \frac{|I_{ij}|}{\min\{|C_i|, |C_j|\}} + \frac{|I_{ij}^{\text{in}}|}{\min\{|C_i^{\text{in}}|, |C_j^{\text{in}}|\}}.$$

where $I_{ij} = C_i \cap C_j$. Basically, $OS(C_i, C_j)$ values the importance of the common nodes and connections shared between C_i and C_j to the smaller community. In comparison with the duplicate distance metric suggested in [37], our overlapping function not only takes into account the fraction of common nodes but also values the fraction of common connections, which is crucial in order to combine network communities. Furthermore, $OS(\cdot, \cdot)$ is symmetric, so it scales well with the size of any community, and the higher the overlapping score is, the more those communities in consideration should be merged. In this paper, we combine communities C_i and C_j if $OS(C_i, C_j) \geq \beta$ (Fig. 11.12b). We, again, emphasize that β is the only parameter required for *DOCA*.

The time complexity of Algorithm 8 is $O(N_0^2)$ where N_0 is the number of local communities detected in Algorithm 7. Clearly, $N_0 \leq M$ and thus, it can be $O(M^2)$. However, when the intersection of two communities is upper bounded, by Lemma 11.8, we know that the number of local communities is also upper bounded by $O(N)$, and thus, the time complexity of Algorithm 8 is $O(N^2)$. In our experiments, we observe that the running time of this procedure is indeed $O(N^2)$.

Lemma 11.8. *The number of raw communities detected in Algorithm 7 is $O(N)$ when the number of nodes in the intersection of any two communities is upper bounded by a constant α.*

Proof. For each $C_i \in \mathscr{C}$, decompose it into overlapped and non-overlapped parts, denoted by C_i^{ov} and C_i^{nov}. We have $C_i = C_i^{\text{ov}} \cup C_i^{\text{nov}}$ and $C_i^{\text{ov}} \cap C_i^{\text{nov}} = \emptyset$. Therefore, $|C_i| = |C_i^{\text{ov}}| + |C_i^{\text{nov}}|$.

Now,

$$\sum_{C_i \in \mathscr{C}} |C_i| = \sum_{C_i \in \mathscr{C}} (|C_i^{\text{ov}}| + |C_i^{\text{nov}}|) \leq N + \sum_{i<j} |C_i^{\text{ov}} \cap C_j^{\text{nov}}|,$$

where $N = \sum_{C_i \in \mathscr{C}} |C_i^{\text{nov}}| + \left| \bigcup_{C_i \in \mathscr{C}} |C_i^{\text{ov}}| \right|$. For an upper bound of the second term, rewrite

$$\sum_{i<j} |C_i^{\text{ov}} \cap C_j^{\text{nov}}| \leq N + \sum_{|C_i \cap C_j| \geq 2} |C_i \cap C_j| \leq N(1+\alpha),$$

where $\alpha = \max\{|C_i \cap C_j| : |C_i \cap C_j| \geq 2\}$

Hence, $\sum_{C_i \in \mathscr{C}} |C_i| \leq N(2+\alpha)$. Let N_0 be the number of raw communities, it follows that $N_0 \min\{|C_i|\} \leq \sum_{C_i \in \mathscr{C}} |C_i| \leq (2+\alpha)N$. Since $\min\{|C_i|\} \geq 4$, we have $N_0 \leq \frac{(2+\alpha)}{4} N = O(N)$.

11.7.5 Simulation Results

We compare the performance of *DOCA* against the most popular method *CFinder* [38] and the most effective method *COPRA* [39].

Data Sets: The best approach to evaluate the performance of our proposed method is to validate it on real-world traces with known overlapping community structures. Unfortunately, we often do not know that structures beforehand or such structures cannot be mined from the network topologies. Even though synthesis networks might not reflect all the statistical properties of real networks, they can provide us known ground truth via planted community structure and the ability to vary network parameters such as sizes, densities, community overlapping levels and so on. We use networks generated by the well-known LFR overlapping benchmark [3], the de facto standard for testing overlapping community detection algorithms. Generated networks follow power-law degree distribution and contain embedded overlapping communities of varying sizes that capture the internal characteristics of real-world networks.

Metrics. To measure the similarity between detected communities and the embedded ground truth, we evaluate following metrics.

- The most important metric is a generalization of NMI [33] special-built for overlapping communities. Basically, $NMI(U,V)$ is 1 if structures U and V are identical and 0 if they are totally separated. The metric resembles closely the standard NMI in case the communities are disjoint.
- The number of communities, ignoring singleton communities and unassigned nodes. A good community detection method should produce roughly the same number of communities with the known ground truth.
- The overlapping ratio, i.e. the average number of communities to which a node belongs to.

Set up. The parameters for LFR benchmark are: the number of nodes N, the mixing parameter μ that decides the overall sharpness of the community structure (each node shares a fraction μ of its edges with vertices in other communities), the minimum community size c_{min}, the maximum community size c_{max}, o_m the maximum number of communities to which a vertex belongs, and the overlapping fraction γ measuring the fraction of nodes with memberships in two communities or more.

To fairly compare with *COPRA* and to avoid being bias, we keep the parameters close to the setting in [39]: the minimum community size is $c_{min} = 10$, the maximum community size is $c_{max} = 50$, each vertex belongs to at most two communities, $o_m = 2$. The number of vertices are $N = 1{,}000$ or $N = 5{,}000$ and the mixing rate is selected between $\mu = 0.1$ and $\mu = 0.3$.

We fix the overlapping threshold in *DOCA* to be 60% (the most desirable results are obtained when this threshold is between 60% and 70%). Since the output of *COPRA* is undeterministic, we run *COPRA* ten times on each instance and select the best result. In addition, we put no time constraint on the *CFinder*. All methods are executed on an Intel(R) Xeon(R) W3540 CPU at 2.93 Ghz.

11.7.6 Overlapping Communities Quality

We show our results in groups of four. For each case we vary the overlapping fraction γ from 0 to 0.5 and analyze the results found by *DOCA*, *CFinder* and *COPRA*. We only present results when corresponding parameters give top performance for *CFinder* and *COPRA*.

Number of communities: First we show in Fig. 11.13a the number of communities found by *DOCA*, *COPRA*, and *CFinder* and compare them with the ground truth. The more we allow communities to overlap in the network, the larger the number of communities. It reveals from this figure that the number of communities found by *DOCA*, marked with squares, is the closest and almost identical to the ground truth in a long run when the overlapping fraction gets higher. There is an exception when $N = 1{,}000, \mu = 0.3$ that we will discuss later.

Normalized Multual Information: NMI is a more accurate metrics to assess the similarity between communities found and the ground truth. We can infer from Fig. 11.13b that *DOCA* achieves the highest performance among all methods with much more stable. A common trend in this test is the performances of all methods degrade (1) when the mixing rate μ increases, i.e. the community structure becomes more ambiguous or (2) when the network's size decreases, keeping the same mixing rate μ. While *DOCA* is not very competitive only when both negative factors happen in the bottom-right char, $N = 1{,}000, \mu = 0.3$, it is the best performer in general.

Overlapping Ratio: Since $o_m = 2$ every vertex in the overlapped regions will belong to exactly two communities. As a consequence, the average number of communities for a vertex is $\gamma + 1$ that explains why the ground truth are associated with straight

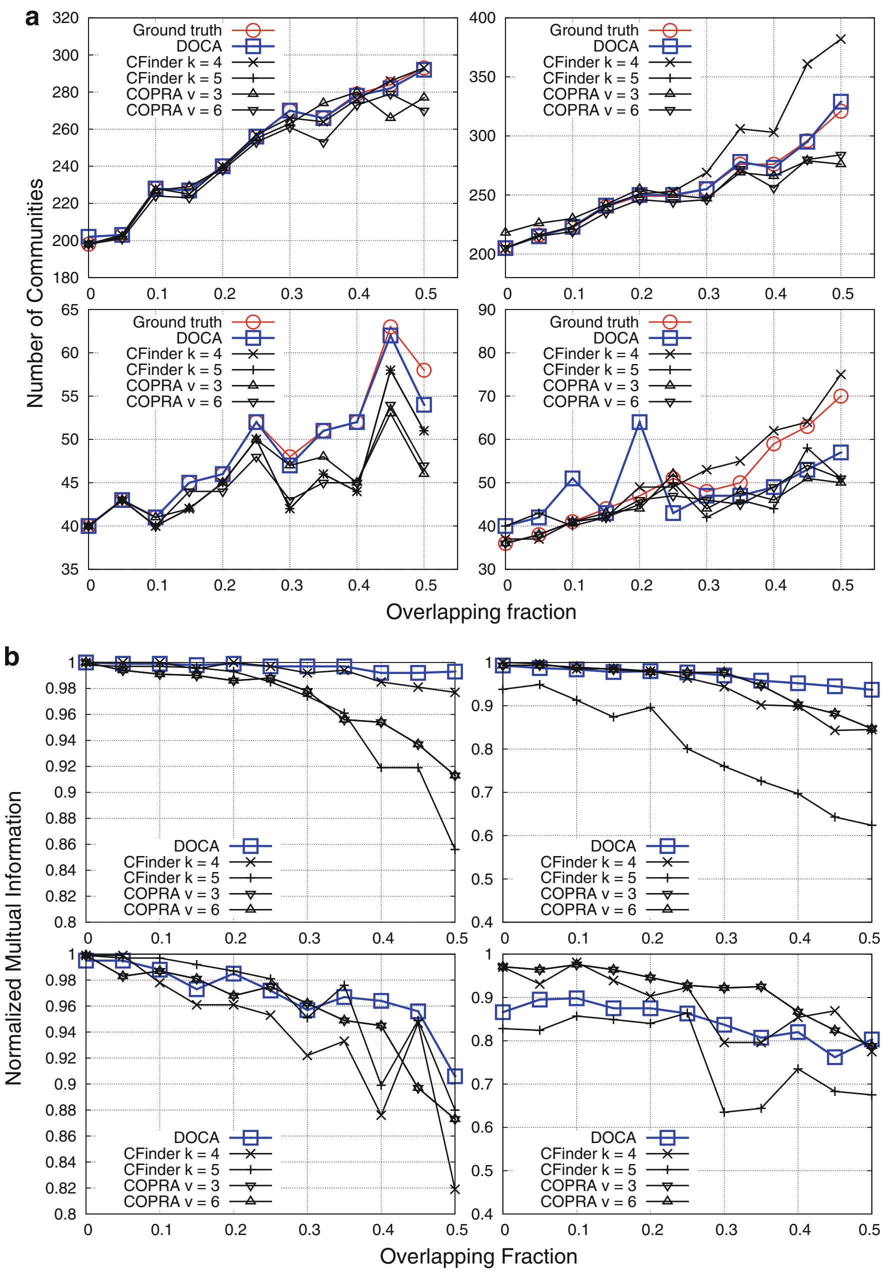


Fig. 11.13 (**a**): Number of communities found by *DOCA*, *COPRA* and *CFinder*. *Top*: $N = 5{,}000$, *Bottom*: $N = 1{,}000$, *Left*: $\mu = 0.1$, *Right*: $\mu = 0.3$. The closer to the ground truth, the better. (**b**): NMI of *DOCA*, *COPRA*, *CFinder*. *Top*: $N = 5{,}000$, *Bottom*: $N = 1{,}000$, *Left*: $\mu = 0.1$, *Right*: $\mu = 0.3$. The higher NMI, the better

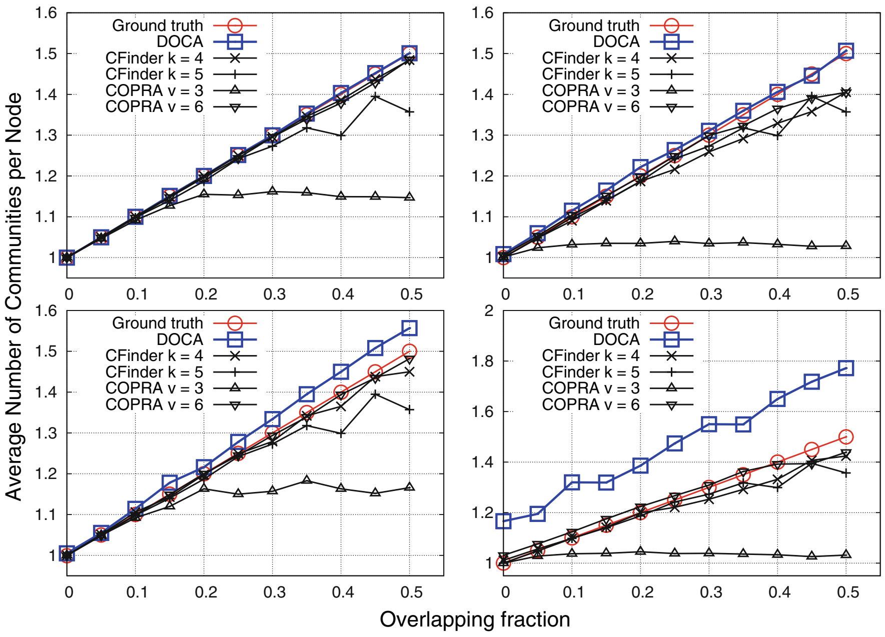


Fig. 11.14 Overlapping fraction, the average number of communities a node belongs to. *Top*: $N = 1{,}000$, *Bottom*: $N = 5{,}000$, *Left*: $\mu = 0.1$, *Right*: $\mu = 0.3$. The closer to the ground truth, the better

line in Fig. 11.14. When $N = 5{,}000$, *DOCA* shares the overlapping rate with the ground truth, but for $N = 1{,}000$, *COPRA* $v = 6$ is much closer to the intended overlapping rates.

The significant gap is observed when the mixing rate gets higher ($\mu = 0.3$) and the network size gets smaller ($N = 1{,}000$). *DOCA* provides less number of communities than the ground truth's but with much higher overlapping rate. The reason is that with a larger mixing rate μ a node will have more edges connecting to vertices in other communities, thus will increase the chance that *DOCA* merges highly overlapped communities. Hence, *DOCA* creates less but with larger size communities. We note that this "weakness" of *DOCA* is controversial as when the mixing rate increases, the ground truth does not necessarily coincide with the structure implied by the network's topology.

Extensive experiments show *DOCA* to give high quality overlapping communities. Moreover, we found *DOCA* run substantially faster than the others when the network contains thousands of nodes due to its small constant factor with average running time less than 10 ms for 5,000 nodes networks while its competitors take seconds. Thus, the method can be implemented effectively applications on complex networks, especially on social networks. In the next section, we illustrate the efficiency of utilizing overlapping communities concept in problem of worm containment on Online Social Networks.

11.8 Related Work

Community detection on static networks has attracted a lot of attentions and many efficient methods have been proposed for this type of networks. The readers are strongly encouraged to read the excellent survey [32] for an overview. Detecting community structure on dynamic networks, however, has so far been an untrodden area. In [40], the authors defined time graphs that captured the link creation as a point phenomena in time of a directed evolving graph. Based on that, the authors studied the evolution of the blogosphere in terms of changes such as in-degree, out-degree, etc. Another work [41] studied the growth of the a wide range of real-world evolving graphs and provided a new kind of graph generator that, based on a forest fire spreading process, produced networks with the discovered patterns. In another work [42], the authors suggested a method for observing the evolution of web communities by first revealing network communities at each specific time point and then, quantifying changes that occurred to network communities based on different types of community changes such as emerging, growing and shrinking.

One of the most seminal work [2] proposed an innovative method for detecting communities on dynamic networks the based on *k*-clique percolation technique. With the proposed method, they analyzed a co-authorship network and a mobile phone network and revealed some interesting characteristics on the number of communities, community sizes, ages, and their correlation as well as autocorrelation. This approach can detect overlapping nodes in different network communities; however, its internal *k*-clique percolation technique may require high computing resources and thus, may be time consuming especially on large scale social networks.

A work in [11] presented GraphScope, a parameter-free methodology for detecting clusters on time-evolving graphs based on mutual information and entropy functions of Information Theory. This method is notable due to its parameter-free property, however, it requires a recomputation of the number of sources and destinations each time the graph segments change (i.e. when users joining in or withdrawing from the network, or when new social connections are introduced or removed) without utilizing its previously computed information. Thus, it might not lend itself effectively to the field of adaptive algorithms. [43] proposed a distributed method for community detection in which modularity was used as a measure instead of objective function. A part from that, [44] attempted to track the evolving of communities over time, using a few static network snapshots.

A recent work of [45] proposed a detection method based on contradicting the network topology and the topology-based *propinquity*, where *propinquity* is the probability of a pair of nodes involved in a community. Another recent attempt to analyze communities and their evolutions in dynamic social networks, which is closely related to our work, includes [46] in which the authors proposed *FacetNet*, a framework to track community evolutions in a unified process. In this framework, the community structure at a given timestep is determined both by the observed the network data and the prior distribution given by historic community structures. A possible limit of this framework is that at each timestep, the underlying algorithm should be executed for multiple values of *m*-the number of communities, which might prevent this framework from being effective when applied to real-world social network traces.

In [47], the authors present a framework for identifying dynamic communities with a constant factor approximation. This is a very nice property, however, this method also requires some predefined costs to penalize people moving in or out of a community, which might be generally unknown in dynamic social networks. A recent work [14] proposes a social-aware routing strategy in MANETs which also makes uses of a modularity-based procedure name MIEN for quickly updating the network structure. In particular, MIEN tries to compose and decompose network modules in order to keep up with the changes and uses fast modularity algorithm [4] to update the network modules. However, this method might be time consuming due to the high complexity of [4].

Palla *et al* proposed CFinder [38], a popular seminal method based on clique-percolation technique which iteratively searches for communities composed of connected *k*-cliques, starting from an initial clique of size *k*. However, due to the sparseness of real networks the communities discovered by CFinder are usually of low quality, as we shall see in the experiments. Gregory recently proposed COPRA [39], a label propagation method with an extended feature to allow multiple community memberships. Recent benchmarks on community detection methods [3][48] reveal that with appropriate parameters set up, COPRA is the best method for detecting overlapping network communities. Other detection trends includes methods based on nodes splitting [49] , modularity [50][51] and link-based methods [31][52].

11.9 Conclusion

In this chapter, we presented *QCA*, an adaptive algorithm for detecting and tracing community structures in dynamic social networks where changes are introduced frequently. We also present *DOCA*, a quick and efficient method for detecting overlapping communities in a complex network. We show that our adaptive algorithms are not only effective in updating and identifying high quality network community structure but also has the great advantage of fast running time, which is suitable for large and rapidly changing online social networks. In addition, we prove some theoretical results, which are the basic observations of our approach. Finally, via two practical applications in MANETs routing strategies and worm containment on social networks, we show that our QCA algorithm promises enormous realistic applications not only on mobile computing but also on online social networks as it can be combined or integrated into many community detection modules.

References

1. M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99, 2002.
2. G. Palla, P. Pollner, A. Barabasi, and T. Vicsek. Social group dynamics in networks. *Adaptive Networks*, 2009.
3. A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical review. E. 80*, 2009.
4. M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E 69*, 2003.
5. V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory and Experiment*, 2008.
6. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phy. Rev. E 69*, 2004.
7. U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2), 2008.
8. A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E 70*, Aug 2004.
9. S. Fortunato and M. Barthelemy. Resolution limit in community detection. *PNAS*, 104, 2007.
10. Z. Ye, S. Hu, and J. Yu. Adaptive clustering algorithm for community detection in complex networks. *Physical Review E*, 78, 2008.
11. S. Jimeng, C. Faloutsos, S. Papadimitriou, and Philip S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD*, 2007.
12. ArXiv dataset. http://www.cs.cornell.edu/projects/kddcup/datasets.html. *KDD Cup 2003*, Feb 2003.
13. B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *2nd ACM SIGCOMM Workshop on Social Networks*, 2009.
14. T. N. Dinh, Y. Xuan, and M. T. Thai. Towards social-aware routing in dynamic communication networks. *IPCCC*, 2009.
15. P. Hui and J. Crowcroft. How small labels create big improvements. *PERCOMW*, 2007.
16. E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc '07*, 2007.
17. A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6, 2007.
18. P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *MobiHoc '08*, 2008.
19. E. Nathan and A. Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4), 2006.
20. Koobface. http://www.pcworld.com/article/155017/\facebook_virus_turns_your_computer_into_a_zombie.html. *PC World*, 2008.
21. Koobface. http://news.cnet.com/koobface-virus-hits-facebook/. *CNET*, 2008.
22. V. Sekar, Y. Xie, M. K. Reiter, and H. Zhang. A multi-resolution approach forworm detection and containment. In *DSN '06*, 2006.
23. N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, 2004.
24. R. Dantu, J. W. Cangussu, and S. Patwardhan. Fast worm containment using feedback control. *IEEE Trans. Dependable Secur. Comput.*, 4(2), 2007.
25. H. Kim and B. Karp. Autograph: toward automated, distributed worm signature detection. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, 2004.
26. N. James, B. Karp, and S. Dawn. Polygraph: Automatically generating signatures for polymorphic worms. In *IEEE Symposium on Security and Privacy*, pages 226–241, 2005.
27. P. Wang, M. C. González, C. A. Hidalgo, and A. Barabasi. Understanding the spreading patterns of mobile phone viruses. *Science*, 324, 2009.

28. A. Bose and K. G. Shin. Proactive security for mobile messaging networks. In *WiSe '06: Proceedings of the 5th ACM workshop on Wireless security*, 2006.
29. A. Bose, X. Hu, K.G. Shin, and T. Park. Behavioral detection of malware on mobile handsets. In *MobiSys '08*, New York, NY, USA, 2008.
30. Z. Zou, G. Cao, S. Zhu, S. Ranjan, and A. Nucci. A social network based patching scheme for worm containment in cellulat networks. In *IEEE INFOCOM*, 2009.
31. J. P. Bagrow Y.Y. Ahn and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.
32. S. Fortunato and C. Castellano. Community structure in graphs. *eprint arXiv: 0712.2716*, 2007.
33. A. Lancichinetti, S. Fortunato, and K. Jnos. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.
34. A. Lzr, D. bel, and T. Vicsek. Modularity measure of networks with overlapping communities. *(Europhysics Letters)*, 90(1), 2010.
35. S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010.
36. J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW08*, pages 695–704. ACM, 2008.
37. C. Lee, F. Reid, A. McDaid, and N. Hurley. Detecting highly overlapping community structure by greedy clique expansion. *KDD*, 2010.
38. G. Palla, I. Derenyi, I. Farkas1, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(10), 2005.
39. Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018, 2010.
40. R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, 2003.
41. J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*, 2005.
42. M. Toyoda and M. Kitsuregawa. Extracting evolution of web communities from a series of web archives. In *HYPERTEXT '03*, 2003.
43. P. Hui, E. Yoneki, S. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *Proc. MobiArch*, 2007.
44. J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Tracking evolving communities in large linked networks. *PNAS*, 101, 2004.
45. Y. Zhang, J. Wang, Y. Wang, and L. Zhou. Parallel community detection on large networks with propinquity dynamics. In *KDD '09*. ACM, 2009.
46. Y-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. Analyzing communities and their evolutions in dynamic social networks. *ACM Trans. Knowl. Discov. Data*, 3, 2009.
47. T. Chayant and B. Tanya. Constant-factor approximation algorithms for identifying dynamic communities. In *KDD '09*, 2009.
48. L. Peel. Estimating network parameters for selecting community detection algorithms. *FUSION*, 2010.
49. S. Gregory. An algorithm to find overlapping community structure in networks. In *PKDD 2007*, pages 91–102. Springer, 2007.
50. V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. *J. Stat. Mech.: Theory and Experiment*, 2009(03):P03024, 2009.
51. H-W. Shen, X-Q. Cheng, and J-F. Guo. Quantifying and identifying the overlapping community structure in networks. *J. Stat. Mech. Theory and Experiment*, 2009(07):P07042+, 2009.
52. T. S. Evans. and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Phys. Rev. E*, 80(1):016105, 2009.

Chapter 12
Path Formation in Human Contact Networks

Nishanth Sastry and Pan Hui

Abstract The Pocket Switched Network (PSN) is a radical proposal to take advantage of short-range connectivity afforded by human face-to-face contacts, and create longer paths by having intermediate nodes ferry data on behalf of the sender. The Pocket Switched Network creates paths over time using transient social contacts. This chapter explores the achievable connectivity properties of this dynamically changing mileu, and gives a community-based heuristic to find efficient routes.

We first employ empirical traces to examine the effect of the human contact process on data delivery. Contacts between a few node pairs are found to occur too frequently, leading to inadequate mixing of data, while the majority of contacts occur rarely, but are essential for global connectivity. We then examine all successful paths found by flooding and show that though delivery times vary widely, randomly sampling a small number of paths between each source and destination is sufficient to yield a delivery time distribution close to that of flooding over all paths. Thus, despite the apparent fragility implied by the reliance on rare edges, the rate at which the network can deliver data is remarkably robust to path failures.

We then give a natural heuristic that finds routes by exploiting the latent social structure. Previous methods relied on building and updating routing tables to cope with dynamic network conditions. This has been shown to be cost ineffective due to the partial capture of transient network behavior. A more promising approach would be to capture the intrinsic characteristics of such networks and utilize them for

N. Sastry
Kings College London, Strand, London, WC2R 2LS
e-mail: firstname.lastname@kcl.ac.uk

P. Hui (✉)
Deutsche Telekom Laboratories, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
e-mail: pan.hui@telekom.de

routing decsions. We design and evaluate BUBBLE, a novel social-based forwarding algorithm, that utilizes the *centrality* and *community* metrics to enhance delivery performance. We empirically show that BUBBLE can efficiently identify good paths using several real mobility datasets.

12.1 Introduction

Consider a scenario in which Alice wants to convey some information to Carol. If Bob happens to meet Alice first and then Carol, he could potentially serve as a messenger for Alice's message. Essentially, this method of communication exploits human contacts to create a path *over time* between a source and destination. Of necessity, the data paths are constructed in a store-carry-forward fashion: Various intermediate nodes *store* the data on behalf of the sender and *carry* it to another contact opportunity where they *forward* the data to the destination or another node that can take the data closer to the destination.

As normally practised, transferring information over social contacts requires manual intervention (e.g. Alice requesting Bob, "When you see Carol, please tell her that…") as well as a knowledge of future contacts (The human actors need to know that Bob will be meeting Carol in the future). Manual intervention can easily be avoided by automatically exchanging data between mobile devices carried by the human actors. Widely supported short-range data-transfer protocols such as bluetooth or Wi-Fi can be used for this purpose. If we do not have knowledge of future contacts, data can still be forwarded *opportunistically* from node to node, but without a guarantee that it will reach the intended destination.

This idea, of leveraging human social contacts, and using ubiquitious mobile devices in people's pockets to opportunistically connect a source and destination over time, has been termed as a Pocket Switched Network (PSN) [16]. As a store-carry-forward network, the PSN can incur long and highly variable delays. On the other hand, it has the advantage of not requiring infrastructure setup or maintenance. It is therefore useful when infrastructure is damaged (e.g. after disasters), or does not exist (e.g. in remote areas). Also, mobility increases network capacity at the expense of delays, providing multi-user diversity gains [15]. Thus, this method can be effective as a multi-hop "sneakernet" for high-bandwidth applications that can tolerate delays.

The question remains as to how "well" the transient, local contacts can support a wider connectivity across the network, if we do not have knowledge of future contacts. The first part of the chapter explores this issue by studying two traces that recorded human contacts over extended time periods. We measure the achievable performance of the contact network over a given time window in terms of the fraction of data delivered (delivery ratio), as well as the time to delivery. The delivery ratio at the end of a time window is indicative of the fraction of node pairs connected during the window and is therefore a measure of the connectivity achieved by the network. The empirically observed cumulative distribution of delivery times can also be interpreted as the evolution in time of delivery ratio, normalised by the ratio eventually achieved at the end of the time window,[1] and thus represents the rate at which connectivity is achieved.

We find that rare contacts are crucial for connectivity: when contacts between node-pairs which meet infrequently are removed, the network breaks apart into smaller, unconnected components. Note that in order for any path to succeed, all the contacts in the path need to occur successfully, and in the right temporal order. Thus, if one of the rare contacts involved in a path does not happen or succeed in transferring data, the path will fail. Thus, this result seems to imply a fragility in the network of social contacts.

Note that failures of individual paths does not automatically imply failure to achieve connectivity. There can be several paths between a source and destination, and if at least one path succeeds, connectivity is achieved. However, it might take longer to reach the destination if some of the quicker paths fail. We study the degradation of delivery times by examining the impact of random path failures among paths found by flooding data. Specifically, we look at two modes of path failures. The first, *proportional flooding*, assumes that a fixed fraction $\mu < 1$ of the paths between every source–destination pair succeeds. The second, *k-copy forwarding*, allows at most k paths to succeed between each sender and destination. In both cases, we find that the time to achieve connectivity is remarkably resilient in that the distribution of delivery times with a large number of path failures remains close to the delivery times achieved when there are no path failures.

In the second part of the chapter, we address the question of how to route data from a given sender to a destination and give a social-based heuristic for finding "good" routes. Many MANET and some DTN routing algorithms [1, 23] provide forwarding by building and updating routing tables whenever mobility occurs. This approach is not appropriate for a PSN, since mobility is often unpredictable, and topology structure is highly dynamic. We need an algorithm which can cope with dynamic, repeated disconnection and re-wiring. Rather than exchange much control traffic to create unreliable routing structures, which may only capture the "noise" of the network, we prefer to search for some characteristics of the network which are less volatile than mobility. A PSN is formed by people. Hence, social metrics are intrinsic properties to guide data forwarding in such kinds of human networks.

In this context, we introduce an social-based forwarding algorithm, BUBBLE, which focuses on two key social metrics: community and centrality. Co-operation binds, but also divides human society into communities. For an ecological community, the idea of correlated interaction means that an organism of a given type is more likely to interact with another organism of the same type than with a randomly chosen member of the population [34]. This correlated interaction concept also applies to humans, so we can exploit this kind of community information to select forwarding paths. Within a community, some people are more popular, and interact

[1]If the empirical probability that the delivery time is less than t is r, then a fraction r of the data that eventually get delivered have been delivered by time t.

with more people than others (i.e., have high *centrality*); we call them hubs. In this chapter, we will show how community and centrality help us to efficiently identify the "good" forwarding paths.

The rest of this chapter will be presented in two parts: In Part 12.2, we will study the feasibility of path formation in human contact networks. In Part 12.3, we will discuss a community-based routing strategy called BUBBLE, and evaluate its performance. Section 17.2 discusses related work and Sect. 12.5 concludes.

12.2 Feasibility of Path Formation

Our first goal is to study the extent to which the temporally changing network of human face-to-face contacts can support $N \times N$ connectivity. Since our goal is to explore feasibility in a future PSN, we employ traces drawn from "naturalistic" settings, lasting at least a month long. The contacts in these traces are highly heterogeneous, with some nodes meeting each other hundreds of times and others meeting fewer than ten times.

Our main results are as follows: Nodes which meet each other frequently turn out to be inefficient for data transfer. Intuitively, nodes which meet each other too often do not have new data to exchange during their second and subsequent contacts. In contrast, the rarely occurring contacts are more effective at "mixing" data and are crucial for reachability. Since the rare contacts do not, by definition, recur often, paths which rely on them are sensitive to chance events which either prevent the contact from occurring or prevents the data from being transferred during the contact. However, we show that connectivity of the network as a whole is not greatly affected by individual path failures. Specifically, we study two different modes of path failures and show that the distribution of delivery times remains close to optimal despite a large number of paths failing.

12.2.1 Setup and Methodology

This section motivates the choice of the traces used in the first part of the chapter, the simulation setup and the performance measures used to study feasibility of path formation.

12.2.1.1 Traces

We imagine the participants of a PSN would be a finite group of people who are at least loosely bound together by some context – for instance, first responders at a disaster situation, who need to send data to each other. Multiple PSNs could co-exist for different contexts, and a single individual could conceivably participate in several different PSNs.[2]

Our model that PSN participants form a cohesive group places the requirement that an ideal PSN should be able to create paths between arbitrary source-destination pairs. This is reflected in our simulation setup, where the destinations for each source node are chosen randomly. Also, our traces are picked to be close to the limits of Dunbar's number ($=147.8$, 95% confidence limits: 100.2–231.1), the average size for cohesive groups of humans [7].

The first trace comes from a four week subset of the UCSD Wireless Topology Discovery [42] project which recorded Wi-Fi Access Points seen by subjects' PDAs. We treat PDAs simultaneously in range of the same Wi-Fi access point as a contact opportunity. This dataset traces contacts between $N = 202$ subjects. The second trace consists of bluetooth contacts recorded from 1 Nov. 2004 to 1 Jan. 2005 between participants of the MIT Reality Mining project [9]. We conservatively set five minutes as the minimum allowed data transfer opportunity and discarded contacts of durations smaller than this cutoff. This trace has contacts between $N = 91$ subjects.

The subjects in the MIT trace consist of a mixture of students and faculty at the MIT Media Lab, and incoming freshmen at the MIT Sloan Business School. The UCSD trace is comprised of a select group of freshmen, all from UCSD's Sixth College. As such, we can expect subjects in both traces to have reasons for some amount of interaction, leading to a loosely cohesive group structure. Prior work on community mining using the same traces supports this expectation [44].

It is important to emphasize that our focus is solely on the capability and efficiency of the human contact process in forming end-to-end paths. The precise choice of the minimum data transfer opportunity is less important – it is entirely possible that a new technology would allow for faster node-node transfers. Indeed, our results are qualitatively similar for other cutoff values tested. Similarly, a different technology for local node-node transfers could have different "reach," allowing more nodes to be in contact with each other simultaneously. Nevertheless, the substantial similarities (see rest of this section) between results based on two different technologies and traces – the Wi-Fi based UCSD trace and the bluetooth based MIT trace – gives us some confidence that the results below may be applicable beyond the traces and technologies we have considered.

Both our traces were chosen to be at least one month long, in order to obtain multiple disjoint time windows over which to test the relevance of our results.

[2]Note that this is in contrast to a single unboundedly large network of socially unrelated individuals as in the famous "small-world" experiment [41] that examined a network essentially comprising all Americans and discovered an average 5.2 (≈ 6) degrees of separation.

12.2.1.2 Simulation Setup and Measurement

Setup: At the beginning of simulation, data is created, marked for a randomly chosen destination, and associated with the source node. An oracle with complete knowledge of the future can choose to transfer data at appropriate contact opportunities and thereby form the quickest path to the destination. To simulate this, we enumerate all possible paths found by flooding data at each contact opportunity, and choose the quickest.

Performance measure: Consider the time-ordered sequence (with ties broken arbitrarily) of contacts that occur globally in the network. Since there are $N(N-1)$ quickest paths between different source–destination pairs, a maximum[3] of $N(N-1)$ contacts in the the global sequence of contacts act as path completion points. Of these, Nd become "interesting" when there are d destinations per sender. Since the destinations are chosen randomly, we might expect that on average, if k path completion points have occured, the *fraction* of these that are interesting is independent of d: When d is greater, more data gets delivered after k path completion points, but there is also more data to deliver.

The above discussion motivates our method of measuring the efficiency of the PSN: At any point in the simulation, the *delivery ratio*, measured as the fraction of data that has been delivered, or equivalently, the number of "interesting" path completion points we have seen, is taken as a figure of merit. The more efficient the PSN is, the faster the delivery ratio evolves to 1, as the number of contacts and time increase.

Unless otherwise specified, our experiments examine delivery ratio evolution statistically averaged over 10 independent runs, with each run starting at a random point in the trace, and lasting for 6,000 contacts. We confirm our intuition in Fig. 12.1, which shows that the delivery ratio evolves similarly, whether d is 1 or a maximum of $N-1$ destinations per sender. We note that the graph also represents the fastest possible evolution of the delivery ratio under the given set of contacts, due to the use of flooding.

12.2.2 Order and Distribution of Contacts

A PSN contact trace is determined by the distribution of contact occurrences and the time order in which these contacts occur. In this section, we examine how these properties affect delivery ratio evolution.

Given two traces, the more efficient one will manage to achieve a given delivery ratio with fewer number of contacts. Our approach is to create a synthetic trace from

[3]The actual number could be lesser because a contact with a rarely active node could complete multiple paths that end in that node.

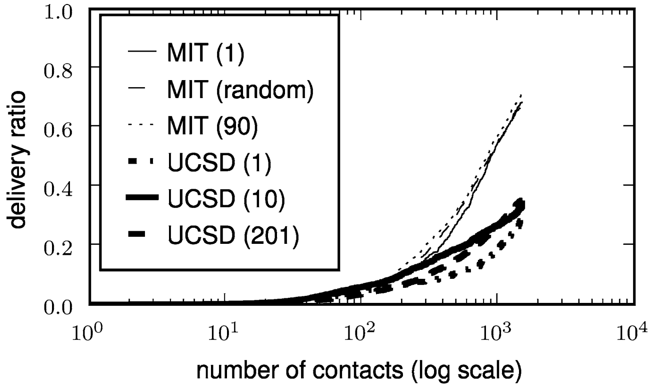


Fig. 12.1 Fraction of data delivered as a function of the number of contacts, for the MIT and UCSD traces (number of destinations per sender shown in brackets). The curves for each network are clustered together, showing that the delivery ratio evolves independently of the load

the original trace by disrupting the property we wish to study. Comparing delivery ratio evolution in the original and synthetic traces informs us about the effects of the property.

Our main findings are that in both the traces we examine, time correlations between contacts that occur too frequently leads to non-effective contacts in which no new data can be exchanged, and that the progress of the delivery ratio as well as the connectivity of the PSN itself are precariously dependent on rare contacts.

12.2.2.1 Frequent Contacts are Often Non-effective

To investigate the effect of the time order in which contacts occur, we replay the trace, randomly shuffling the time order in which links occur. Observe in Fig. 12.2 that the curve marked "shuffled" evolves faster than "trace" implying that the delivery ratio increases faster after random shuffling. The random shuffle has the effect of removing any time correlations of contacts in the original trace. Thus, the improved delivery ratio evolution implies that time correlations of the contacts in the original data slowed down the exchange of data among the nodes, causing them to be delivered later.

Manual examination reveals several time correlated contacts where two nodes see each other multiple times without seeing other nodes. At their first contact, one or both nodes could have data that the other does not, which is then shared by flooding. After this initial flooding, both nodes contain the same data – subsequent contacts are "non-effective," and only increase the number of contacts happening in the network without increasing the delivery ratio.

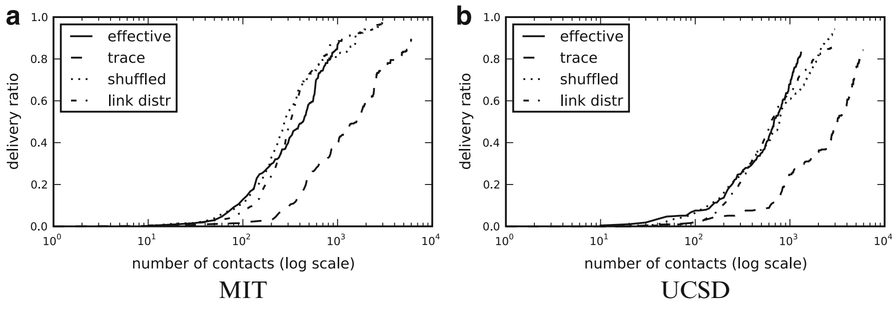


Fig. 12.2 Delivery ratio evolution for synthetically derived variants of MIT and UCSD traces. "Trace" is the original. "Shuffled," the same trace with time order of contacts randomly shuffled. "Effective" replays "trace," counting only contacts where data was exchanged. "Link distr" is an artificial trace with the same size and contact occurrence distribution as the original

To quantify the impact, in the curve marked "effective" on Fig. 12.2, we plot delivery ratio evolution in the original trace, counting only the contacts in which data could be exchanged. This coincides well with the time-shuffled trace, showing that non-effective contacts are largely responsible for the slower delivery ratio evolution in the original trace.

Next, we construct a synthetic trace that has the same number of nodes as the original trace, as well as the same contact occurrence distribution. By this, we mean that the probability of contact between any pair of nodes is the same as in the original trace. The delivery ratio evolution of this trace, depicted as "link distr" in Fig. 12.2, is seen to evolve in a similar fashion as the time-shuffled trace. This indicates that once time correlations are removed, the delivery properties are determined mainly by the contact occurrence distribution.

12.2.2.2 Connectivity Depends on Rare Contacts

The fact that three different traces (shuffled, effective, and link distr), which are based on the same contact occurrence distribution, essentially evolve in the same manner leads us to examine this distribution further.

Figure 12.3 shows that the contact occurrence distribution has both highly rare contacts (involving node pairs that meet fewer than ten times in the trace) as well as frequent contacts (nodes which meet hundreds of times). A randomly chosen contact from the trace is much more likely to be a rare contact than a frequent one.

Figure 12.4 shows that the rare contacts are extremely important for the nodes to stay connected. When contacts that occur fewer than a minimum cutoff number of times are removed, the number of nodes remaining in the trace falls sharply. This implies that there are a number of nodes which are connected to the rest of the nodes by only a few rare contacts.

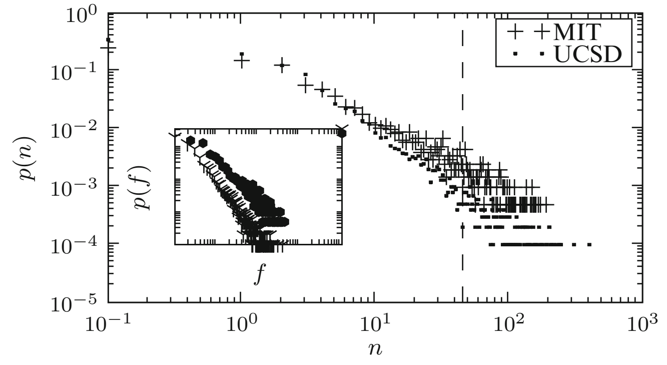


Fig. 12.3 Contact occurrence distributions (log–log): A random edge appears n times with probability $p(n)$. To the left of the dashed line at $n = 45$, the distributions for both traces coincidentally happen to be similar. The inset shows the difference when normalised by the number of contacts in the trace. In the inset, a random edge constitutes a fraction f of the trace with probability $p(f)$

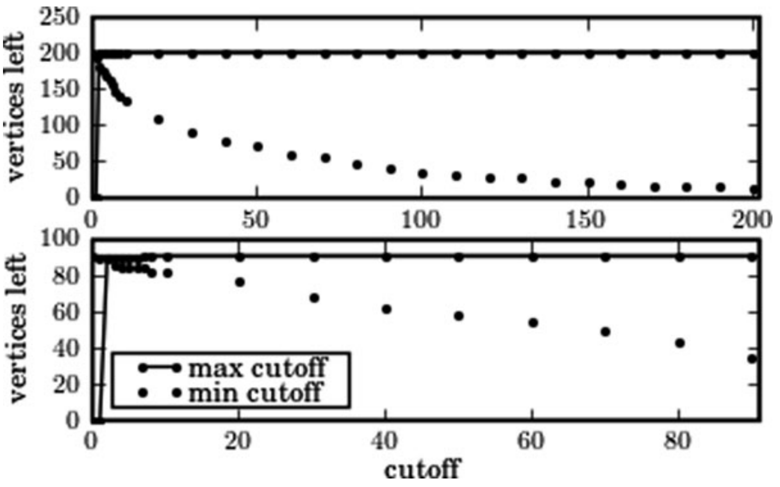


Fig. 12.4 Robustness to cutoff: MIT (below), UCSD (above). Max cutoff specifies a maximum cutoff for the frequency of contacts, thus removing the most frequently occurring ones. Min cutoff specifies a minimum frequency of contacts – removing the rarest contacts causes the number of nodes that are connected to drop precipitously

On the other hand, removing the frequent contacts (by removing contacts occurring more than a maximum cutoff number of times) does not affect connectivity greatly. For instance, the MIT trace remains connected even when the maximum cutoff is as low as 10 (i.e., contacts occurring more than ten times are removed). This suggests that nodes which contact each other very frequently are also connected by other paths, comprising only rare edges.

12.2.3 Resilience to Path Failures

In order for a path to succeed in a temporally changing network like the PSN, all edges have to occur in the right order. Therefore, the reliance on rare edges, as shown in the previous section, could lead to a large number of path failures. Note that this does not automatically imply bad connectivity. Since only one path between every source–destination pair needs to succeed for data delivery, individual path failures do not greatly impact the delivery ratio achieved at the end of a time window (unless all paths between a source–destination pair fail, disconnecting the network).

However, path failures can affect the rate at which the delivery ratio evolves: Suppose the quickest path between a pair of nodes would have arrived at t_1, but cannot be used because of a failure. If the first usable path connects the nodes at time $t_2 > t_1$, then between t_2 and t_1 the fraction of data delivered is decreased on account of the path failure. In other words, there is a delay in data delivery, which temporarily shifts the cumulative distribution of delivery times to the right.

This section looks at the effects of path failures by studying the effects of failures on paths found by flooding. Given a sequence of contacts, flooding achieves the best possible delivery times by exploring *every* contact opportunity and thereby finding the path with the *minimum* path delay.

We look at instances of the PSN over fixed time-windows and wish to study the degradation in the delivery time distribution when not all of the paths found by flooding can be explored. Specifically, we study two failure modes. The first, proportional flooding, explores a fixed fraction μ of the paths found by flooding between each source and destination. We show that a constant increase in the fraction of paths explored brings the delivery time distribution of proportional flooding exponentially closer to that of flooding over all paths. The second failure mode, k-copy flooding, explores no more than a fixed number $k > 1$ of the paths found by flooding between each source and destination. Again, a constant increase in k brings the delivery time distribution exponentially close to the optimal delivery time distribution of flooding all paths. Empirically, even small values of k (e.g. $k = 2$ or $k = 5$) closely approximate delivery times found by flooding.

The results of this section imply that the human contact network is remarkably resilient to path failures and the delivery ratio evolves at a close-to-optimal rate even when the majority of paths fail and only a small fraction or a small, bounded number of paths can transport data to the destination. Note that we only admit paths from the original flood-tree, and do not include new paths that repair failures by joining the affected nodes to the flood tree at later contacts. Thus, our results in fact underestimate the resilience of the network.

The success of k-copy flooding can provide a loose motivation for routing algorithms that use multiple paths between each sender and destination pair since this could obtain a close-to-optimal delivery time distribution. However, heuristics-based routing algorithms may not find the same paths as found by flooding. Thus, the correspondence is not exact.

Table 12.1 Summary of notation used to characterise components of path delay and delivery times

H	Hop delay, or time to next hop. Time until a path expands by one more node.
N	Number of edges per path. $N \sim Poisson(mean = \lambda)$
L	Number of paths between a random src-dest pair.
D	Path delay for a random path
D^*	Delivery time (minimum path delay across all paths between a random source–destination pair)
$G_X(s)$	Probability-generating function of X
$M_X(s)$	Moment-generating function of X. $M_X(s) = G_X(e^X)$

12.2.3.1 Path Delay Distribution D

[14, 39, 45] model the performance of epidemic routing and its variants and derive a closed form for delivery time distribution, showing it to be accurate for certain common mobility models. However, several simplifying assumptions are made, including an exponential inter-contact time between node pairs. Unforunately, human contact networks are known to have power law inter-contact times with exponential tails [4, 24]. Furthermore, [14, 39, 45] use a constant (averaged) contact rate, whereas the contact rates in our empirical traces are highly heterogeneous (see Sect. 12.2.2). Plugging in the mean contact rate from our empirical traces into their expressions yields bad fits.

Thus, in order to obtain a handle on delays incurred on paths, we take a very coarse grained and simplified approach. In particular, we only assume that path delays (D) can be treated as being independent of each other, that the distribution of time to next hop (H) can be described by a moment-generating function $M_{\mathrm{H}}(s)$, and that the number of hops (N) on the paths found by flooding follows a Poisson distribution with mean λ. See Table 12.1 for a full summary of our notation.

Our most specialized assumption is that the number of hops on a path formed by flooding during a fixed time-window follows the Poisson distribution. This is justified by a surprisingly good fit in our empirical traces (Fig. 12.5). We conjecture that this is a result of several factors which work together to limit the number of hops in a successful path. First, we only consider paths that form during a fixed time window. Second, the small-world nature of the human contact graph makes for short paths to a destination; and paths are frozen at the destination because the destination does not forward data further. Third, each node can join the flood-tree at most once. As the tree grows, the number of nodes available to grow the tree and extend a path decreases. Thus, extremely long paths are rare.

Using the above assumptions, the path delay D can be written as

$$M_D(s) = G_N(M_{\mathrm{H}}(s)). \tag{12.1}$$

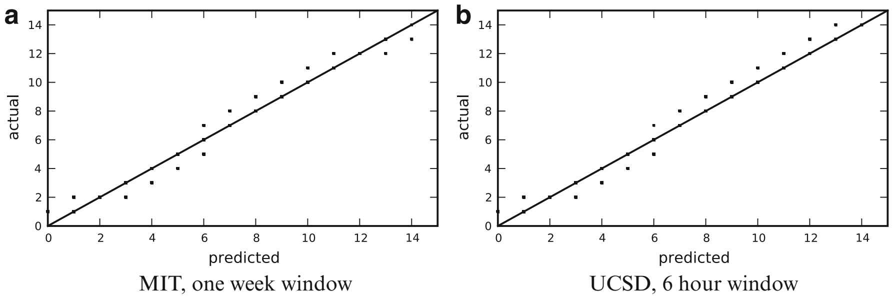


Fig. 12.5 Number of hops follows the Poisson Distribution. Each Q-Q plot shows fit through correspondence between sample deviates generated according to the theoretical distribution (predicted) and empirical (actual) values. Closeness to predicted = actual diagonal indicates better fit. Different combinations of trace and time window sizes are used to show generality of fit

We can apply a Chernoff-type bound and write

$$P[D \geq t] \leq \min_{s>0} \mathrm{e}^{-st} M_D(s) = \min_{s>0} \mathrm{e}^{\lambda(M_{\mathrm{H}}(S))-st} = \exp(F_{\mathrm{H}}(t)), \tag{12.2}$$

where $F_{\mathrm{H}}(t) = \lambda M_{\mathrm{H}}(s_{\min}(t)) - s_{\min}(t)t - \lambda$ and $s_{\min}(t)$ minimises s in the Chernoff bound.

12.2.3.2 Proportional Flooding

Consider an arbitrary source–destination pair. As described previously, we will model the path delays between them as being chosen independently and identically from the distribution in (12.1). Suppose copies of the data are sent along l randomly chosen paths between them. The obtained delivery time D_l^* is the *minimum* of the path delays across all l paths. Using (12.2) we can write

$$P[D_l^* \leq t] = 1 - \prod_{i=1}^{l} P[D \geq t] \geq 1 - \mathrm{e}^{-lF_{\mathrm{H}}(t)}. \tag{12.3}$$

Note that the above assumes that the l path delays are independent. In reality, paths found by flooding all fan out from a single source node, and the first few hops, close to the source, are typically shared with other paths, violating the independence assumption. Therefore, the model in this section is to be considered only as a simple formulation designed to gain insight into proportional flooding. It is worth mentioning however that in the empirical data sets, we frequently find that the major component of path delay is contributed by the part of the paths closest to the destination, which are not shared with other paths. Also, in the case when only a few paths on the flood-tree are being randomly sampled, the number of hops shared is limited.

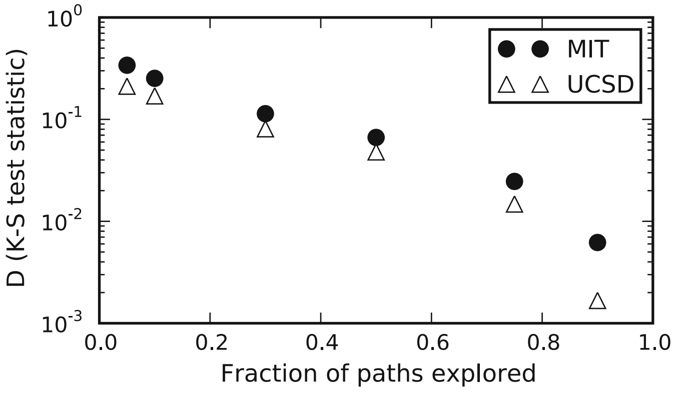


Fig. 12.6 K-S statistic (D) measuring the difference between the delivery time distributions of full flooding and proportional flooding for different μ. X-axis is linear, Y-axis is log-scale.

Consider source–destination pairs with $L = m$ paths connecting them. Full flooding finds the quickest of all m paths and obtains a delivery time distribution $P[D_L^* \leq t|L=m]$. Proportional flooding chooses a fraction μ of them. From (12.3), the difference $\Delta(t;\mu)$ in the delivery time distributions between full and proportional flooding, is upper bounded by

$$\Delta(t;\mu) \leq P[D_L^* \leq t|L=m] - 1 + \mathrm{e}^{-\mu m F_{\mathrm{H}}}(t), \tag{12.4}$$

Remark 1. A constant increase in μ has an exponential effect on Δ: For any t, if μ is increased by some constant, the fraction of data delivered by proportional flooding during $[0,t]$ becomes exponentially closer to that delivered by full flooding. Thus, proportional flooding quickly becomes very effective as μ is increased.

The exponential decrease in Δ with a constant increase in μ is obtained as long as $F_{\mathrm{H}}(t) < 0$. In other words, our results hold when there are a Poisson number of hops in paths formed over fixed time windows, for any hop delay distribution H that has a moment generating function and satisfies $F_{\mathrm{H}}(t) < 0$.

Also, since

$$\frac{\partial \Delta}{\partial \mu} = mF_{\mathrm{H}}(t)\mathrm{e}^{\mu m F_{\mathrm{H}}(t)} < 0,$$

Δ decreases when μ is increased. Furthermore, the rate of decrease is higher for smaller μ – increasing μ from $\mu = 0.1$ to $\mu = 0.2$ results in a greater decrease than an increase from $\mu = 0.6$ to $\mu = 0.7$.

Fig. 12.6 empirically shows the difference between $D^*(t)$, the delivery time distribution obtained by flooding over all paths, and $D_\mu^*(t)$, the delivery time distribution for proportional flooding using a randomly selected fraction μ of paths between every source and destination. The difference is measured using the Kolmogorov–Smirnov statistic given by $D = \max_t (D^*(t) - D_\mu^*(t))$. Note that the Y-axis is log scale; a constant increase in μ shows an exponential decrease in D.

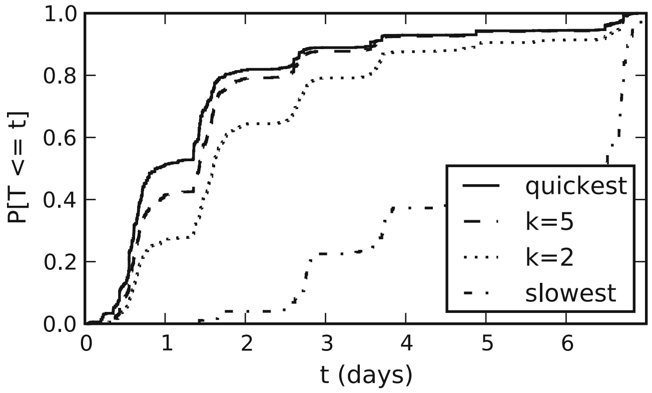


Fig. 12.7 *k*-copy flooding: Nodes are connected by multiple paths with different delays (CDFs of the *quickest* and *slowest* are shown). Yet, randomly choosing at most *k* of the paths to each destination closely approximates the quickest, even for small *k*. (MIT trace, one week window)

12.2.3.3 From Proportional to Bounded Number of Paths

Proportional flooding offers a mechanism to gracefully degrade from full flooding by exploring a fraction of the paths. However, in the worst case, there can be up to $N-1$ paths to a destination in a N node PSN, and proportional flooding can be costly. This leads us to define a bounded cost strategy that explores at most a small, fixed number, k of the paths to a destination, and still achieves delivery times similar to that of proportional flooding. Unlike proportional flooding, k-copy flooding explicitly limits the number of paths explored, and therefore can tolerate a larger number of path failures in the worst case, when there are a large number of paths between a node-pair.

Fig. 12.7 shows empirically that in our data sets, even for small k $(= 2,5)$, the delivery time distribution of k-copy flooding starts to closely approximate full flooding. To see why, consider the *equivalent fraction* μ_k of paths in proportional flooding that gives the same expected number of paths as k-copy forwarding:

$$\sum_{l=0}^{k} lP[L=l] + kP[L>k] = \mu_k \mathbb{E}[L]. \tag{12.5}$$

Suppose k is increased by a constant h, resulting in a new equivalent fraction μ_{k+h}. (12.5) becomes

$$\sum_{l=0}^{k} lP[L=l] + \sum_{j=1}^{h} (k+j)P[L=k+j] + (k+h)P[L>k+h] = \mu_{k+h}\mathbb{E}[L].$$

Regrouping, we get

$$\sum_{l=0}^{k} lP[L=l] + k\left(\sum_{j=1}^{h} P[L=k+j] + P[L>k+h]\right)$$
$$+\sum_{j=1}^{h} jP[L=k+j] + hP[L>k+h] = \mu_{k+h}\mathbb{E}[L].$$

Comparing with (12.5), we can write

$$\mu_k\mathbb{E}[L] + \sum_{j=1}^{h} jP[L=k+j] + hP[L>k+h] = \mu_{k+h}\mathbb{E}[L].$$

Thus, the *increase* in the equivalent fraction of paths is

$$\begin{aligned}\mu_{k+h} - \mu_k &\geq \frac{h}{\mathbb{E}[L]}\left(\sum_{j=1}^{h} P[L=k+j] + P[L>k+h]\right)\\ &= h\left(P[L>k]/\mathbb{E}[L]\right).\end{aligned} \tag{12.6}$$

Remark 2. A constant increase in k is equivalent to at least a (scaled) constant increase in the fraction of paths explored by proportional flooding. Thus, as a simple consequence of Remark 1, a constant increase in the number of paths explored in k-copy forwarding moves its delivery time distribution exponentially closer to that of full flooding.

This explains why exploring at most a small number k of paths has a delivery time distribution approaching that of flooding over all paths. Fig. 12.8 empirically shows the equivalent fractions μ_k for the $k=2$ and $k=5$ cases discussed previously.

12.3 BUBBLE: A Community Based Routing Algorithm

After establishing the feasibility of $N \times N$ connectivity, in this second part of the chapter, we present a concrete mechanism to route data over human contacts. The key insight is to exploit the structure inherent in human social contacts. Bubble leverages the heterogeneity in popularity – certain individuals, such as a postman, are likely to have contacts with many different persons and are therefore useful in bridging disjoint nodes. In BUBBLE , messages "bubble" up and down the social hierarchy in order to reach the destination. Messages traverse the hierarchy, using the highly central nodes to bridge data between disjoint communities where necessary, until they reach the destination.

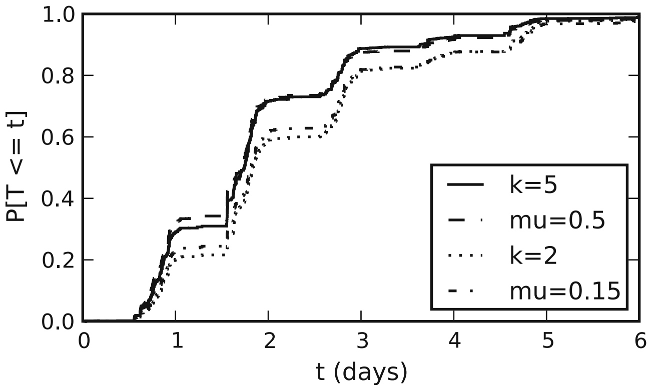


Fig. 12.8 Proportional flooding with $\mu_2 = 0.15$ of paths has similar delivery times as $k = 2$-copy routing. Similarly, $k = 5$ corresponds to $\mu_5 = 0.5$. (MIT, one week window)

12.3.1 Traces

For evaluating BUBBLE, we use four experimental datasets gathered by the Haggle Project[4] over two years, referred to as *Infocom05*, *HongKong*, *Cambridge*, *Infocom06* and one dataset from the MIT Reality Mining Project [8], referred to as *Reality*. Previously, the characteristics of these datasets such as inter-contact and contact distribution have been explored in several studies [3, 25, 28], to which we refer the reader for further background information.

- In *Infocom05*, the devices were distributed to approximately fifty students attending the Infocom student workshop. Participants belong to different social communities (depending on their country of origin, research topic, etc.).
- In *Hong-Kong*, the people carrying the wireless devices were chosen independently in a Hong-Kong bar, to avoid any particular social relationship between them. These people have been invited to come back to the same bar after a week. They are unlikely to see each other during the experiment.
- In *Cambridge*, the iMotes were distributed mainly to two groups of students from University of Cambridge Computer Laboratory, specifically undergraduate year1 and year2 students, and also some PhD and Masters students. This dataset covers 11 days.
- In *Infocom06*, the scenario was very similar to *Infocom05* except that the scale is larger, with 80 participants. Participants were selected so that 34 out of 80 form 4 subgroups by academic affiliations.

[4]http://www.haggleproject.org

Table 12.2 Characteristics of the five experimental data sets

Experimental data set	Infocom05	Hong-Kong	Cambridge	Infocom06	Reality
Device	iMote	iMote	iMote	iMote	Phone
Network type	Bluetooth	Bluetooth	Bluetooth	Bluetooth	Bluetooth
Duration (days)	3	5	11	3	246
Granularity (seconds)	120	120	600	120	300
Number of experimental devices	41	37	54	98	97
Number of internal contacts	22,459	560	10,873	191,336	54,667
Average # Contacts/pair/day	4.6	0.084	0.345	6.7	0.024
Number of external devices	264	868	11,357	14,036	NA
Number of external contacts	1,173	2,507	30,714	63,244	NA

- In *Reality*, 100 smart phones were deployed to students and staff at MIT over a period of 9 months. These phones were running software that logged contacts with other Bluetooth enabled devices by doing Bluetooth device discovery every five minutes.

The five experiments are summarised in Table 12.2. A remark about the datasets is that the experiments do not have the same granularity and the finest granularity is limited to 120 s. This is because of the trade-off between the duration of the experiments and the accuracy of the samplings.

The four Haggle datasets were chosen to allow us greater insight into the actual (ground truth) community structure, whereas the *Reality* dataset is used to demonstrate that BUBBLE is robust to inferred community structure as well.

12.3.2 Inferring Human Communities

In a PSN, the social network could map to the computer network since people carry the computing devices. In this section, we introduce and evaluate two centralised community detection algorithms: *K*-CLIQUE by Palla et al. [35] and weighted network analysis (WNA) by Newman [31]. We use these two centralised algorithms to uncover the community structures in the mobile traces. We believe our evaluation of these algorithms can be useful for future traces gathered by the research community.

Many centralised community detection methods have been proposed and examined in the literature (see the review papers by Newman [32] and Danon et al. [6]). The criteria we use to select a centralised detection method are the ability to uncover overlapping communities, and a high degree of automation (low manual involvement). In real human societies, one person may belong to multiple communities and hence it is important to be able to detect this feature. The *K*-CLIQUE method satisfies this requirement, but was designed for binary graphs, thus we must threshold the edges of the contact graphs in our mobility traces to use this method, and it is difficult to choose an optimum threshold manually [35]. On the other hand, (WNA) can work on weighted graphs directly, and does not need thresholding, but it cannot detect overlapping communities [31]. Thus we chose to use both *K*-CLIQUE and WNA; they each have useful features that complement one another.

12.3.2.1 Contact Graphs

In order to help us to present the mobility traces and make it easier for further processing, we introduce the notion of a *contact graph*. The way we convert human mobility traces into weighted contact graphs is based on the number of contacts and the contact duration, although we could use other metrics. The nodes of the graphs are the physical nodes from the traces, the edges are the contacts, and the weights of the edges are the values based on the metrics specified such as the number of contacts during the experiment. We can measure the relationship between two people by how many times they meet each other and how long they stay with each other. We naturally think that if two people spend more time together or see each other more often, they are in a closer relationship.

First, we find the distribution of contact durations and number of contacts for the two conference scenarios are quite similar. To prevent redundancy, in the later sections we only selectively show one example, in most cases *Infocom06*, since it contains more participants.

Figure 12.9 and Figure 12.10 show the contact duration and number of contacts distribution for each pair in four experiments. For the *HongKong* experiment we include the external device because of the network sparseness, but for the other three experiments we use only the internal devices. These contact graphs created are used for the community detection in the following subsections.

12.3.2.2 K-CLIQUE Community Detection

Palla et al. [35] define a *k*-clique community as a union of all *k*-cliques (complete subgraphs of size *k*) that can be reached from each other through a series of adjacent *k*-cliques, where two *k*-cliques are said to be adjacent if they share $k-1$ nodes. As *k* is increased, the *k*-clique communities shrink, but on the other hand become more cohesive since their member nodes have to be part of at least one *k*-clique. We have applied this on all the datasets above. Figure 12.11 shows the 3-clique communities in the *Infocom06* dataset. More detailed descriptions about the *k*-clique communities on these datasets can be found in our previous work [18, 19].

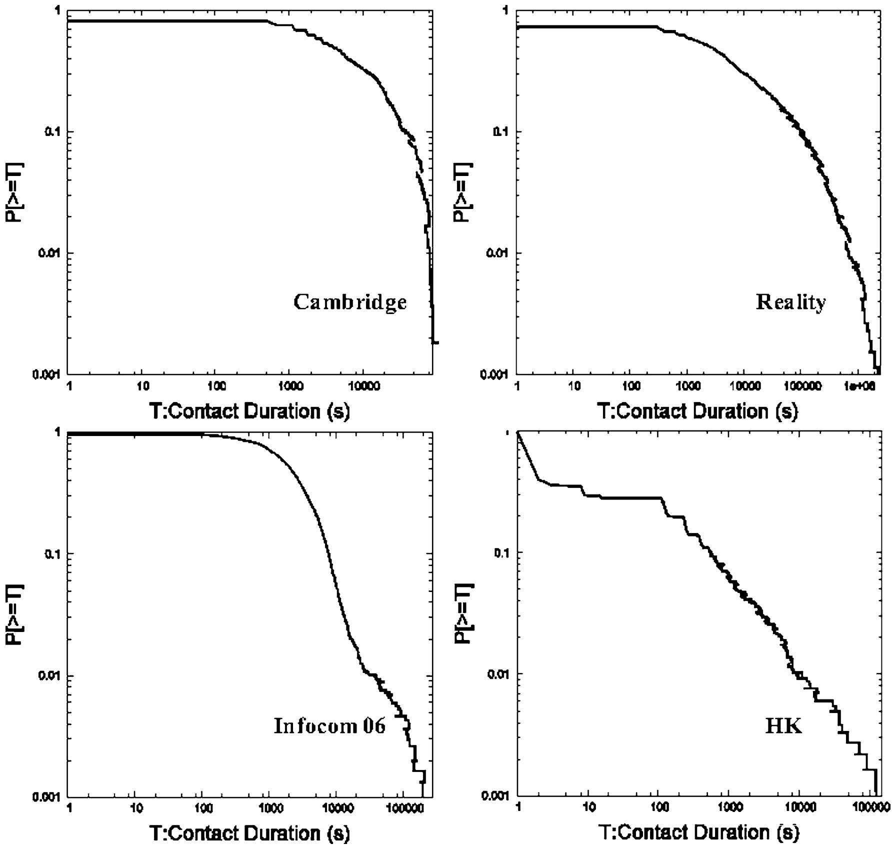


Fig. 12.9 The distribution of pair-wise contact durations

12.3.2.3 Weighted Network Analysis

In this section, we implement and apply Newman's WNA for our data analysis [31]. This is an extension of the unweighted *modularity* method proposed in [33] to a weighted version. We use this as a measurement of the fitness of the communities it detects.

For each community partitioning of a network, one can compute the corresponding modularity value using the following definition of *modularity* (Q):

$$Q = \sum_{vw} \left[\frac{A_{vw}}{2m} - \frac{k_v k_w}{(2m)^2} \right] \delta(c_v, c_w), \tag{12.7}$$

where A_{vw} is the value of the weight of the edge between vertices v and w, if such an edge exists, and 0 otherwise; the δ-function $\delta(i, j)$ is 1 if $i = j$ and 0 otherwise;

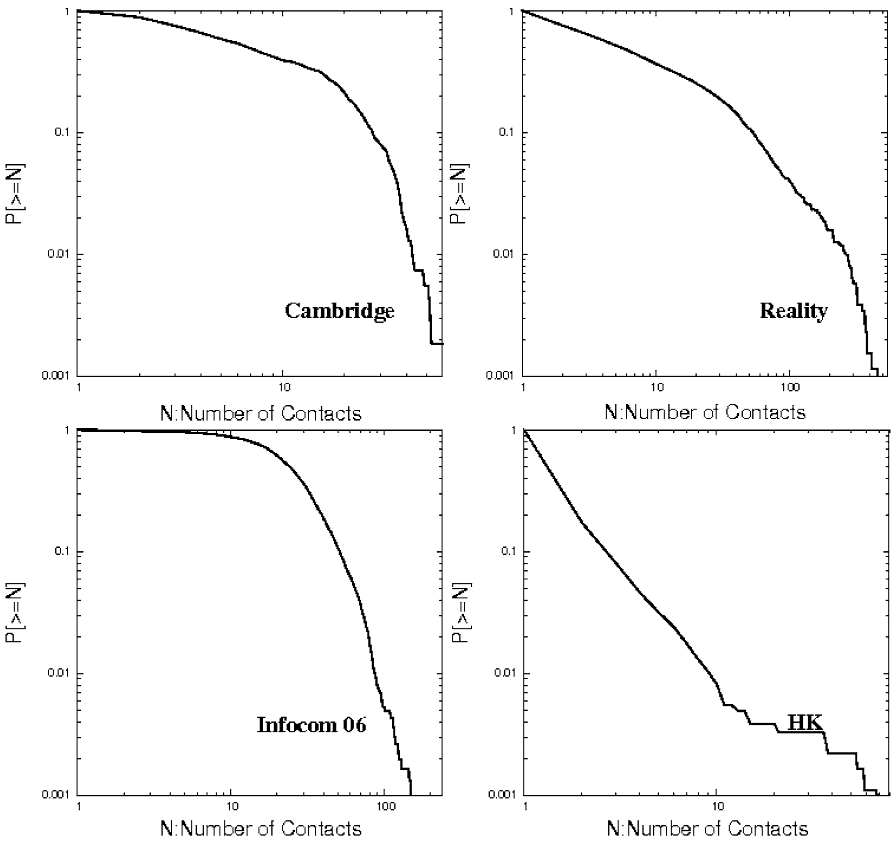


Fig. 12.10 The distribution of pair-wise number of contacts

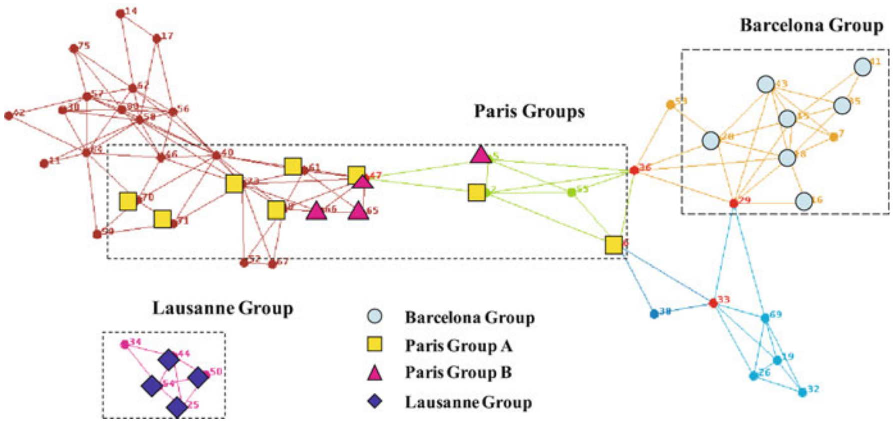


Fig. 12.11 3-clique communities based on contact durations with weight threshold that equals 20,000 s (Infocom06; circles, Barcelona group; squares, Paris group A; triangles, Paris group B; diamonds, Lausanne group)

Table 12.3 Communities detected from the four datasets

Dataset	Info06	Camb	Reality	HK
$Q_{\max}$	0.2280	0.4227	0.5682	0.6439
Max. community size	13	18	23	139
No. communities	4	2	8	19
Avg. community size	8.000	16.500	9.875	45.684
No. community nodes	32	33	73	868
Total no. of nodes	78	36	97	868

$m = \frac{1}{2}\sum_{vw} A_{vw}$; k_v is the degree of vertex v defined as $\sum_w A_{vw}$; and c_i denotes the community vertex i belongs to. *Modularity* is defined as the difference between this fraction and, the fraction of the edges that would be expected to fall within the communities if the edges were assigned randomly, but we keep the degrees of the vertices unchanged. The algorithm is essentially a genetic algorithm, using the modularity as the measurement of fitness. Rather than selecting and mutating current best solutions, we enumerate all possible merges of any two communities in the current solution, and evaluate the relative fitness of the resulting merges, and choose the best solution as the seed for the next iteration.

Table 12.3 summarises the communities detected by applying WNA on the four datasets. According to Newman [31], non-zero Q values indicate deviations from randomness; values around 0.3 or more usually indicate good divisions. For the *Infocom06* case, the $Q_{\max}$ value is low; this indicates that the community partition is not very good in this case. This also agrees with the fact that in a conference the community boundary becomes blurred. For the *Reality* case, the Q value is high; this reflects the more diverse campus environment. For the *Cambridge* data, the two groups spawned by WNA exactly match the two groups (1st year and 2nd year) of students selected for the experiment.

These centralised community detection algorithms give us rich information about the human social clustering and are useful for offline data analysis on mobility traces collected. We can use them to explore structures in the data and hence design useful forwarding strategies, security measures, and killer applications.

12.3.3 Heterogeneity in Centrality

In human society, people have different levels of popularity: salesmen and politicians meet customers frequently, whereas computer scientists may only meet a few of their colleagues once a year [18]. Here, we want to employ heterogeneity in popularity to help design more efficient forwarding strategies: we prefer to choose popular hubs as relays rather than unpopular ones.

A temporal network or time-evolving network is a kind of weighted network. The centrality measure in traditional weighted networks may not work here since the edges are not necessarily concurrent (i.e., the network is dynamic and edges are time-dependent). Hence, we need a different way to calculate the centrality of each node in the system. Our approach is as follows:

1. Carry out a large number of emulations of unlimited flooding with different uniformly distributed traffic patterns created.
2. Count the number of times a node acts as a relay for other nodes on all the shortest delay deliveries. Here, the shortest delay delivery refers to the case when the same message is delivered to the destination through different paths, where we only count the delivery with the shortest delay.

We call the number calculated above the *betweenness centrality* of this node in this temporal graph. Of course, it can be normalised to the highest value found. Here we use unlimited flooding since it can explore the largest range of delivery alternatives with the shortest delay. This definition captures the spirit of Freeman centrality [13].

For the emulation, we developed an emulator called *HaggleSim* [17], which can replay the collected mobility traces and emulate different forwarding strategies on every contact event. This emulator is driven by contact events. The original trace files are divided into discrete sequential contact events, and fed into the emulator as inputs. In all the simulations for the BUBBLE algorithm (including the evaltuions in Sect. 12.3.4), we divided the traces into discrete contact events with granularity of 100 s Our emulator reads the file line by line, treating each line as a discrete encounter event, and makes a forwarding decision on this encounter based on the forwarding algorithm under study.

Figure 12.12 shows the number of times a node falls on the shortest paths between all other node pairs. We can treat this simply as the centrality of a node in the system. We observe very wide heterogeneity in each experiment. This clearly shows that there is a small number of nodes which have extremely high relaying ability, and a large number of nodes that have moderate or low centrality values, across all experiments. One interesting point from the HK data is that the node showing highest delivery power in the figure is actually an external node. This node could be some popular hub for the whole city, i.e., a postman or a newspaper man in a popular underground station, who relayed a certain amount of cross city traffic. The 30th, 70th percentiles, and the means of normalised individual node centrality are shown in Table 12.4. These numbers summarise the statistical property of the centrality values for each system shown in Fig. 12.12.

12.3.4 Social-Based Routing

The contribution of this section is to combine the knowledge of both centralities of nodes and community structure, to achieve further performance improvements in forwarding. We show that this avoids the occurrence of the dead-ends encountered

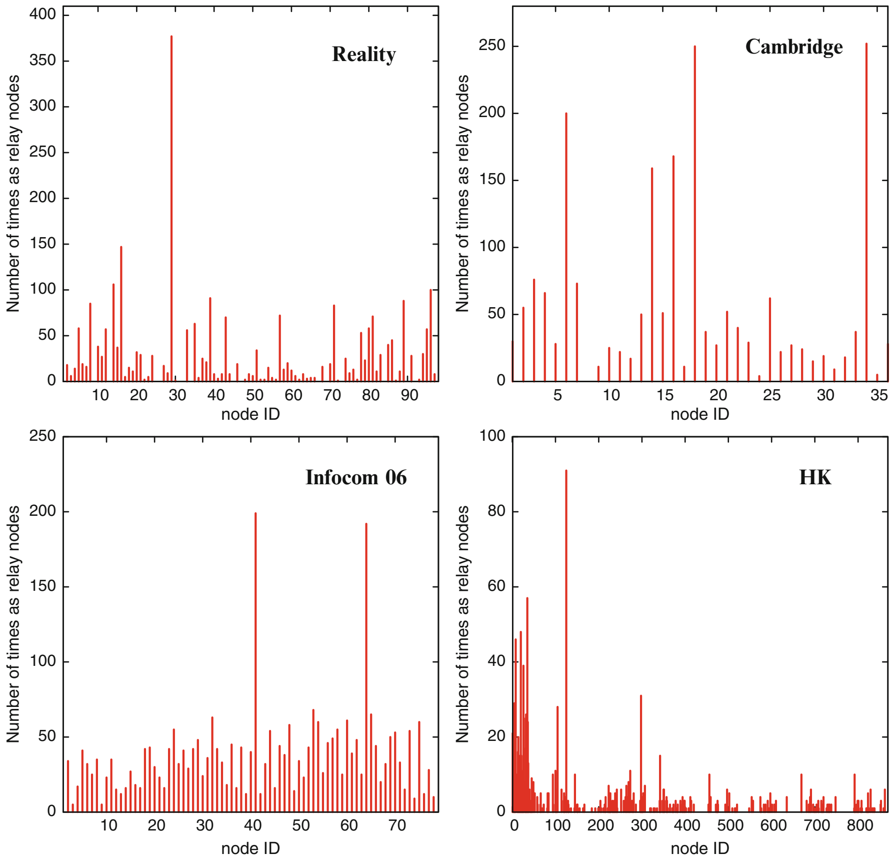


Fig. 12.12 Number of times a node as relays for others on four datasets

Table 12.4 Statistics about normalised node centrality in 4 experiments

Experimental dataset	30th percentile	Mean	70th percentile
Cambridge	0.052	0.220	0.194
Reality	0.005	0.070	0.050
Infocom06	0.121	0.188	0.221
Hong Kong	0.000	0.017	0.000

with pure global ranking schemes. We call the protocols here BUBBLE, to capture our intuition about the social structure. Messages bubble up and down the social hierarchy, based on the observed community structure and node centrality, together with explicit label data. Bubbles represent a hybrid of social and physically observable heterogeneity of mobility over time and over community.

12.3.4.1 Overview of Forwarding Algorithms

In order to compare and evaluate the efficiency of the forwarding algorithms in finding the good paths for the destination. Forwarding algorithms can be divided into two broad categories: those that are aware of the social structure, and those oblivious to social structure. BUBBLE exploits the latent social structure. We evaluate its performance in relation to the following naïve strategies which attempt to forward data without using any social structure:

- WAIT: Hold onto a message until the sender encounters the recipient directly, which represents the lower bound for delivery cost. WAIT is the only single-copy algorithm in this chapter.
- FLOOD: Messages are flooded throughout the entire system, which represents the upper bound for delivery and cost.
- Multiple-Copy-multiple-hoP (MCP): Multiple copies are sent subject to a time-to-live hop count limit on the propagation of messages. By exhaustive emulations, the 4-copy-4-hop MCP scheme was found to be the most cost-effective scheme in terms of delivery ratio and cost for all naive schemes among most of the datasets.

In contrast to the above, we explore four different algorithms which leverage various different aspects of social structure:

- LABEL: Explicit labels are used to identify forwarding nodes that belong to the same organisation. Optimisations are examined by comparing label of the potential relay nodes and the label of the destination node.This is in the human dimension, although an analogous version can be done by labelling a *k*-clique community in the physical domain.
- RANK: The forwarding metric used in this algorithm is the node centrality. A message is forwarded to nodes with higher centrality values than the current node. It is based on observations in the network plane, although it also reflects the hub popularity in the human dimension.
- DEGREE: The forwarding metric used in this algorithm is the node degree, more specifically the observed average of the degree of a node over a certain time interval. Either the last interval window (S-Window), or a long-term cumulative estimate, (C-Window) is used to provide a fully decentralised approximation for each node's centrality, and then that is used to select forwarding nodes.
- BUBBLE: The BUBBLE family of protocols combines the observed hierarchy of centrality of nodes and observed community structure with explicit labels, to decide on the best forwarding nodes. BUBBLE is an example algorithm which uses information from both human aspects and also the physically observable aspects of mobility.

BUBBLE is a combination of LABEL and RANK. It uses RANK to spread out the messages and uses LABEL to identify the destination community. For this algorithm, we make two assumptions:

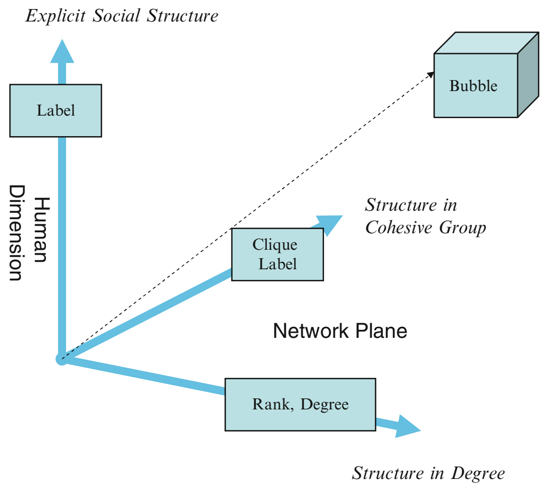


Fig. 12.13 Design space for forwarding algorithms

- Each node belongs to at least one community. Here, we allow single node communities to exist.
- Each node has a global ranking (i.e., global centrality) in the whole system and also a local ranking within its community. It may belong to multiple communities and hence may have multiple local rankings.

Figure 12.13 shows the design space for the social-based forwarding algorithms. The vertical axis represents the explicit social structure. This is the social or human dimension. The two horizontal axes represent the network structural plane, which can be inferred purely from observed contact patterns. The Structure-in-Cohesive Group axis indicates the use of localised cohesive structure, and the Structure-in-Degree axis indicates the use of node ranking and degree. These are observable physical characteristics. In our design framework, it is not necessary that physical dimensions are orthogonal to the social dimension, but since they represent two different design parameters, we would like to separate them. The design philosophy here is to consider both the social and physical aspects of mobility.

12.3.4.2 Two-Community Case

In order to make the study more systematic, we start with the two-community case. We use the *Cambridge* dataset for this study. By experimental design, and as confirmed using our community detection algorithm, we can clearly divide the *Cambridge* data into two communities: the undergraduate year-one and year-two group. In order to make the experiment more fair, we limit ourselves to just the two 10-clique groups found with a number-of-contact threshold of 9; that is where each node at least meet another 9 nodes frequently. Some students may skip lectures and cause variations in the results, so this limitation makes our analysis yet more plausible.

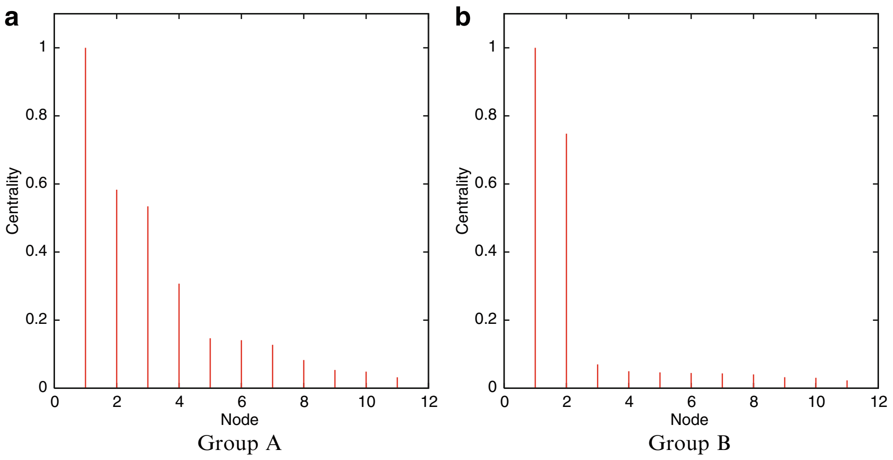


Fig. 12.14 Node centrality in 2 groups in *Cambridge* data, see Sect. 12.3.3 for the method of calculating the centrality values. (**a**) Group A (**b**) Group B

First we look at the simplest case, for the centrality of nodes within each group. In this case, the traffic is created only for members within the same community and only members in the same community are chosen as relays for messages. We can clearly see from Fig. 12.14a and 12.14b that inside a community, the centrality of each node is different. In Group B, there are two nodes which are very popular, and have relayed most of the traffic. All the other nodes have low centrality value. Forwarding messages to the popular nodes would make delivery more cost effective for messages within the same community.

Then we consider traffic which is created within each group and only destined for members in another group. To eliminate other outside factors, we use only members from these two groups as relays. Figure 12.15 shows the individual node centrality when traffic is created from one group to another and the correlation of node centrality within an individual group and inter-group (for data deliveries only to other groups but not to its only group) centrality. We can see that points lie more or less around the diagonal line. This means that the inter- and intra- group centralities are quite well correlated. Active nodes in a group are also active nodes for inter-group communication. There are some points on the left hand side of the graph which have low intra-group centrality but moderate inter-group centrality. These are nodes which move across groups. They are not important for intra-group communication but will be useful when we need to move traffic from one group to another.

Figure 12.16 shows the correlation of the local centrality of Group A and the global centrality of the whole population. We can see that quite a number of nodes from Group A lie along the diagonal line. In this case, the global ranking can help to push the traffic toward Group A. However, the problem is that some nodes which have very high global rankings are actually not members of Group A, e.g. node D.

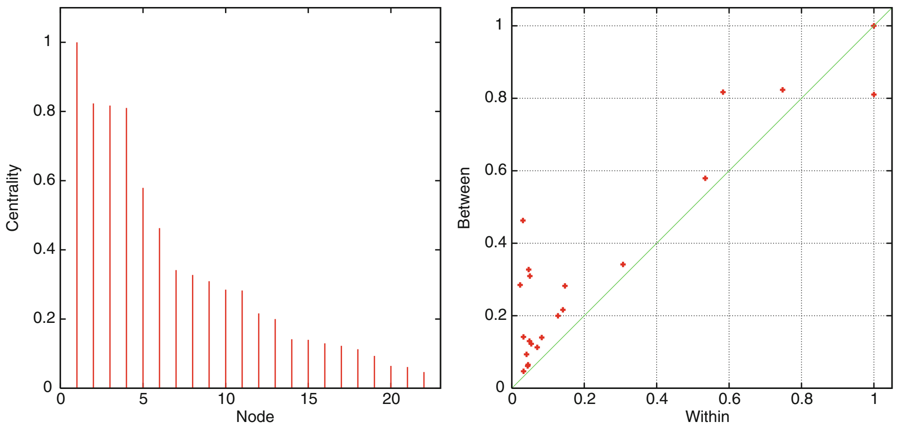


Fig. 12.15 Inter-group centrality (*left*) and correlation between intra- and inter-group centrality (*right*), *Cambridge*

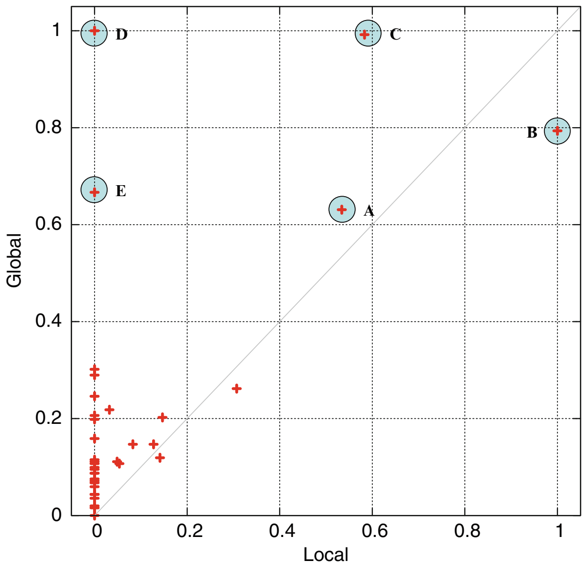


Fig. 12.16 Correlation of local centrality of group A and the global centrality (*Cambridge*)

Just as in real society, a politician could be very popular in the city of Cambridge, but not a member of the Computer Laboratory, so may not be a very good relay to deliver message to the member in the Computer Laboratory. Now we assume there is a message at node A to deliver to another member of Group A. According to global ranking, we would tend to push the traffic toward B, C, D, and E in the graph. If we pushed the traffic to node C, it would be fine, and to node B it would be

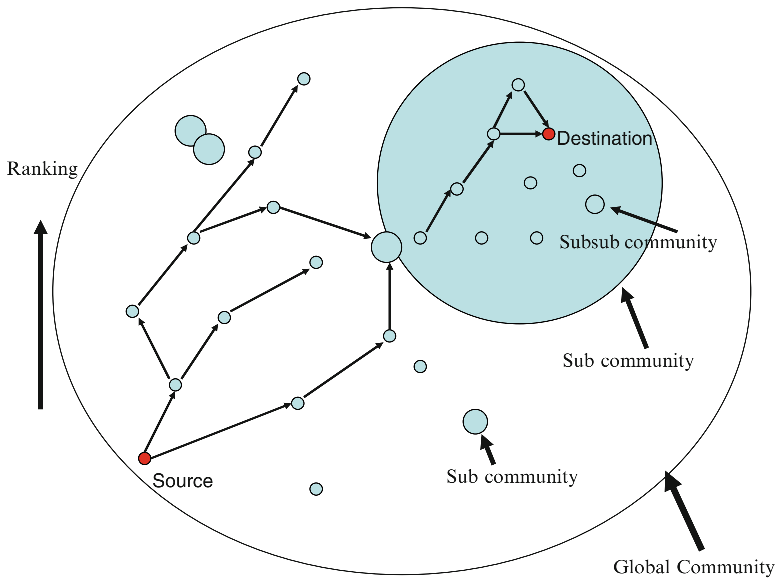


Fig. 12.17 Illustration of the BUBBLE forwarding algorithm

perfect. But if it pushed the traffic to node D and E, the traffic could get stuck there and not be routed back to Group A. If it reaches node B, that is the best relay for traffic within the group, but node D has a higher global ranking than B, and would tend to forward the traffic to node D, where it would probably get stuck again. Here, we propose the BUBBLE algorithm to avoid these dead-ends.

Forwarding is carried out as follows. If a node has a message destined for another node, this node would first bubble this message up the hierarchical ranking tree using the global ranking until it reaches a node which has the same label (community) as the destination of this message. Then the local ranking system will be used instead of the global ranking and continue to bubble up the message through the local ranking tree until the destination is reached or the message expired. This method does not require every node to know the ranking of all other nodes in the system, but just to be able to compare ranking with the node encountered, and to push the message using a greedy approach. We call this algorithm BUBBLE, since each world/community is like a bubble. Figure 12.17 illustrates the BUBBLE algorithm and the pseudo code can be found in our previous work [19].

This fits our intuition in terms of real life. First, you try to forward the data via people around you and are more popular than you, and then bubble it up to well-known popular people in the society, such as a postman. When the postman meets a member of the destination community, the message will be passed to that community. This community member will try to identify the more popular members within the community and bubble the message up again within the local hierarchy until the message reaching a very popular member, or the destination itself, or the message expires.

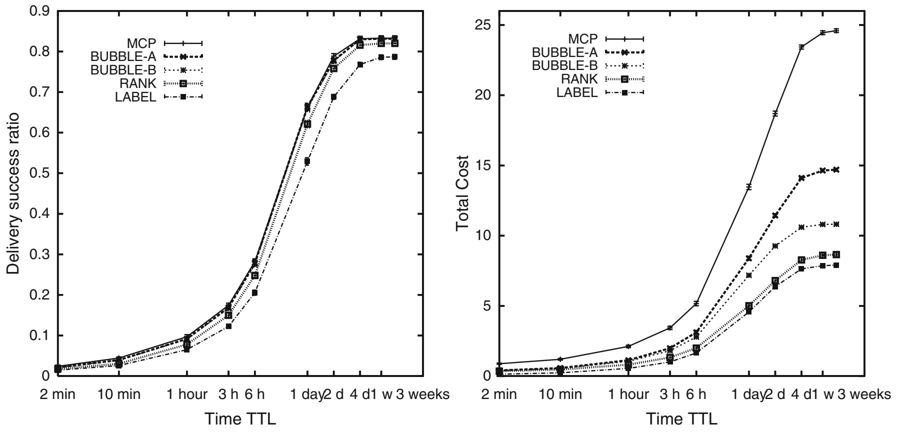


Fig. 12.18 Comparisons of several algorithms on *Cambridge* dataset

A modified version of this strategy is that whenever a message is delivered to the community, the original carrier can delete this message from its buffer to prevent it from further dissemination. This assumes that the community member would be able to deliver this message. We call this protocol with deletion, strategy BUBBLE-B, and the original algorithm introduced above BUBBLE-A.

We can see from Fig. 12.18 that both BUBBLE-A and BUBBLE-B achieve almost the same delivery success rate as the 4-copy-4-hop MCP. Although BUBBLE-B has the message deletion mechanism, it achieves exactly the same delivery as BUBBLE-A. BUBBLE-A only has 60% the cost of MCP and BUBBLE-B is even better, with only 45% the cost of MCP. Both have almost the same delivery success as MCP.

12.3.4.3 Multiple-Community Cases

To study the multiple-community cases, we use the *Reality* dataset. To evaluate the forwarding algorithm, we extract a 3-week session during term time from the whole 9-month dataset. Emulations are run over this dataset with uniformly generated traffic.

There is a total of 8 groups within the whole dataset. Figure 12.19 shows the node centrality in 4 groups, from small-size to medium-size and large-size groups. We can see that within each group, almost every node has different centrality.

In order to make our study easier, we first isolate the largest group in Fig. 12.19, consisting of 16 nodes. In this case, all the nodes in the system create traffic for members of this group. We can see from Fig. 12.20 that BUBBLE-A and BUBBLE-B perform very similarly to MCP most of the time in the single group case, and even outperform MCP when the time TTL is set to be larger than 1 week. BUBBLE-A only

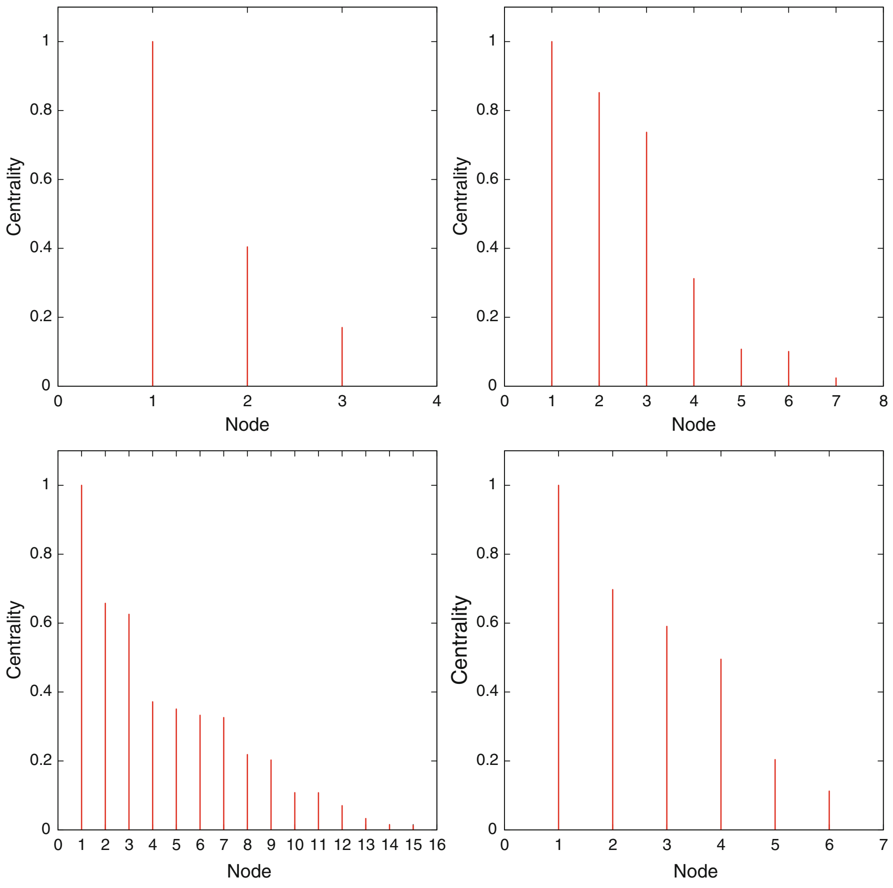


Fig. 12.19 Node centrality in several individual groups (*Reality*)

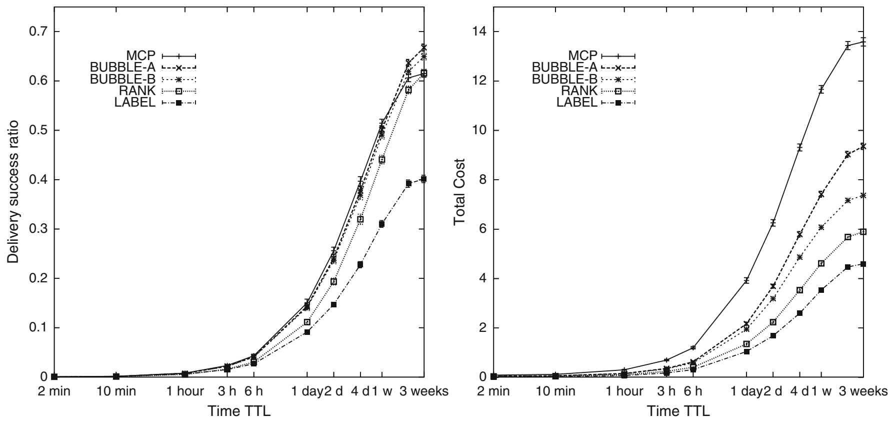


Fig. 12.20 Comparisons of several algorithms on *Reality* dataset, single group

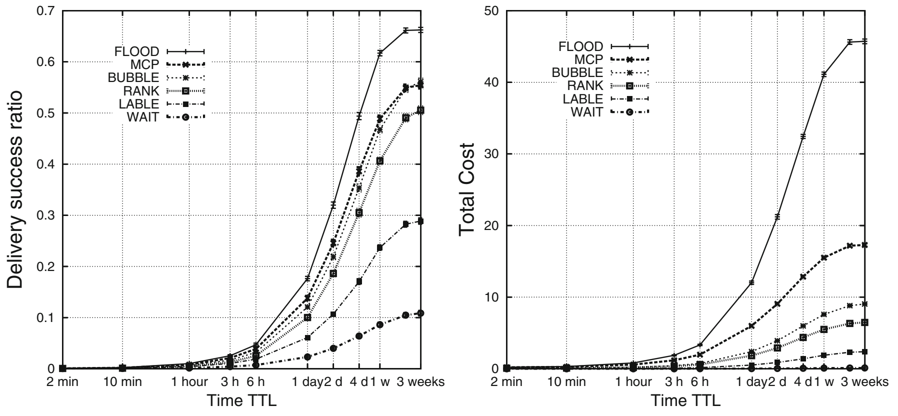


Fig. 12.21 Comparisons of several algorithms on *Reality* dataset, all groups

has 70% and BUBBLE-B only 55% of the cost of MCP. We can say that the BUBBLE algorithms are much more cost effective than MCP, with high delivery ratio and low delivery cost.

After the single group case, we start looking at the case of every group creating traffic for other groups, but not for its own members. We want to find the upper cost bound for the BUBBLE algorithm, so we do not consider local ranking (i.e., only global ranking); messages can now be sent to all members in the group. This is exactly a combination of direct LABEL and greedy RANK, using greedy RANK to move the messages away from the source group. We do not implement the mechanism to remove the original message after it has been delivered to the group member, so the cost here will represent an upper bound for the BUBBLE algorithms.

From Fig. 12.21, we can see that of course flooding achieves the best performance for delivery ratio, but the cost is 2.5 times that of MCP, and 5 times that of BUBBLE. BUBBLE is very close in performance to MCP in multiple groups case as well, and even outperforms it when the time TTL of the messages is allowed to be larger than 2 weeks.[5] However, the cost is only 50% that of MCP. Figure 12.22 shows the same performance evaluations with the *Infocom06* dataset. In this case, the delivery ratio of RANK is approaching that of MCP but with less than half of the cost. The performance of BUBBLE over RANK is not as significant as in the *Reality* case because in a conference scenario the people are very mixing and hence the community factors are less dominating. We can also see that even in this case,

[5] Two weeks seems to be very long, but as we have mentioned before, the *Reality* network is very sparse. We choose it mainly because it has long experimental period and hence more reliable community structures can be inferred. The evaluations here can serve as a proof of concept of the BUBBLE algorithm, although the delays are large in this dataset.

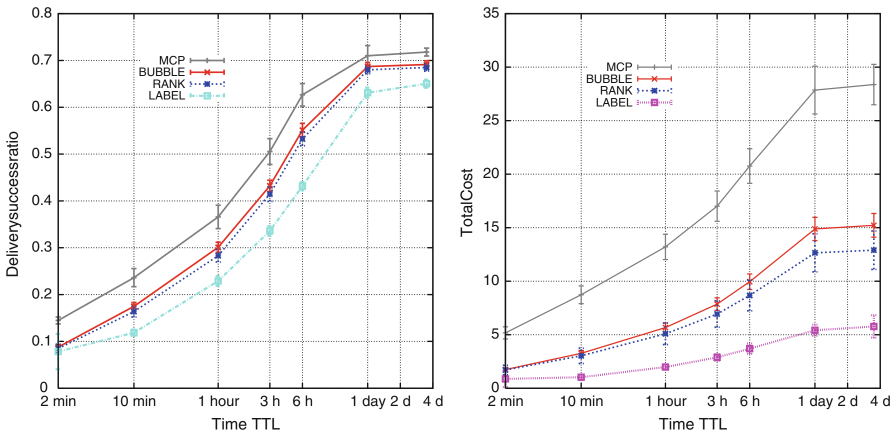


Fig. 12.22 Comparisons of several algorithms on *Infocom06* dataset, all groups

the delivery cost for BUBBLE only increases slightly, which indicates that even in a mixing environment, BUBBLE is still very robust towards the possible misleading of the community factors.

In BUBBLE and RANK algorithm, nodes with high centrality are more likely to act as relay nodes than the others. Excessive traffic through a node might cause the node to run out of battery or possibly lead to package losses. An easy fix is to impose admission control at each node. Each node maintains a limited buffer for storing data for other nodes and if the buffer has reached its limit, it will not admit incoming data. This may lower the delivery efficiency but can get rid of the excessive traffic problem. We will further study the trade-off and optimal buffer size in future work.

12.4 Related Work

Conceptually, PSNs are Delay-Tolerant Networks [12], and generic results from that framework apply. For instance, a forwarding algorithm that has more knowledge about contacts is likely to be more successful [22], and the best performance is achieved by an oracle with knowledge of future contacts.

Nevertheless, the fact that our underlying network is made up of human contacts and is less predictable has a large impact: for instance, reasonably predictable traffic patterns of buses allow a distributed computation of route metrics for packets in vehicular DTNs [2, 22]. Similarly, fixed bus routes allow the use of throwboxes [47] to reliably transfer data between nodes that visit the same location, but at different times.

The variability of PSNs has naturally led to a statistical approach: The inter-contact time distribution of human social contacts has been used to model transmission delay between a randomly chosen source–destination pair [4, 24]. In this work, we take a more macroscopic view and look at the ability of the PSN to simultaneously deliver data between multiple source–destination pairs. This leads us to look at the distribution of the *number* of contacts between randomly chosen source–destination pairs, and find that this distribution is not only crucial for global data delivery performance, but also for the connectivity of the PSN itself.

[14, 39, 45] model the performance of epidemic routing and its variants. In particular, they derive a closed form for delivery time distribution, and show it to be accurate for certain common mobility models. However, several simplifying assumptions are made, including an exponential inter-contact time between node pairs. Unforunately, human contact networks are known to have power law inter-contact times with exponential tails [4, 24]. Furthermore, [14, 39, 45] use a constant contact rate, whereas our studies show that human contacts are highly heterogeneous. [29] considers heterogeneous contact rates between mobile devices but only in the context of establishing an epidemic threshold for virus spread.

The number of paths found by flooding is crucial to the success of proportional and k-copy flooding. Counting differently, [10] reports a phenomenon of "path explosion" wherein thousands of paths reach a destination shortly after the first, many of which are duplicates, shifted in time. In contrast, duplicate paths are prevented in our method of counting, by having nodes remember if they have already received some data, resulting in a maximum of $N-2$ paths between a source and destination.

The power of using multiple paths has been recognised. Binary Spraying, which forms the basis for two schemes (spray and wait, spray and focus) has been shown to be optimal in the simple case when node movement is independent and identically distributed [40]. [11] noted that among routing schemes evaluated, those using more than one copy performed better. Furthermore, all algorithms employing multiple paths showed similar average delivery times. The success of k-copy flooding suggests a possible explanation for this result. Similarly, [21] finds that the delivery ratio achieved by a given time is largely independent of the propensity of nodes to carry other people's data. They suggest the existence of multiple paths as an explanation. At an abstract level, the refusal of a node to carry another node's data can be treated as a path failure. Thus, Sect. 12.2.3 corroborates [21] and provides a direct explanation.

For distributed search for nodes and content in power-law networks, Sarshar et al. [37] proposed using a probabilistic broadcast approach: sending out a query message to an edge with probability just above the bond[6] percolation threshold of the network. They show that if each node caches its directory via a short random walk, then the total number of accessible contents exhibits a first-order phase transition, ensuring very high hit rates just above the percolation threshold.

[6] A percolation which considers the lattice edges as the relevant entities.

For routing and forwarding in DTNs and mobile ad hoc networks, there is much existing literature. Vahdat et al. proposed epidemic routing, which is similar to the "oblivious" flooding scheme we evaluated in this chapter [43]. Spray and Wait is another "oblivious" flooding scheme but with a self-limited number of copies [40]. Grossglauser et al. proposed the two-hop relay schemes to improve the capacity of dense ad hoc networks [15]. Many approaches calculate the probability of delivery to the destination node, where the metrics are derived from the history of node contacts, spatial information and so forth. The pattern-based Mobyspace Routing by Leguay et al. [27], location-based routing by Lebrun et al. [26], context-based forwarding by Musolesi et al. [30] and PROPHET Routing [1] fall into this category. PROPHET uses past encounters to predict the probability of future encounters. The transitive nature of encounters is exploited, where indirectly encountering the destination node is evaluated. Message Ferry by Zhao et al. [46] takes a different approach by controlling the movement of each node.

Recent attempts to uncover a hidden stable network structure in DTNs such as social networks have been emerged. For example, SimBet Routing [5] uses ego-centric centrality and its social similarity. Messages are forwarded towards the node with higher centrality to increase the possibility of finding the potential carrier to the final destination. LABEL forwarding [17] uses affiliation information to help forwarding in PSNs based on the simple intuition that people belonging to the same community are likely to meet frequently, and thus act as suitable forwarders for messages destined for members of the same community. We have compared BUBBLE with LABEL and demostrate that by the exploitation of both community and centrality information, BUBBLE provide further improvement in forwarding efficiency. The mobility-assisted Island Hopping forwarding [36] uses network partitions that arise due to the distribution of nodes in space. Their clustering approach is based on the significant locations for the nodes and not for clustering nodes themselves. Clustering nodes is a complex task to understand the network structure for aid of forwarding.

Interested readers can obtain further details about the research presented in this chapter from [20] and [38].

12.5 Conclusion

This chapter discussed the idea of PSNs, which proposes to use human contacts to opportunistically transfer data over time from sender to destination. We first examined the feasibility of using local contacts for achieving global $N \times N$ connectivity in the temporal network formed by human contacts and showed that although the frequently occurring edges are not very effective for data transfer, the network exhibits a remarkable resilience in the face of path failures. We also showed that it is possible to uncover important characteristic properties of social network from a diverse set of real world human contact traces and demonstrated that community and centrality social metrics can be effectively used in forwarding decisions. Our

BUBBLE algorithm is designed for a delay tolerant network environment, built out of human-carried devices, and we have shown that it has similar delivery ratio to, but much lower resource utilisation than flooding and control flooding. We believe that this approach represents an early step in combining rich multi-level information of social structures and interactions to drive novel and effective means for disseminating data. A great deal of future research can follow.

References

1. A.Lindgren, A.Doria, O.Schelen: Probabilistic routing in intermittently connected networks. In: Proc. SAPIR (2004)
2. Balasubramanian, A., Levine, B.N., Venkataramani, A.: DTN Routing as a Resource Allocation Problem. In: SIGCOMM (2007)
3. Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., Scott, J.: Impact of human mobility on the design of opportunistic forwarding algorithms. In: Proc. INFOCOM (2006)
4. Chaintreau, A., et al.: Impact of human mobility on opportunistic forwarding algorithms. IEEE Transactions on Mobile Computing **6**(6), 606–620 (2007)
5. Daly, E., Haahr, M.: Social network analysis for routing in disconnected delay-tolerant manets. In: Proceedings of ACM MobiHoc (2007)
6. Danon, L., Duch, J., Diaz-Guilera, A., Arenas, A.: Comparing community structure identification. J. Stat. Mech. p. P09008 (2005)
7. Dunbar, R.I.M.: Co-evolution of neocortex size, group size and language in humans. Behavioral and Brain Sciences **16** (1993)
8. Eagle, N., Pentland, A.: Reality mining: sensing complex social systems. Personal and Ubiquitous Computing **V10**(4), 255–268 (2006)
9. Eagle, N., Pentland, A.S.: CRAWDAD data set mit/reality (v. 2005-07-01). http://crawdad.cs.dartmouth.edu/mit/reality
10. Erramilli, V., Chaintreau, A., Crovella, M., Diot, C.: Diversity of forwarding paths in pocket switched networks. In: Proceedings of ACM Internet Measurement Conference (2007)
11. Erramilli, V., Crovella, M., Chaintreau, A., Diot, C.: Delegation forwarding. In: MobiHoc (2008)
12. Fall, K.: A delay-tolerant network architecture for challenged internets. In: "SIGCOMM" (2003)
13. Freeman, L.C.: A set of measuring centrality based on betweenness. Sociometry **40**, 35–41 (1977)
14. Groenevelt, R., Nain, P., Koole, G.: The message delay in mobile ad hoc networks. Perform. Eval. **62**(1-4), 210–228 (2005)
15. Grossglauser, M., Tse, D.N.C.: Mobility increases the capacity of ad hoc wireless networks. IEEE/ACM Trans. Netw. **10**(4), 477–486 (2002)
16. Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J., Diot, C.: Pocket switched networks and the consequences of human mobility in conference environments. In: Proceedings of ACM SIGCOMM first workshop on delay tolerant networking and related topics (2005)
17. Hui, P., Crowcroft, J.: How small labels create big improvements. In: Proc. IEEE ICMAN (2007)
18. Hui, P., Crowcroft, J.: Human mobility models and opportunistic communications system design. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **366**(1872), 2005–2016 (2008)
19. Hui, P., Crowcroft, J., Yoneki, E.: Bubble rap: Social-based forwarding in delay tolerant networks. In: MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking & computing (2008)

20. Hui, P., Crowcroft, J., Yoneki, E.: BUBBLE rap: Social-based forwarding in delay-tolerant networks. Mobile Computing, IEEE Transactions on **10**(11), 1576–1589 (2011). doi:10.1109/TMC.2010.246
21. Hui, P., Xu, K., Li, V., Crowcroft, J., Latora, V., Lio, P.: Selfishness, altruism and message spreading in mobile social networks. In: Proc. of First IEEE International Workshop on Network Science For Communication Networks (NetSciCom09) (2009)
22. Jain, S., Fall, K., Patra, R.: Routing in a delay tolerant network. In: SIGCOMM (2004)
23. Jones, E.P.C., Li, L., Ward, P.A.S.: Practical routing in delay-tolerant networks. In: Proc. WDTN (2005)
24. Karagiannis, T., Le Boudec, J.Y., Vojnovic, M.: Power law and exponential decay of inter contact times between mobile devices. In: MOBICOM (2007)
25. Karagiannis, T., Le Boudec, J.Y., Vojnović, M.: Power law and exponential decay of inter contact times between mobile devices. In: ACM MobiCom '07 (2007)
26. Lebrun, J., Chuah, C.N., Ghosal, D., Zhang, M.: Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks. IEEE VTC **4**, 2289–2293 (2005)
27. Leguay, J., Friedman, T., Conan, V.: Evaluating mobility pattern space routing for DTNs. In: Proc. INFOCOM (2006)
28. Leguay, J., Lindgren, A., Scott, J., Friedman, T., Crowcroft, J.: Opportunistic content distribution in an urban setting. In: ACM CHANTS, pp. 205–212 (2006)
29. Mickens, J.W., Noble, B.D.: Modeling epidemic spreading in mobile environments. In: WiSe '05: Proceedings of the 4th ACM workshop on Wireless security (2005)
30. Musolesi, M., Hailes, S., et al.: Adaptive routing for intermittently connected mobile ad hoc networks. In: Proc. WOWMOM (2005)
31. Newman, M.E.J.: Analysis of weighted networks. Physical Review E **70**, 056,131 (2004)
32. Newman, M.E.J.: Detecting community structure in networks. Eur. Phys. J. B **38**, 321–330 (2004)
33. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69** (2004). DOI 10.1103/PhysRevE.69.026113. URL http://arxiv.org/abs/cond-mat/0308217
34. Okasha, S.: Altruism, group selection and correlated interaction. British Journal for the Philosophy of Science **56**(4), 703–725 (2005)
35. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**(7043), 814–818 (2005). DOI 10.1038/nature03607. URL http://dx.doi.org/10.1038/nature03607
36. Sarafijanovic-Djukic, N., Piorkowski, M., , Grossglauser, M.: Island hopping: Efficient mobility-assisted forwarding in partitioned networks. In: IEEE SECON (2006)
37. Sarshar, N., Boykin, P.O., Roychowdhury, V.P.: Scalable percolation search in power law networks (2004). URL http://arxiv.org/abs/cond-mat/0406152
38. Sastry, N., Manjunath, D., Sollins, K., Crowcroft, J.: Data delivery properties of human contact networks. Mobile Computing, IEEE Transactions on **10**(6), 868–880 (2011). doi:10.1109/TMC.2010.225
39. Small, T., Haas, Z.J.: The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In: MobiHoc (2003)
40. Spyropoulos, T., Psounis, K., Raghavendra, C.S.: Efficient routing in intermittently connected mobile networks: the multiple-copy case. IEEE/ACM Trans. Netw. **16**(1) (2008)
41. Travers, J., Milgram, S.: An experimental study of the small world problem. Sociometry **32**(4), 425–443 (1969)
42. "UCSD": Wireless topology discovery project. http://sysnet.ucsd.edu/wtd/wtd.html (2004)
43. Vahdat, A., Becker, D.: Epidemic routing for partially connected ad hoc networks. Tech. Rep. CS-200006, Duke University (2000)
44. Yoneki, E., Hui, P., Crowcroft, J.: Visualizing community detection in opportunistic networks. In: CHANTS'07: Proc. of the second ACM workshop on Challenged Networks, pp. 93–96 (2007). DOI http://doi.acm.org/10.1145/1287791.1287810

45. Zhang, X., Neglia, G., Kurose, J., Towsley, D.: Performance modeling of epidemic routing. Comput. Netw. **51**(10), 2867–2891 (2007)
46. Zhao, W., Ammar, M., Zegura, E.: A message ferrying approach for data delivery in sparse mobile ad hoc networks. In: Proceedings of the MobiCom 2004 (2004)
47. Zhao, W., et al.: Capacity Enhancement using Throwboxes in DTNs. In: Proc. IEEE Intl Conf on Mobile Ad hoc and Sensor Systems (MASS) (2006)

Chapter 13
Social Forwarding in Mobile Opportunistic Networks: A Case of PeopleRank

Abderrahmen Mtibaa, Martin May, and Mostafa Ammar

Abstract The proliferation of powerful portable devices has created a new environment for networking. In such environment, devices are able to communicate between "challenged" networks such as sensor networks, mobile ad-hoc networks, or opportunistic ad-hoc networks using a set of protocols designed to accommodate disconnection. In particular, forwarding in mobile opportunistic networks needs to deal with such disconnections, and limited resources. As opposed to conventional communication that relies on infrastructure, these devices can use hop-by-hop opportunistic data forwarding between each other. In this environment, a device should decide whether or not to transfer a message at the time it meets another one. How to optimally select the next hop towards the destination in a way to minimize delay and maximize success rate is so far unknown. In opportunistic networks, a device has to decide whether or not to forward data to an intermediate node that it encounters. In this chapter, we describe PeopleRank as systematic approach to the use of social interaction as a means to guide forwarding decisions in an opportunistic network. PeopleRank ranks nodes using a tunable weighted combination of social and contact information. It gives higher weight to the social information in cases where there is correlation between that information and the contact trace information. More specifically, PeopleRank is an opportunistic forwarding algorithm that ranks the "importance" of a node using a combination of social and contact-graph information.

A. Mtibaa (✉)
Carnegie Mellon University Doha, Qatar
e-mail: amtibaa@cmu.edu

M. May
Technicolor Paris, France
e-mail: martin.may@technicolor.com

M. Ammar
Georgia Institute of Technology Atlanta, Georgia, USA
e-mail: ammar@cc.gatech.edu

13.1 Introduction

In the ancient times, social interaction primarily took place through physical meeting. The telegraph and telephone networks made a first step toward remote social interaction. More recently, the Internet added multiple social interaction techniques not based on physical meeting: email, chat, and Online Social Network services (OSN) such as Facebook, Orkut, MySpace, or LinkedIn, etc. These applications create a virtual space where users can build the social network of their acquaintances independently of where they are located, and allow these social networks (or communities) to interact freely using a large set of Internet applications. However, when people, with similar interests or common acquaintances, get close to each others in streets or conferences, they have no automated way to identify this potential "relationship." With geolocation applications, it is now highly likely that OSNs will include in a near future some representation of user location, and offer services to "link" mobile users. However, the relation between virtual social interactions and physical meeting remains largely unexplored.

In this chapter, we present a systematic approach to the use of social interaction as a means to guide forwarding decisions in an opportunistic ad-hoc network. Generally, social interaction information alone is not sufficient and needs to be augmented in some way with information about contact statistics. The approach described in this chapter combines these two pieces of information.

This approach relies on the modeling of social relations using a *social graph* and modeling contact information as a time-varying graph. Such a social graph can be extracted from Online Social Networks or alternatively from shared interest information obtained through surveys or other means. The main challenge in combining social and contact information to guide forwarding decisions stems from the significant structural differences between these two sources of information. To approach this problem PeopleRank was designed to rank nodes using a tunable weighted combination of social and contact information. Such technique gives higher weight to the social information in cases where there is correlation between that information and the contact trace information.

More specifically, we introduce in this chapter an opportunistic forwarding algorithm that uses PeopleRank which ranks the "importance" of a node using a combination of social and contact-graph information. PeopleRank is inspired by the PageRank algorithm [2] used in Google's search engine to measure the relative importance of a Web page within a set of pages.

13.2 Web Ranking

Web ranking techniques are challenging because of the size of the Web: it is impossible for users to browse through millions of web search results to identify the most relevant ones. Thus, ranking techniques try to identify the most relevant (interested) results in a small top pages list.

13.2.1 Effective Web Ranking Algorithms

In most previous work, three "famous" algorithms tried to rank the web pages:

- The *InDegree* Algorithm [12] is considered to be the first study to rank the pages according to their popularity. Page popularity is defined as a number of pages that link to this page. This simple heuristic was applied by several search engines in the early days of Web search.
- The PageRank Algorithm [2] improves the *InDegree* Algorithm by adding weights to pages according to their qualities (the quality of a page measures the "importance" of the pages' content). Links from pages of high quality should result in a higher weight. It is not only important to know how many pages point to a page, but also whether the relevance of these pages is high or low.

  The PageRank algorithm performs a random walk on the World Wide Web graph, where the nodes are pages, and the edges are links among the pages. It gives the probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. The PageRank is given by the following equation:

$$\mathrm{PR}(p_i) = \frac{1-d}{n} + d \sum_{p_j \in M(i)} \frac{\mathrm{PR}(p_j)}{L(j)}, \tag{13.1}$$

  where $p_1, p_2, \ldots, p_n$ are the pages, $M(i)$ is the set of pages that link to p_i, $L(j)$ is the number of outbound links on page p_j, and d is a damping factor which is defined as the probability, at any step, that the person will continue clicking on links. Various studies have tested different damping factors, but it is generally assumed that the damping factor should be set to around 0.85 for best performance.
- The *Hits* Algorithm [11] proposed by Kleinberg is based on the observation that not only "important" pages link to "important" pages, and special nodes could act as hubs containing a list of links to "important" pages. He defined two weights, authorities a and hub h to compute the importance of Web pages, and they are given by:

$$a_i = \sum_{j \in M(i)} h_j \quad \text{and} \quad h_j = \sum_{i \in L(j)} a_i. \tag{13.2}$$

  Where a_i and h_i are, respectively, the authority and the hub weights of the web page p_i.

The three previous algorithms were followed by a huge number of extensions and improvements. However, most of these contributions are limited to the case of Web search. Only few researchers tried to exploit these link analysis concepts in other domains. e.g., J. Morrison et al. [13] proposed a method based on the PageRank algorithm to generate prioritized gene lists using biological information.

13.2.2 PageRank Description

The PageRank algorithm performs a random walk on the World Wide Web graph, where the nodes are pages, and the edges are links between pages. It gives the probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank values are given by (13.1).

Google describe the idea behind PageRank as if we consider a random web surfer that starts from a random page, spend some time t, and chooses the next page by clicking on one of the links in the current page. If we assume that the rank of the page is proportional to the fraction of time that the surfer spent on that page, pages that are linked by many other pages (or by important ones) will be visited more often, which justifies the definition. Equation 13.1 allows the surfer to restart with probability $1-d$ from another page chosen randomly, instead of following a link.

Google also describes the PageRank as a vote from page A to page B if page A links to B. Moreover, the importance of the page is proportional to the volume of votes this page could give. We apply the same idea in our algorithm to tag people as "important" if they are linked (in a social sense) to many other "important" people. We assume that only friends could vote for each others because they are more likely to recommend each other.

In the same way, web pages are hyperlinked, one could establish a social graph between persons linked through social relationships such as friendship, or common interests. We denote such a social graph $G=(V,E)$ as a finite undirected graph with a vertex set V and an edge set E. An edge $(u,v)\in E$ if, and only if, there is a social interaction between nodes u and v (i.e., u and v are friends or they are sharing k common interests).

13.3 Network Models

We are interested in delivering data among a set of N mobile wireless nodes. Communication between two nodes is established when they are within radio range of each other. Data is forwarded from source to destination over these *contacts*. We model the evolution of contacts in the network by a time varying graph $G(t)=(V,E(t))$ with $N=|V|$. We assume that the network starts at time t_0 and ends at time T (T can be infinite). We call this temporal network the *contact graph*. Each $G(t)$ describes the contacts between nodes existing at time t. Such a time-varying graph model can be obtained from a mobility/contact trace[1] or from a mobility model along with knowledge of radio properties (e.g., radio range).

[1] http://www.crawdad.org, http://www.haggleproject.org

Because such networks exhibit intermittent connectivity, data is typically stored in intermediate nodes awaiting appropriate contact formation. Paths are constructed as a concatenation of contacts that are followed as they appear in the time varying graph. Among these paths, a path from $s \in V$ to $d \in V$ starting at time t_k is delay-optimal if it reaches the destination d in the earliest possible time. Delay-optimal paths for any starting time and any source–destination pair can be computed efficiently via dynamic programming. We assume next that nodes have infinite buffer size, and data could be exchanged through any contact between nodes.

Traditional work on routing in intermittently-connected networks uses available knowledge about the properties of the time-dependent contact graph to inform forwarding decisions. In this chapter, we are interested in augmenting this knowledge with information about social relationships among nodes. We model such social relationships using a (non-time varying) graph representing the social relationship between the mobile nodes, which we denote as $G_\mathrm{s} = (V_s, E_\mathrm{s})$. In general, we assume that $V_s \supseteq V$, that is some nodes in the social graph will not be part of our mobile network set. Social graphs reflect the interaction or interrelation between persons. Such information is available either in online social applications or could be extracted from the phone history or other sources. A link in the social graph between two nodes implies that these nodes are socially "connected" (e.g., friends in facebook or sharing a common interest).

Our main premise in this work is that one can, in many instances, expect that $G(t)$ and G_s to show some correlation as shown in [14]. Such correlation is exhibited in many ways: for example, two nodes that are socially connected may experience more direct contacts than nodes that are not, or a node that is well-connected socially with a large number of neighbors in the social graph may also experience a large number of contacts with a variety of nodes in the contact graph. However, there is no reason to believe that these two graphs are perfectly correlated.

13.3.1 Experimental Data Sets

Our analysis relies on five data sets collected in conference environments and the virtual world of SecondLife. In addition to the contact information, we established the social relationships between the experimentalists. Next, we describe in detail each data set; a summary of the corresponding parameters is given in Table 13.1.

MobiClique07 was described in details in Sect. 14. MobiClique07 is the most complete data set for an evaluation of opportunistic social forwarding, i.e., it contains mobility information and information about the social relation between the participants. Visitors of the CoNEXT 2007 conference were asked to carry a Smartphone device during three consecutive days with the MobiClique application installed. Prior to the experiment start, each participant was asked to indicate the participants of all CoNEXT participants he knew or had a connection to. During the experiment, the social networking application indicated when a contact, or a contact

Table 13.1 Data sets properties

	MobiClique07	MobiClique08	SecondLife	Infocom (Int.)	Infocom (FB)	Infocom (Profile)	Hope
Duration (days)	3.5	3.5	10	3	3	3	3
Social patterns	*Explicit*	*Explicit*	*Implicit*	*Implicit*	*Explicit*	*Explicit*	*Implicit*
# Connected nodes	27	22	150	65	47	62	414
# Edges	115	102	2,452	835	219	423	6,541
Average degree	9.5	9.2	32.7	25.7	9.3	13.6	31.6
Diameter	4	4	5	3	4	4	6
Median inter-contact	10 mn	10 mn	25 mn	15 mn	15 mn	15 mn	30 mn
Median contact time(s)	240	180	180	150	150	150	90

of a contact, was in Bluetooth range/neighborhood. This connection neighborhood was then displayed on the user's device which in turn could add new connections or delete existing connections based on the physical interaction consequent to the application notification.

MobiClique08 [15] experiment was performed at CoNEXT 2008 conference using smartphones with the MobiClique application installed. The main difference with MobiClique07 experiment is in the parameterization: we had 22 participants and the neighborhood discovery was randomized to be executed at intervals of 120 $+/-$ 45 s. In addition, the social profile of MobiClique was initialized based on the user's Facebook profile. Prior to the experiment, each participant was asked to join a Facebook group of the experiment. During the initialization, participants could choose the people they considered as friends from the list of members in that group (instead of using the list of friends in Facebook). The initial list of interests contained user-selected Facebook groups and networks from his profile. As in MobiClique07 experiment, the social network evolved throughout the experiment as users could make new friends and discover (and create) new groups (i.e., interest topics) and leave others. For the analysis we consider the collected contact trace and the final social graph of 22 devices (the rest of devices were not collecting data on each day of the experiment).

SecondLife dataset [17] is of different nature and illustrates virtual mobility combined with real social relations. The dataset is a collection of avatar movements in a popular region in SecondLife.[2] The trace was collected during 10 consecutive days. It includes the avatar positions every 30 s within this region, and their group

[2]http://secondlife.com

of interests (such as sport, music, jazz, etc.). We assume that contact opportunities are available when the geo-distance between two avatars is less than or equal to ten meters (usually the Bluetooth range used in simulations).

The social graph between avatars is constructed based on common interest groups between two avatars; two avatars that are members of the same group are linked in the social graph. Avatars are only tracked inside a crawled region, they may meet outside this region, however their contact parameters (i.e., contact duration, inter-contact time, etc.) are not logged in our trace. The crawling process used to collect this dataset provides view of a specific region in a virtual world. We consider only the avatars that were visiting the region at least three times during the 10 days of the experiment. The reason for this restriction are both technical and practical. Technical since the huge amount of data is difficult to process. But also practical; avatars seen once can not receive messages after they left the region. That means, the delay needed to reach these nodes is inf (we set this time to 10 days, the duration of the measurements). The delay distribution is therefore dominated by these values.

Infocom06 dataset also contains real user mobility as described in [3], however, since individual social relations were unknown, we had to use information on users' interests to determine the social network between the participants. The trace was collected with 78 participants during the IEEE Infocom 2006 conference. People were asked to carry an experimental device (i.e., an iMote) with them at all time. These devices were logging all contacts between participating devices (i.e., called here internal contacts) using a periodic scanning every 2 s. In addition, they logged connections established with other external Bluetooth-enabled devices (e.g., cell phones, PDAs). For this study, we are using results for internal contacts only. Questionnaires were given to participants to fill theirs nationalities, languages, countries, cities, academic affiliations and topic of interests. Based on theses information, we consider three different social graphs for this experiment; based on (1) their common topics of interest when two users are sharing k common interest, (2) their Facebook connectivity (obtained offline), and (3) their social profile (union of nationality, language, affiliation, and city). These three social graphs are presented respectively in Table 13.1 by Infocom(Int), Infocom(FB), and Infocom(Profile).

Hope dataset[3] was collected during the 17th HOPE conference. This experiment had a huge number of participants (around 770) to collect and exchange contact information (after an explicit connection setup using send/receive pings). The dataset contains the location of participants (30 s granularity) as well as their topics of interest in the conference. The dataset is publicly available in the CRAWDAD[4] database. The contact graph is computed based on geo-distance between two nodes

[3] http://www.thelasthope.org/

[4] http://www.crawdad.org/hope/amd

(similar to the method used for the SecondLife trace) and the social graph was build using common interests between two users. Note that there are only 414 nodes connected in the social graph.

13.3.2 Methodology: Paths in Temporal Networks

Each data set may be seen as a temporal network. More precisely, we represent it as a graph where edges are all labeled with a time interval, and there may be multiple edges between two nodes. A vertex represents a device. An edge from device u to device v, with label $[t^{\mathtt{beg}};t^{\mathtt{end}}]$, represents a contact, where u sees v during this time interval. The set of edges of this graph therefore includes all the contacts recorded by each device.

13.3.2.1 Paths Associated with a Sequence of Contacts

We intend to characterize and compute in an efficient way *all* the sequences of contacts that are available to transport a message in the network. Note that such paths might be using a direct connectivity (a contact period between the source and the destination) or may be made of several hops where intermediate contacts are used.

A sequence $(e_i = (v_{i-1}, v_i, [t_i^{\mathtt{beg}};t_i^{\mathtt{end}}]))_{i=1,\dots,n}$ of contacts is *valid* if it can be associated with a time respecting path from v_0 to v_n. In other words, it is valid if there exists a non-decreasing sequence of times $t_1 \le t_2 \le \dots \le t_n$ such that $t_i^{\mathtt{beg}} \le t_i \le t_i^{\mathtt{end}}$ for all i. An equivalent condition is given by:

$$\forall i = 1,\dots,n,\ t_i^{\mathtt{end}} \ge \max_{j<i} \left\{ t_j^{\mathtt{beg}} \right\}. \tag{13.3}$$

The time-respecting path associated with a sequence of contacts $(e_1,\dots,e_n)$ is not unique, but we can characterize all of them as follows. Let us formally define the *last departure* of this sequence as $\mathtt{LD(e)} = \min_i \left\{ t_i^{\mathtt{end}} \right\}$; and the *earliest arrival* as $\mathtt{EA(e)} = \max_i \left\{ t_i^{\mathtt{beg}} \right\}$.

From the definition of a time respecting path, we have:

(1) All paths associated with this sequence of contacts verify $t_1 \le \mathtt{LD}$ and $t_n \ge \mathtt{EA}$.

This property shows that the last departure is in fact the maximum possible starting time of a path using this sequence of contacts. Similarly the earliest arrival denotes the minimum possible ending time for a path using this sequence. These two optimums are attained, as one can immediately check the following.

(2) If $\mathtt{LD} \le \mathtt{EA}$, there is a path with $t_1 = \mathtt{LD}$, $t_n = \mathtt{EA}$.
(3) If $\mathtt{EA} \le \mathtt{LD}$, there is a path with $t_1 = t_2 = \dots = t_n = t$ for all $t \in [\mathtt{EA};\mathtt{LD}]$.

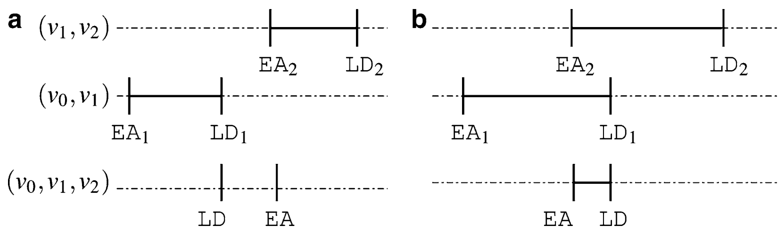


Fig. 13.1 Two examples of concatenation

13.3.2.2 Concatenation

Note that concatenating two sequences of contacts that both verify (13.3) does not necessarily create a compound sequence that verifies (13.3). However, we can characterize exactly when this concatenation is possible:

(4) Two sequences $(\mathtt{e}), (\mathtt{e}')$ of contacts such that $v_n = v'_0$ and that both verify (13.3) can be concatenated into a sequence of contacts $\mathtt{e}' \circ \mathtt{e}$ satisfying (13.3) if and only if $\mathtt{EA}(\mathtt{e}) \leq \mathtt{LD}(\mathtt{e}')$.

When the condition above is verified, we can deduce the values $\mathtt{LD}, \mathtt{EA}$ associated with the concatenated sequence as follows: $\mathtt{EA}(\mathtt{e}' \circ \mathtt{e}) = \max(\mathtt{EA}(\mathtt{e}), \mathtt{EA}(\mathtt{e}'))$, and $\mathtt{LD}(\mathtt{e}' \circ \mathtt{e}) = \min(\mathtt{LD}(\mathtt{e}), \mathtt{LD}(\mathtt{e}'))$ (see examples in Fig. 13.1). Note that $\mathtt{EA} = t^{\mathtt{beg}} \leq t^{\mathtt{end}} = \mathtt{LD}$ for a sequence made with a single contact, but sequences with multiple contacts, like Fig. 13.1 (a), might not verify $\mathtt{EA} \leq \mathtt{LD}$.

13.3.2.3 Delay-Optimal Paths

So far we have been describing a method to characterize when a sequence of contacts supports a time-respecting path, and when we can concatenate them. However, computing all of them in general is very costly. We show later how we can neglect many of the sequence of contacts to compute only those that are associated by a delay-optimal path.

13.3.2.4 Delivery Function

As a consequence of (2) and (3), for a message created at v_0 at time t, if $t \leq \mathtt{LD}$ then there exists a path associated with the sequence of contacts $\mathtt{e}$, that transports this message and delivers it to v_n at time $\max(t, \mathtt{EA})$. Otherwise, when $t > \mathtt{LD}$, no path based on these contacts exists to transport the message. The optimal delivery time of a message created at time t, on a path using this sequence of contacts, is given by

$$\mathtt{del}(t) = \begin{cases} \max(t, \mathtt{EA}) & \text{if } t \leq \mathtt{LD}, \\ \infty & \text{else.} \end{cases}$$

Similarly, the optimal delivery time for any paths that use one of the sequences of contacts $\mathtt{e}_1, \dots, \mathtt{e}_n$ is given by the minimum

$$\mathtt{del}(t) = \min\{\max(t, \mathtt{EA}_k), 1 \le k \le n \text{ s.t. } t \le \mathtt{LD}_k\}, \qquad (13.4)$$

where, following a usual convention, the minimum of an empty set is taken equal to ∞.

13.3.2.5 Optimal Paths

We say that a time respecting path, leaving device v_0 at time $t_{\mathtt{dep}}$, arriving in device v_n at time $t_{\mathtt{arr}}$, is strictly dominated in case there exists another path from v_0 to v_n with starting and ending times $t'_{\mathtt{dep}}, t'_{\mathtt{arr}}$ such that $(t'_{\mathtt{dep}} \ge t_{\mathtt{dep}} \text{ and } t'_{\mathtt{arr}} \le t_{\mathtt{arr}})$, and if at least one of these inequalities is strict. A path is said *optimal* if no other path strictly dominates it. In other words, any other path from v_0 to v_n, departing at $t'_{\mathtt{dep}}$ and arriving at $t'_{\mathtt{arr}}$ verifies:

$$(t'_{\mathtt{dep}} < t_{\mathtt{dep}}) \text{ or } (t'_{\mathtt{arr}} > t_{\mathtt{arr}}).$$

According to (2) and (3) above, among the paths associated with a sequence of contacts with values $(\mathtt{LD}, \mathtt{EA})$ the optimal ones are the following: if $\mathtt{LD} \le \mathtt{EA}$, this is the path starting at time $\mathtt{LD}$ and arriving at time $\mathtt{EA}$. Otherwise, when $\mathtt{LD} > \mathtt{EA}$, all paths that start and arrive at the message generation time $t \in [\mathtt{EA}; \mathtt{LD}]$ are optimal.

An example of delivery function is shown in Fig. 13.2. Note that the value of the delivery function (y-axis) may be infinite. Pairs $(\mathtt{LD}_1, \mathtt{EA}_1)$ to $(\mathtt{LD}_3, \mathtt{EA}_3)$ satisfy $\mathtt{EA} \le \mathtt{LD}$, they may correspond to direct source–destination contacts, or sequence of contacts that all intersect at some time; the fourth pair verifies $\mathtt{LD}_4 < \mathtt{EA}_4$, hence it does not correspond to a contemporaneous connectivity. The message needs to leave the source before $\mathtt{LD}_4$, and remains for sometime in an intermediate device before being delivered later at time $\mathtt{EA}_4$.

13.3.2.6 Efficient Computation of Optimal Paths

We construct the set of optimal paths, and delivery function for all source–destination pairs, as an induction on the set of contacts in the traces. We represent the delivery function for a given source–destination pair by a list of pairs of value $(\mathtt{LD}, \mathtt{EA})$. The key element in the computation is that only a subset of these pairs is needed to characterize the function $\mathtt{del}$. This subset corresponds to the number of discontinuities of the delivery function, and the number of optimal paths that can be constructed with different contact sequences.

We use the following observation: We assume that the values $(\mathtt{LD}_k, \mathtt{EA}_k)_{k=1,\dots,n}$, used to compute the delivery function as in (13.4), are increasing in their first coordinate. Then, as $k = n, n-1, \dots,$ we note that the kth pair can always be removed, leaving the function $\mathtt{del}$ unchanged, unless this pair verifies:

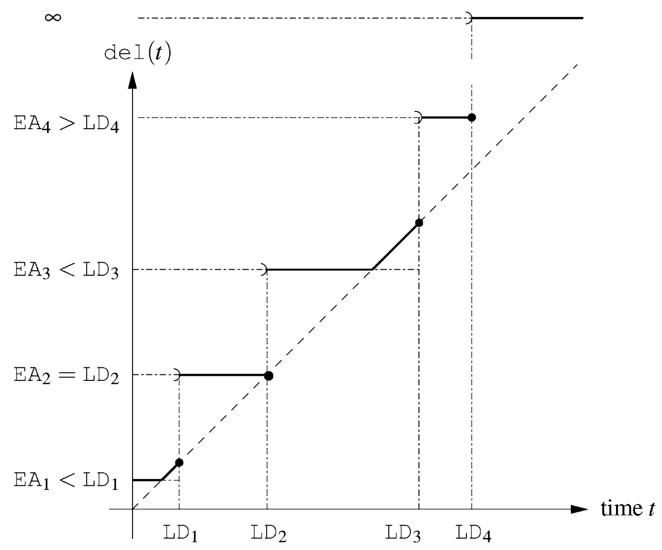


Fig. 13.2 Example of a delivery function, and the corresponding pairs of values $(\mathtt{LD}_i, \mathtt{EA}_i)_{i=1,2,3,4}$

$$\mathtt{EA}_k = \min\{\mathtt{EA}_l \ | \, l \geq k\} \, . \tag{13.5}$$

In other words, a list such that all pairs verify this condition describes all optimal paths, and the function `del`, using a minimum amount of information.

As a new contact is added to the graph, new sequences of contacts can be constructed thanks to the concatenation rule (fact (4) shown above). This creates a new set of values $(\mathtt{LD}, \mathtt{EA})$ to include in the list of different source–destination pairs. This inclusion can be done so that only the values corresponding to an optimal path are kept, following condition (13.5).

We show that our method can also be used to identify all paths that are optimal inside certain classes, for instance the class of paths with at most k hops. This can be done by computing all the optimal paths associated with sequences of at most k contacts, starting with $k = 1$, and using concatenation with edges on the right to deduce the next step.

Compared with previous generalized Dijkstra's algorithm [10], this algorithm computes directly representation of paths for all starting times. That is essential to have an exhaustive search for paths at any time-scale. We have introduced here an original specification through a concise representation of optimal paths which makes it feasible to analyze long traces with hundred thousands of contacts.

Recently, we have found that another algorithm has been developed independently to study minimum delay in DTN [18]. It works as follows: a packet is created for any beginning and end of contacts; a discrete event simulator is used to simulate flooding; the results are then merged using linear extrapolation.

13.3.3 Social Forwarding Algorithms

Let us consider a source node s which generated, at time t_0, a message m for a destination node d. We assume that each node $u \in G(t \geq t_0)$ can be a forwarder of this message m according to the store-carry-forward scheme.

We define a *social forwarding algorithm* as a store-carry-forward algorithm which uses a social utility function $f(G_s)$ in order to identify the most likely nodes to relay m (i.e., whether send m to the encountered node or not). Such social forwarding algorithms spread the message m among nodes (called also *relays*) who have specific social properties relying on $f(G_s)$.

All social forwarding algorithms we consider in this chapter, fit in the following general model: depending on the source s and the destination d, a path construction rule defines a subset of directed pairs of nodes $(u \to v)$ such that only the contacts occurring for pairs in the subset are allowed in forwarding path. We consider the following construction rules.

neighbor(k): $(u \to v)$ is allowed if and only if u and v are within distance k in the social graph.

destination-neighbor(k): $(u \to v)$ is allowed if and only if v is within distance k of d.

non-decreasing-centrality: $(u \to v)$ is allowed if and only if $C(u) \leq C(v)$.

non-increasing-distance: $(u \to v)$ is allowed if and only if the social distance from v to d is no more than the one from u to d.

target(k): $(u \to v)$ is allowed if and only if v and d are within distance k in the social graph.

In addition, we assume in addition that pairs $(u \to d)$ are allowed for all u, as any opportunity to complete the path with a single hop should not be missed. Each rule above defines a heuristic method to select among all the opportunistic contacts the ones that are crucial in order to connect source and destination quickly over time. Our objective is to design a rule that reduces as much as possible the contacts used, while allowing quasi-optimal delays.

In our evaluation, we deduce from the sequence of delay-optimal paths the delay obtained by the optimal path at all time. We combine all the observations of a mobility trace uniformly among all sources, destinations, and for every starting time (in seconds).

13.4 The PeopleRank Algorithm

In general, global knowledge of network topology can make for very efficient routing and forwarding decisions. Collecting and exchanging topology information in opportunistic networks is cumbersome because of their intermittent connectivity

Table 13.2 Our proposed PeopleRank algorithms

	Centralized	Distributed
Social input	PeopleRank	D-PeopleRank
Social+Contact input	CA-PeopleRank	DCA-PeopleRank

and unpredictable mobility. Routing schemes for such networks typically rely on partial knowledge and on prediction of future contacts which results in degraded routing performance.

With the emergence of Online Social Network platforms and applications such as Facebook, Orkut, or MySpace, information about the social interaction of users has became readily available. Moreover, while opportunistic contact information is changing constantly, the links and nodes in a social network remain rather stable.

The idea, is to use this more stable social information to augment available partial contact information in order to provide efficient data routing in opportunistic networks. The intuition behind this idea is that socially well connected nodes are better suited to forward messages towards any given destination. More specifically, we present a ranking algorithm that is inspired by more famous web page ranking. Thus, apply it to rank nodes based on their position in a social graph. This ranking is used then as a guide for forwarding decisions. More specifically, a node u forwards data to a node v that it meets if the rank of v is higher than the rank of u.

13.4.1 The Idea

We consider several approaches for computing a node's rank. As mentioned before, we are inspired by the PageRank algorithm used to rank web pages. By crawling the entire web, this algorithm measures the relative importance of a page within a graph (web). PeopleRank uses similar technique to rank the nodes in a social graph. We note that as opposed to PageRank algorithm, PeopleRank uses opportunistic contacts (physical proximity between two devices/users) to trigger the updates.

We consider two versions of PeopleRank : (1) a "pure" version in which only social graph information is used to compute the ranking and (2) a *contact-aware* version that augments the social ranking with some information about contact statistics between two social neighbors. Nodes with a higher PeopleRank value will generally be more "central" to either some combination of the social and the contact graphs.

As illustrated in Table 13.2, we consider four opportunistic forwarding algorithms based on assumptions using additional information (social and contact information) available during transfer opportunities. We, first, present two centralized algorithms; PeopleRank and *Contact Aware*-PeopleRank (CA-*PeopleRank*) using respectively either only social relationships between two nodes or social relationships augmented with contact frequency between these two nodes. Then, we consider two distributed implementations: Distributed-PeopleRank (D-*PeopleRank*) and Distributed Contact Aware-PeopleRank (DCA-*PeopleRank*).

13.4.2 *Centralized* PeopleRank

PeopleRank tags people as "important" when they are socially linked to many other "important" people. We assume that only neighbors in the social graph have an impact of the popularity (i.e., the ranking). In other words, we assume that friends are more likely to recommend theirs friends.

In the same way that web pages are hyperlinked, we establish a social graph between persons when they are socially related to each other. We denote such a social graph $G_\mathrm{s} = (V_s, E_\mathrm{s})$ as a finite undirected graph with a vertex set V and an edge set E_s. In the following, we define a social relationship between two nodes u and v either (1) if they are declared friends or (2) if they are sharing k common interests.

Consequently, the PeopleRank value is given by the following equation:

$$\mathrm{PeR}(N_i) = (1-d) + d \sum_{N_j \in F(N_i)} \frac{\mathrm{PeR}(N_j)}{|F(N_j)|}, \tag{13.6}$$

where $N_1, N_2, \ldots, N_n$ are the nodes, $F(N_i)$ is the set of neighbors that links to N_i, and d is the damping factor which is defined as the probability, at any encounter, that the social relation between the nodes helps to improve the rank of these nodes. This means that, the higher the value of d, the more the algorithm accounts for the social relation between the nodes. As a result, the damping factor is a very useful in controlling the weight given to the social relations for the forwarding decision. Such a mechanisms is very important since social graphs are built on different types of information. One could expect that a "friendship" between two individuals defines a stronger social relation than one defined by one or multiple common interests. When using PeopleRank for message forwarding, the damping factor should then be set to a value close to one for strong social relations and smaller for more loosely defined social graphs. In the next section, we address this issue in more detail and examine the impact of the damping factor on the PeopleRank performance.

The "pure" PeopleRank algorithm described above favors socially connected nodes. However, such social information defined by online social network applications does not always lead to a very accurate prediction of physical contact opportunities. In fact, people interact with each other in different ways; some have regular physically meetings, some are connected only in an online environment, and some are connected without ever communicating with each other. Since we are investigating how PeopleRank can be used for message forwarding, we augment its pure form to use statistical contact information *in addition* to social networking information.

To motivate the enhanced approach, we consider the contact and social graphs shown by Fig. 13.3. In addition to the social graph connecting the $N = 8$ nodes (Fig. 13.3(b)), we define the contact graph (Fig. 13.3(a)) where links are represented on: (1) solid lines, if nodes meet once every time unit, (2) dashed line (link 1–2)

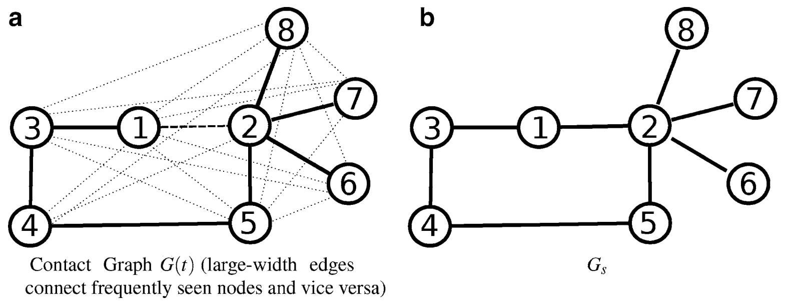

a
b
8
7
3
1
2
6
4
5
Contact Graph $G(t)$ (large-width edges connect frequently seen nodes and vice versa)
G_s

Fig. 13.3 Example of (**a**) contact graph, and (**b**) the associated social graph connecting 8 nodes

if nodes meet once every ten time units, and (3) dotted lines, if nodes meet once every 20 time units. We consider the case where *node 1* want to send a message to *node 6*.

In this example, suppose *node 1* has data to send to *node 6*, (13.6) yields that $\text{PeR}(3) < \text{PeR}(1) < \text{PeR}(2)$. As a result, *node 1* will wait to meet *node 2* to forward the message (because *node 2* is better ranked than *node 1*). However, *node 1* meets *node 3* nine times more than it meets *node 2* and the message may reach the destination faster if it is forwarded to *node 3*. Consequently, we propose to use the contact frequency between nodes as a weight of their social relationship[5].

With the above motivation in mind, We propose a Contact Aware-PeopleRank (CA- PeopleRank) which is an enhancement of the previous ("pure") version of PeopleRank . CA-PeopleRank uses the contact frequency between two social neighbors as a weight for the social link between these two nodes. The CA-PeopleRank value is given by:

$$\text{CA}-\text{PeR}(N_i) = (1-d) + d \sum_{N_j \in F(N_i)} \frac{w_{i,j} \times \text{CA}-\text{PeR}(N_j)}{|F(N_j)|}, \tag{13.7}$$

$$w_{i,j} = \frac{\pi(i,j)}{\sum_{j \in F(i)} \pi(i,j)}, \tag{13.8}$$

where $\pi(i,j)$ denotes the number of times node i and node j are in contact with each other.

Revisiting our example in Fig. 13.3, using 13.7 yields that $\text{CA}-\text{PeR}(1) < \text{CA}-\text{PeR}(2) < \text{CA}-\text{PeR}(3)$, and therefore *node 1* forwards the message to either *node 3* and *node 2*. This is the more desireable behavior since it will reduce message delay.

[5]Note that this method is not an aggregation of the contact graph into a social graph as proposed in [7, 8]) In our work, we augment social relationships with additional contact properties.

Algorithm 1 D-PeopleRank (i)

```
Require: $|F(i)| \geq 0$
  $\text{PeR}(i) \leftarrow 0$
  while 1 do
    while $i$ is in contact with $j$ do
      if $j \in F(i)$ then
        $update(\pi(i,j))$
        $send(\text{PeR}(i), |F(i)|)$
        $receive(\text{PeR}(j), |F(j)|)$
        $update(\text{PeR}(i))$                    (13.6 or 13.7)
      end if
      while $\exists\, m \in buffer(i)$ do
        if $\text{PeR}(j) \geq \text{PeR}(i)$ OR $j = destination(m)$ then
          $Forward(m, j)$
        end if
      end while
    end while
  end while
```

The techniques we described so far are suitable for centralized implementations where the social graph and the contact statistics are known a-priori. Clearly, this may not be feasible in the mobile wireless environment we are considering. We describe distributed versions of our algorithms in the next section.

Centralized architectures in general are of limited scalability since all data/information has to the transported and processed in a centralized entity which cause additional delays and increased transmission overhead. For web page indexing, centralized solutions are well suited, in a mobile ad hoc setting however, decentralized solutions are mandatory for efficient forwarding solutions (due to restrictions in energy, memory, bandwidth). Therefore, instead of centrally determine the rank of all nodes, we present in following a fully decentralized solution for calculating the PeopleRank value of each node.

13.4.3 Distributed PeopleRank

Centralized PeopleRank is computed centrally on a static graph by running the PeopleRank algorithm on the entire social graph (global knowledge of the social graph). The distributed version of PeopleRank is shown in Algorithm 1.

In this version, whenever two neighbor nodes in the social graph meet, they run separately a PeopleRank update process. They first update their contact counters ($\pi(i,j)$). Then, they exchange two pieces of information: (1) their current PeopleRank values and (2) the number of social graph neighbors they have. Next, the two neighbors update their PeopleRank values using either the formula given in (13.6) – for the D-PeopleRank – or using the one given in (13.7) – for the DCA-PeopleRank version. Finally, they (neighbor or not) run the message exchange process following a non-decreasing PeopleRank rule; e.g., assuming that node A has a message to send to a destination node D and meets another node B, A forward the message to B in two cases: (1) B is a destination D ($B = D$) or (2) PeopleRank$(A) \leq$ PeopleRank(B).

Note that, while centralized version of PeopleRank computes the ranking uniformly among all the friends, in the distributed algorithm frequently seen nodes update their PeopleRank values more often. Implicitly, this distributed version of the algorithm exploits the mobility and contact behavior of the nodes since the PeopleRank value is updated every time the nodes meet. In fact, the more often two nodes meet, the faster their rank increases. This will tend to "inflate" the social ranking (PeopleRank) for frequently seen nodes. However, this inflation is diminished after a certain time as PeopleRank values will converge to the centralized PeopleRank values.

Next, we describe in more detail the damping factor. Such factor is very important since social graphs are built on different types of information. We study the impact of the damping factor and the social patterns on the PeopleRank performance.

13.5 The Damping Factor

"Pure" social forwarding schemes are effective only for very accurate social information. If the social interaction is not very well captured in the social graph, it is more convinient to use smaller values of d to compensate for the mismatch between the social graph and the contact graph. And consequently, the more accurate the social information is, the more we can rely on the social graph for forwarding and hence, the closer to 1 we will chose the damping factor.

13.5.1 Optimal Damping Factor

We illustrate the dependency of the damping factor and the underlying social graph in Fig. 13.4 and Fig. 13.5.

We plot the CDF of the delay among all sources, destinations, and for every starting time using the MobiClique07 data set in Fig. 13.4(a), and the SecondLife data set in Fig. 13.4(b). We notice that the "optimal" d value may be different from one social trace to another (d around 0.9 in the MobiClique trace, and 0.8 in SecondLife). Note that, for a damping factor equal to 0.9 in the MobiClique07 data set, PeopleRank gives near to optimal success rates; the difference between the two algorithms is less than 2% within a 10 min timescale.

Figure 13.5 plots a normalized success rate of PeopleRank for a TTL of 10 min with different damping factors (i.e., we measure the success rate of PeopleRank normalized by the success rate of flooding within a delay of 10 min). It can be seen from this plot that optimal damping factor values change with the different

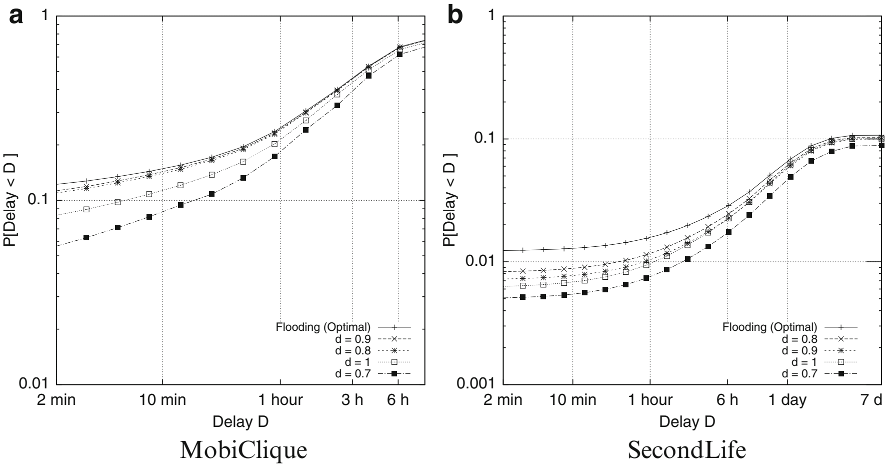


Fig. 13.4 PeopleRank performances in function of damping factors

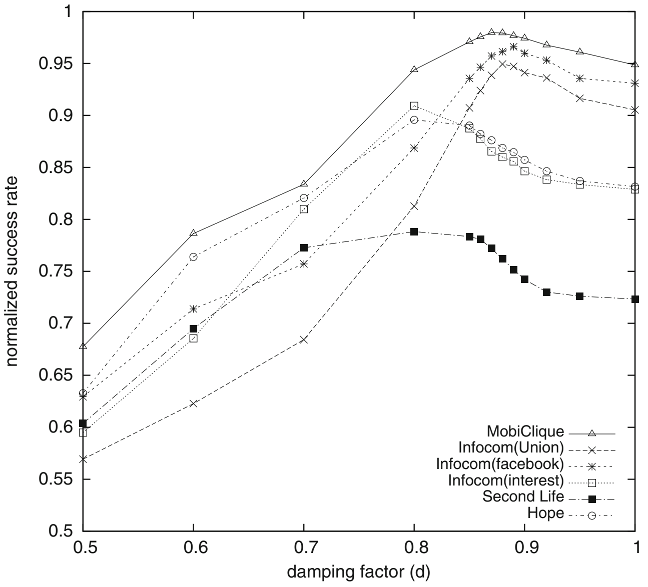


Fig. 13.5 Optimal damping factors in six data sets (within 10 min timescale)

traces (and the underlying social information). In fact, three traces (MobiClique07, Infocom(FB), and Infocom(Profile)) show the best performance with an optimal damping factor around 0.87 while the others traces (Hope, SecondLife, and Infocom(interest)) perform best for $d \approx 0.8$. The first three traces are based on explicitly defined connections, the three latter traces are built on implicit social connections. We conjecture that the reason for this difference lies in the way the social interactions are defined in each dataset. Implicit connections are defined as contacts between persons that share some common interests; we define a connection as explicit only when the two nodes declared a direct connection (e.g. links in applications like Facebook). Obviously, in traces with an explicit social pattern, the social graph information is more likely to be suitable for forwarding, and hence, damping factors of close to 1 perform best.

Figure 13.5 also shows that the curves decrease for a damping factor close to 1. This effect can be explained by the fact that a damping factor d equal to 1 considers exclusively socially connected nodes. Since also socially disconnected nodes are potentially able to deliver messages, the performance decreases when those nodes are not considered in the forwarding path. That means that even in networks with a high correlation between the social and contact graphs, some randomized forwarding is beneficial.

13.5.2 Determining Damping Factor

As mentioned earlier, the higher the value of d, the more differentiation there is among the nodes' PeopleRank values based on their position in the social graph. As a result we would like d to be high if there is high correlation between the contact and social graphs and low otherwise. To achieve this we define $div(G_s, G(t))$ as a metric that measures the divergence between the social and contact graphs. we then set $d = 1 - \mathrm{div}(G_s, G(t))$

Defining the difference between the social and contact graphs is challenging because the contact graph is dynamic. In the following, we propose two heuristic measures.

13.5.2.1 Edge-by-Edge Divergence

The Edge-by-Edge divergence compares the edges of both social and contact graphs. It measure the structural difference between the two graphs. We denote it div_1 and it is defined as:

$$\mathrm{div}_1(G_s, G(t)) = \sum_t \frac{|E(t) \setminus E_s|}{T \times |E(t)|}, \tag{13.9}$$

where $E(t)$ is the edge set of the contact graph at time t, E_s is the edge set of the social graph and T denotes the network lifetime. The summation above can be replaced by an integration if one were to consider continuous time in the edge set variation of the contact graph.

Because div_1 is comparing graphs on an edge-by-edge basis, the above metric is most meaningful if the vertex sets of the social and contact graphs are similar. Since

Table 13.3 Heuristics to determine damping factors

	Edge-by-Edge divergence (13.9)	Distance-based divergence (13.10)	Optimal damping factor (Fig. 13.4)
MobiClique	0.86	0.91	0.87
SecondLife	0.57	0.77	0.8
Infocom(FB)	0.78	0.89	0.88

this may not be the case, we propose another divergence metric which measures the correlation between the frequency of contacts between two nodes and their distance in the social graph.

13.5.2.2 Distance-Based Divergence

The second considered heuristic is the distance-based divergence which compares the contact rate between nodes and their social distance in the social graph. We denote this heuristic by div$_2$:

$$\mathrm{div}_2(G_\mathrm{s}, G(t)) = \frac{|\{d(u,w) \le d(v,w) \; and \; \pi(u,w) < \pi(v,w)\}|}{|V_s|(|V_s|-1)}, \tag{13.10}$$

where $d(u,w)$ denotes the distance in the social graph (i.e., considering a friendship graph, friends have distance 1, friends of friends have distance 2, etc.) between nodes u and w ($u \neq v \in V_s, w \in V_s$), and $\pi(u,v)$ denotes the number of times node u and node v are in contact with each other.

Table 13.3 presents a comparison of our two heuristics and the optimal damping factor given by the three datasets MobiClique, SecondLife, and Infocom. The heuristic based on (13.10) leads to a better approximation of the damping factor except for the MobiClique data set. However, as can be seen in Fig. 13.4, PeopleRank performs equally well for all damping factors between 0.91 and 0.97 (difference less than 2% of success rate). We conclude that the heuristic based on (13.10) is well suited to estimate an optimal value for d and consequently, use it in the rest of our analysis.

13.6 PeopleRank Performances

The performance of a forwarding algorithm like PeopleRank is determined by two conflicting factors: (1) the average message delivery delay and (2) the overhead (or cost) induced by the forwarding mechanism, i.e., the number of message replicas in the system. In the following, we assess the PeopleRank performance with regard to these two performance indicators.

We evaluate the forwarding algorithm using analysis on real traces. Specifically, we used the following experimental datasets: MobiClique, SecondLife, Infocom06, and Hope. Each data set includes both, a mobility or contact trace and a social interaction graph as described in Table 13.1.

13.6.1 Comparison to Social Algorithms

We compare the PeopleRank performance to the two previously described algorithms: degree-based and centrality-based forwarding algorithms. As a reminder, theses two algorithms are based on:

- Centrality-based forwarding: u forwards a message to v if, and only if, $C(u) \leq C(v)$. Where $C(u)$ denotes the betweenness centrality of node u measured as the occurrence of this node in all shortest paths connecting all other pairs of nodes.
- Degree-based forwarding: u forwards a message to v if, and only if, $d(u) \leq d(v)$. $d(u)$ denotes the degree of node u in the social graph (in a friendship graph, the degree is the number of friends of node u).

In the following, we plot the success rate of PeopleRank normalized by the success rate of flooding as a function of the message delivery delay for five different data sets. In addition, we indicate the cost of each algorithm in brackets as the fraction of contacts used by each forwarding algorithm normalized by the the fraction of contacts used by flooding.

Figure 13.6 plots the PeopleRank performance in the MobiClique07 dataset. Clearly, the PeopleRank algorithms outperform all other forwarding schemes in this dataset. D-PeopleRank and the centrality-based algorithm perform at around 90% of the optimal success rate (with a 10 min timescale) with a heavily reduced overhead; it uses only 50% of contacts compared to those used by flooding (see brackets in Fig. 13.6). Note that in MobiClique07, the explicit social information (friendship) is a rather good indicator helping to identify nodes that are likely to meet the destination. The CA-PeopleRank outperforms all the other algorithms used in the evaluation, it achieves roughly 96% of the success rate obtained with the Epidemic algorithm while using only 51% of the contacts.

Moreover, the distributed algorithms (D-PeopleRank and DCA-PeopleRank) perform with a similar good performance compared to the centralized algorithm (CA-PeopleRank). This is a surprising and unexpected to discover that the distributed version outperforms even the centralized version of PeopleRank ; indeed, in the centralized version, we assume that nodes have a global knowledge of all social interaction in the networks, however in the distributed version, nodes have only a local view of social interaction. Such surprising performances could be explained by the fact that, the distributed implementation and thanks to its opportunistic update process, favors frequently seen friends than others. Therefor, frequently seen friends are more involved to participate in the PeopleRank update process. However, in the

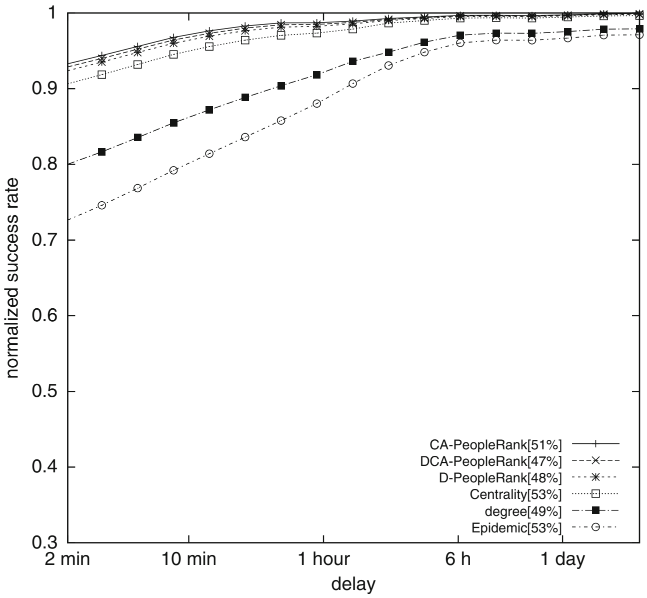


Fig. 13.6 Comparison of PeopleRank, Centrality-based, and Degree-based algorithms in the MobiClique07 data set– success rate (*main frame*) and cost (*in brackets*)

centralized version, all friends are involved with the same probability in the update process (i.e., even those who never meet physically in real life. Those nodes are unlikely to forward messages in opportunistic networks).

In the SecondLife data set (see Fig. 13.7), we observe also that PeopleRank algorithms outperform the centrality-based and the degree-based algorithms. D-PeopleRank achieves 8% and 14% higher success rate than the centrality and degree-based algorithms (for 10 min message delay). Moreover, adding contact information further improves performance of the algorithms; DCA-PeopleRank increases the success rate by 4% compared to D-PeopleRank . We observe, specifically in the SecondLife data set, that the degree-based forwarding algorithm outperforms the centrality-based algorithm for longer timescales (more than six hours). In other words, in the SecondLife data set, messages spending already a long time in the network without reaching destinations, are more likely to be forwarded to nodes with a high degree than others who are central in the social graph.

We get a good explanation in [17]. Indeed, Varvello et al. [17] show that nodes with a high degree of social connections tend to stay longer connected in SecondLife and hence, are more suitable to store and forward messages to other neighbors. They observed that roughly 30% of regions in SecondLife do not attract any visitors, and in few popular regions some avatars spend more than 12 h connected to a region. These avatars join many groups of interest and meet other avatars to interact with them. As result, these avatars become more and more connected in the social graph (large degree) and are likely to meet additional people and discuss with them.

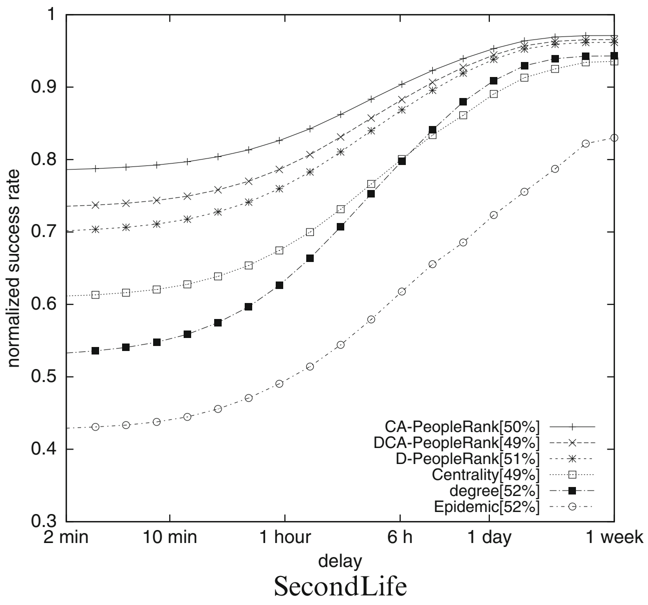


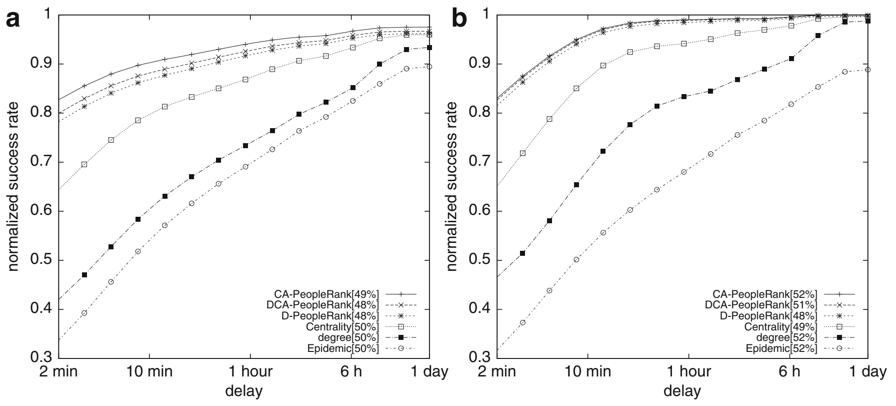
Fig. 13.7 Comparison of PeopleRank, Centrality-based, and Degree-based algorithms in the SecondLife data set– success rate (*main frame*) and cost (*in brackets*)

However, central nodes are not necessary always connected to the region. They join and leave the region, and socialize for a short time with others compared to well connected nodes (i.e., avatars with higher degree).

In Fig. 13.8, we plot the performance of the PeopleRank algorithms using the same contact trace of the Infocom participants but we establish two different social graphs for them; one (in Fig. 13.8(a)) uses the connections published by their Facebook profiles, the other graph (in Fig. 13.8(b)) is build based on common interests.

Comparing the two figures, Fig. 13.8(a) and (b), illustrates the difference in performance obtained with the two different social graphs. It emphasizes the impact of the relevance of social inputs on forwarding performances. PeopleRank uses (1) in Fig. 13.8(a), shared interests as an implicit social input, and (2) in Fig. 13.8(b), the friendship graph extracted from Facebook as an explicit social input in order to make forwarding decisions which reduce the number of message retransmissions and achieve good success rate . One can notice that PeopleRank algorithms achieve better performances relying on explicit social input (roughly 95% of success rate for CA-PeopleRank compared to the success rate of flooding within 10 min timescale).

In Fig. 13.8(b), the PeopleRank algorithms perform with a success rate roughly 95% normalized by the optimal flooding delay. They improve the performance by 45% compared to a random distribution while keeping the number of retransmissions small. However, for a social graph built on common interests, the D-PeopleRank performs only with a success rate of 82%. The combined social and



Infocom(Int): Social graph based on shared interests

Infocom(FB): Facebook social graph

Fig. 13.8 Comparison of PeopleRank, Centrality-based, and Degree-based algorithms in the Infocom06 data set– success rate (*main frame*) and cost (*in brackets*)

contact based DCA-PeopleRank algorithm however further improves the success rate. However, in Fig. 13.8(a), historical contacts are needed to improve the quality of the social input (implicit definition) and help PeopleRank algorithms to perform with roughly 90% of normalized success rate within the same timescale.

A similar observation can be made in Fig. 13.9: the D-PeopleRank algorithm performs with a 83% success rate while the DCA-PeopleRank achieves more than 89% at 10 min timescale. It appears that solely relying on social information for forwarding is not sufficient if the social graph is build on implicit social information. In situations where the social graph is not well captured, additional information about the node contacts are very helpful. The superior performance of DCA-PeopleRank indicates that adding historical contacts to shared interest information improves the estimation of the social network and improves the performance of the forwarding.

13.6.2 Comparison to Contact-Based Algorithms

In order to compare the performance of PeopleRank algorithm to non-social information based algorithms, we implemented additional contact-based forwarding algorithms. Those algorithms represent the most well-known algorithms in the literature, and as the name indicates, they use locally available contact information to decide whether to forward a message when two nodes meet:

- Last_Contact [4]: Node i forwards messages to node j if j has contacted any other node more recently than node i.

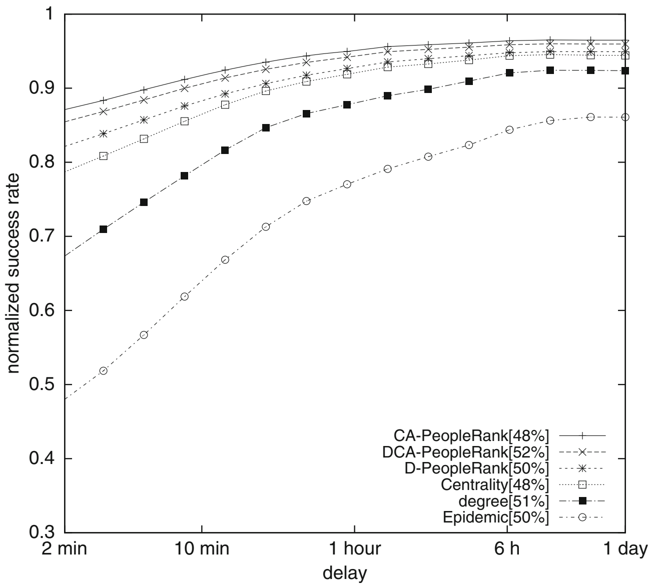


Fig. 13.9 Comparison of PeopleRank, Centrality-based, and Degree-based algorithms in the Hope data set– success rate (*main frame*) and cost (*in brackets*)

- Destination_Last_Contact [4]: Node i forwards messages to node j if j has contacted the message's destination more recently than has node i.
- Frequency [5]: Node i forwards the message to node j if j has more total contacts than node i.
- Spray&Wait [16]: the source node creates R (we use $R = 8$) replicas of the message. If node i has $k > 1$ replicas of the message and j has no replicas, i forwards $k/2$ replicas to j. Otherwise ($k = 1$) i wait the destination.
- Wait_Destination: the message is forwarded only if the source node meets its destination. Obviously, this algorithm results in minimum cost , however causes higher delay and lower success rate.

Figure 13.10 plots, for the MobiClique07 data set, the shortest-delay distribution and cost of D-PeopleRank , and the four selected forwarding schemes. Note that, instead of normalizing the success rate of all algorithms, we show in this figure, the original shortest-delay distribution given by these algorithms.

D-PeopleRank outperforms the Frequency, Last_Contact, and Destination_Last_Contact algorithms. In fact, D-PeopleRank uses almost the same number of message transmission (roughly 50% of the contacts used by flooding to forward data) but is more efficient (10–15% better success delivery within a 1-hour timescale) than the others four algorithms considered in this study. The Spray&Wait algorithm leads to a smaller number of retransmissions, however at a price of a poor success rate. Indeed, Spray&Wait controls the number of messages (a fixed number) in the networks.

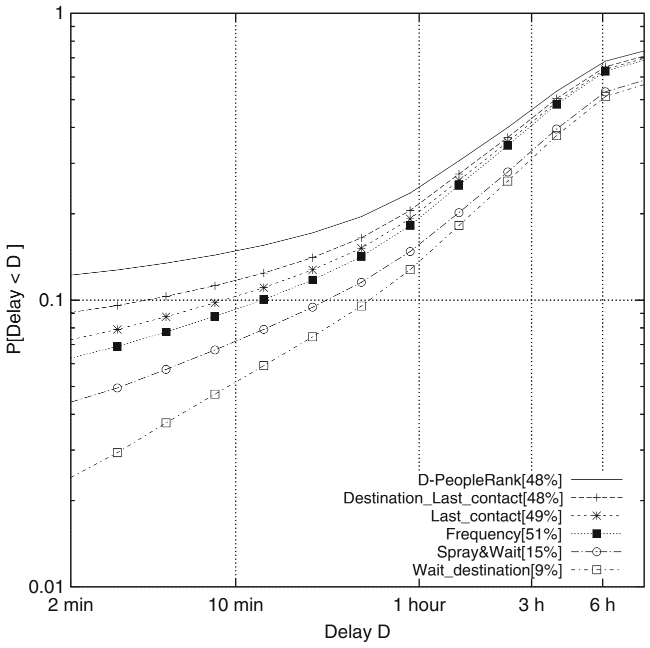


Fig. 13.10 Comparison of social, and contact-based forwarding in the MobiClique07 data set– success rate (*main frame*) and cost (*in brackets*)

However, it does not use any utility function to select the next forwarding node, it just sends the message to the first encounters. Spray&Wait success delivery is only 2% above the lower bound (obtained with the Wait_Destination technique).

Note that also other versions of PeopleRank algorithms outperform the contact-based forwarding schemes; for better readability we did not plot them in this figure. Indeed, we have shown in Fig. 13.6, that CA-PeopleRank , and DCA-PeopleRank outperforms the D-PeopleRank success rate which outperforms the contact-based algorithms' performances. We conclude that the PeopleRank algorithm efficiently reduces the number of message retransmission while keeping the success rate close to the optimal value. Key advantage of PeopleRank is that it avoids to retransmit messages to people that are unlikely to meet the destination.

13.6.3 Comparison to Ranking-Based Forwarding Algorithms

To evaluate the PeopleRank algorithms with regard to ranking-based forwarding algorithms, we next determine the results obtained with an "optimal" ranking forwarding algorithm. Following the methodology described above, we compute offline the sequence of optimal paths found between any source and destination

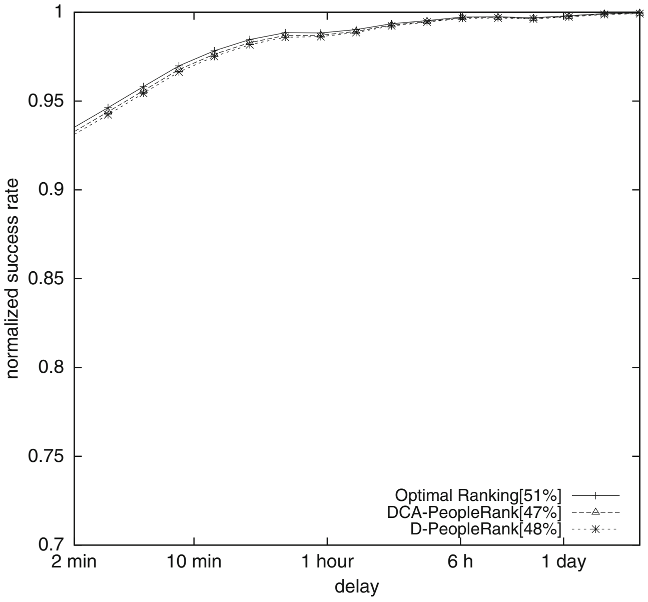


Fig. 13.11 Comparison of ranking-based forwarding algorithms in the MobiClique07 data set– success rate (*main frame*) and cost (*in brackets*)

in the network. We analyze the sequence of intermediate nodes used in delay-optimal paths, and we compute the “best” ranking vector. The optimal ranking algorithm then uses the “best” ranking vector and forces the forwarding to go through the maximum identified sequence of intermediate nodes used in delay-optimal paths. We implement this algorithm in order to compare its performance to those given by PeopleRank algorithms. Note that, this optimal ranking algorithm differs from the flooding algorithm since it uses a subset of nodes to transmit data instead of using all node (flooding). However, this algorithm gives optimal performance in its category (i.e., ranking forwarding algorithms).

Next, we compare the normalized success rate and cost of D-PeopleRank , DCA-PeopleRank , and the optimal ranking algorithm for three data sets; MobiClique07, SecondLife, and Hope. In Fig. 13.11, DCA-PeopleRank and D-PeopleRank lead to very similar performance compared to the optimal ranking scheme in the MobiClique07 data set (less than 0.5% difference between success rates). Indeed, thanks to the explicit social input used in MobiClique07, PeopleRank algorithms succeeded to make near to optimal forwarding decisions. However, for data sets where the social graph is defined by implicit social relations (Hope (Fig. 13.13) and SecondLife (Fig. 13.12)), D-PeopleRank falls of the performance obtained with the optimal ranking schemes. But again, the additional use of contact information improves significantly the success rate and the DCA-PeopleRank performance is

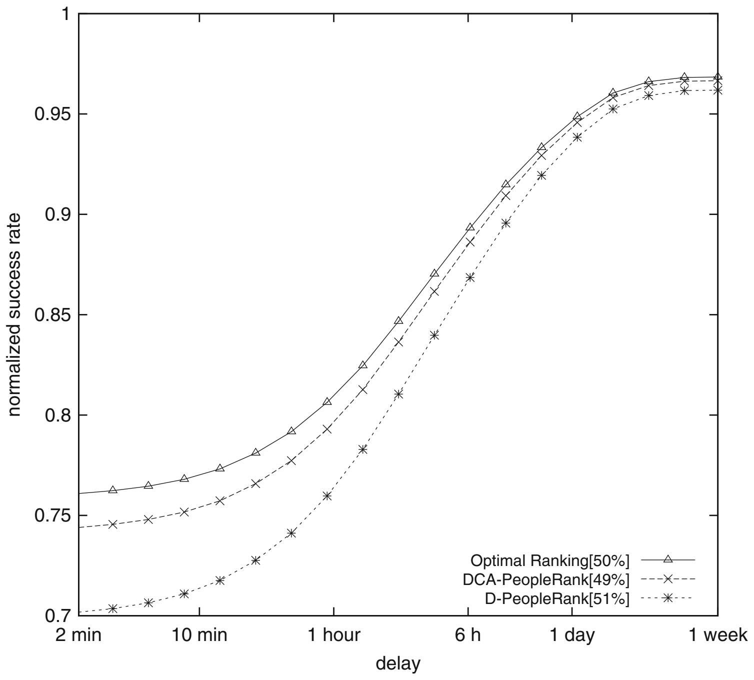


Fig. 13.12 Comparison of ranking forwarding algorithms in the SecondLife data set– success rate (*main frame*) and cost (*in brackets*)

close to the optimal ranking algorithm (success rate of DCA-PeopleRank is only 3% less than the optimal ranking for SecondLife (Fig. 13.12) and 5% for Hope (Fig. 13.13)).

13.6.4 Cost Reduction

The cost of a forwarding algorithm, defined as the fraction of contacts involved in the forwarding process, is very important in opportunistic networks. We have shown that PeopleRank algorithm reduces the number of message transmission (cost) by a factor of 2 compared to flooding. In this section, we study more options in order to reduce the cost of PeopleRank algorithms by a factor greater than 2. To reduce the cost of the D-PeopleRank algorithm, we combine the algorithm with other existing methods such as Spray&wait, TimeToLive (TTL: defined as the maximum number of hops allowed to reach the destination), or delegation forwarding [6]. The main idea of delegation forwarding is to associate, in addition to the node's rank, a threshold value to the message (initially equal to rank of the node). With such threshold, low ranked nodes are unlikely to receive messages since the threshold is continuously increasing.

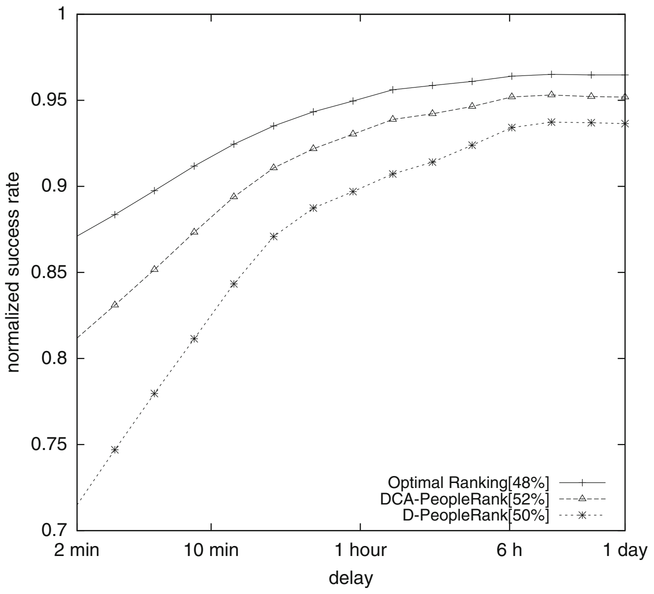


Fig. 13.13 Comparison of ranking forwarding algorithms in the Hope data set– success rate (*main frame*) and cost (*in brackets*)

Figure 13.14 plots the shortest-delay distribution and the cost of D-PeopleRank algorithm combined with a delegation forwarding, TTL = 2, and TTL = 4. When reducing the number of replicated messages with the D-PeopleRank algorithm, we also reduce the number of opportunities to meet the destination. However, D-PeopleRank combined with a TTL = 4 gives near optimal performance (i.e., performance without any mechanism to reduce the cost). This phenomena has already been shown in [3]; opportunistic mobile networks are characterized by a small diameter, a destination device is reachable using only a small number of relays under tight delay constraint. Delegation forwarding could also be effective to reduce the number of message replicas in the network. It achieves one of the best cost/success rate trade-offs. It reduces the number of replicas by more than 10% (compared to D-PeopleRank) but loses only 2% of the forwarding opportunities within 10 min timescale.

13.6.5 Fair Load Distribution

Memory and energy consumption are important for small mobile devices. To avoid the abuse of some few nodes for message forwarding, we assume that fair share of ressources is an important issue. PeopleRank is designed to efficiently use socially well connected node. Theses nodes could suffer under higher message forwarding

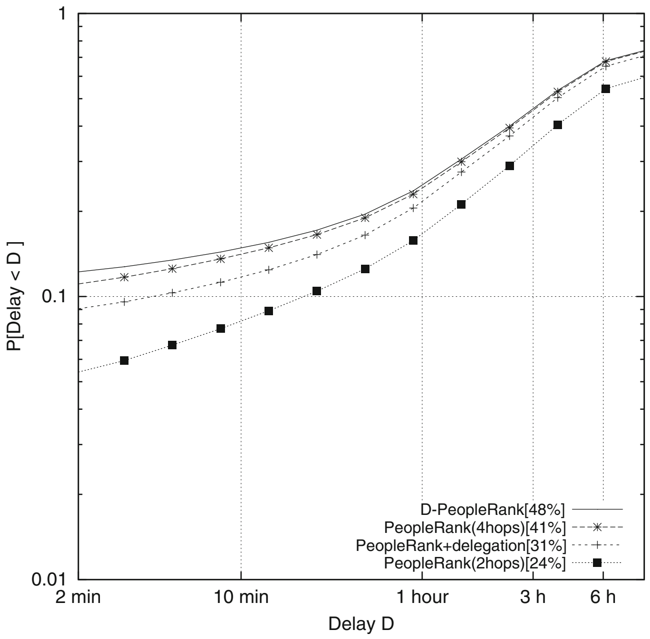


Fig. 13.14 Cost of the different PeopleRank algorithms in the MobiClique07 data set

requests than others (less ranked). Few nodes could be very solicited which may causes unfair capacity and ressource allocations. To emphasize this problem we measure the fairness metric to determine the impact of damping factors on ressource sharing in the network. We use the Jain's fairness metric [9], denoted by F, measure the fairness of ressource allocation in the network:

$$F = \frac{\left(\sum_{1}^{n} x_i\right)^2}{\left(n \cdot \sum_{1}^{n} x_i^2\right)}, \tag{13.11}$$

where x_i is the number of messages in the node i's buffer ($1 \le i \le N$). The best case is reached when F is close to 1 (fair allocation), and the worst case is given by $1/N$.

Figure 13.15 plots Jain's fairness values F in four data sets (MobiClique07, SecondLife, Infocom(FB), and Infocom(Int)). As noticed before, with a smaller damping factors, PeopleRank algorithm selects nodes uniformally to forward packets (F values are close to 1). The fairness index decreases by roughly 20% for all the dataset studied when the damping factor d increases. Since we are not interested in damping factors closer to zero (it gives more randomness to PeopleRank), only 5–10% of additional nodes become unfair from d equal to 0.4 to d equal to 1, and it

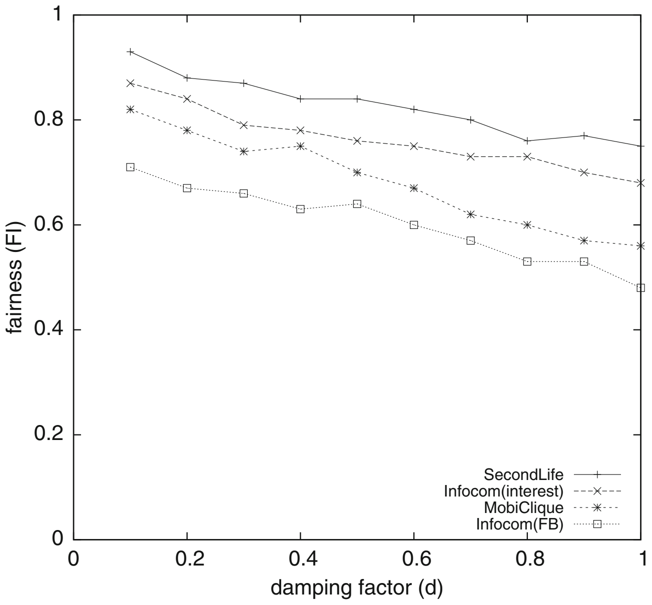


Fig. 13.15 Fairness values as a function of damping factors

is still closer to the best case ($F = 1$) than the worst case (given by $F = \frac{1}{N} = 0.03$ for MobiClique07). Similar observations are shown in the other data sets considered in Fig. 13.15 (SecondLife, Infocom(FB), and Infocom(Int) data sets).

In reality, fairness does not necessary mean equal distribution of message replicas among nodes in the network. In some cases, it is justifiable to use more ressources of some nodes. However, this "unfairness" should not affect a few set of nodes. It was shown (in using conferences data sets) that PeopleRank did not select THE best ranked node ever to reach the destination, it just selects a "relatively" well ranked node to reduce "efficiently" (i.e., keeping near optimal end-to-end delay and the success ratio) the number of message replicas.

13.6.6 Summary and Limitations

The results highlight more generally the superior performance of PeopleRank algorithm; it achieves an end-to-end delay and a success rate close to those obtained by flooding while reducing the number of retransmission to less than 50% of the ones induced by flooding.

These results, which are derived from three real human mobility traces and one virtual human mobility (SecondLife) trace, are restricted to small communities such as conferences and regions in SecondLife. However in large networks, the transmission of messages through the most socially important people will ultimately consume most of theirs resources.

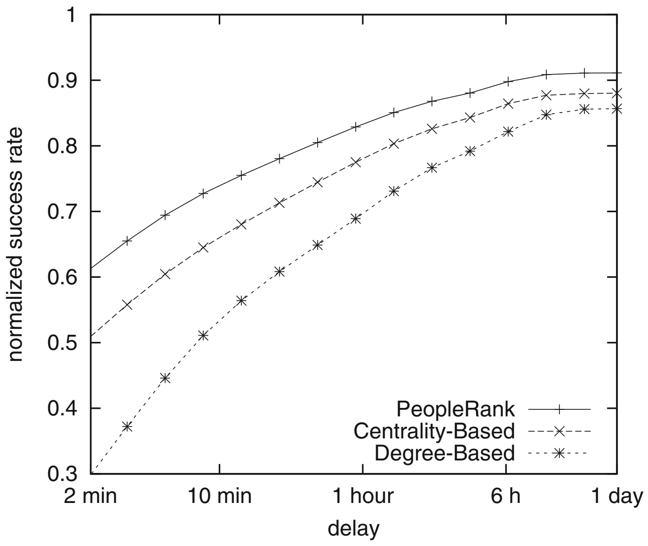


Fig. 13.16 Scalability issues of social-based algorithm in the Dartmouth data set

Moreover, it is hard to defend the assumption that a subset of socially well ranked nodes will physically meet all other nodes in a large network. Such assumption was verified in the previous sections of this chapter on small data sets regarding single social communities. Next, we verify this assumption in a large network regarding a multi-communities social graph.

We used the WiFi access network of Dartmouth campus [1] to consider a larger experimental data set. This data set is especially useful for user mobility research since it spans 200 acres (roughly 1300$\times$1300 square meters) and over 160 buildings, and about 550 802.11 b connectivity throughout. The data set contains logs of client MAC addresses, and names of access points (as well as their positions). We assume that two nodes are able to communicate if they are attached at the same time to the same access point. Dartmouth college covers student residences, sport infrastructures, administrative buildings, and academic buildings.

Since we are measuring, here, the scalability of PeopleRank as well as other social forwarding algorithms (Degree-Based and Centrality-Based forwarding), we considered an optimal scenario where the social interaction between nodes are well correlated to their contact patterns (given by the Dartmouth campus data set). We artificially created a social graph regarding the contact rates between nodes. We have seen in previous observations that such accurate social inputs deal with a "satisfactory" forwarding performances in single community data sets.

Figure 13.16 plots the normalized success rate of three social forwarding algorithms (D-PeopleRank , Degree-Based, and Centrality-Based forwarding) with respect to the Dartmouth data set described previously in this section. One could notice that despite the fact that social inputs match with the contact properties of nodes, there are 25–55% of losses compared to Epidemic forwarding, within these 10 min timescales. In fact, in large scale networks social forwarding algorithms loose many opportunities to reach destinations in optimal delays.

Therefore, social forwarding suffers in large scale networks networks, where people are divided into communities. Within a community, people are more likely to meet and interact with each others. They may be socially linked to other nodes from other communities, however, they are unlikely to meet each other in real life. In the next section, we present an extention of PeopleRank to deal with this issue. Such extention is a general framework that could be integrated to most of social forwarding algorithms in order to operate in large scale networks.

13.7 Multi-Communities Social Forwarding

Forwarding in mobile ad hoc networks faces extreme challenges from potentially very large number of mobile nodes, very large areas, and limited communication resources such as bandwidth and energy. Such conditions make forwarding more challenging in large-scale networks. In the previous section, and particularly in Fig. 13.16, we have noticed that using social inputs in large-scale areas may present weaknesses. Our main guess is that in large-scale networks when multi-communities may exist, social prediction has its limitations and two people socially connected may not meet frequently because they could be long away from each others. Let us assume that communities represents cities in real world, people could be good friends (e.g., in Facebook); however, distances prevent these friends to meet physically.

13.7.1 Performances Per Community

A common property of social networks are cliques, circles of friends or acquaintances in which every member knows every other member. First of all, we verify the performance of PeopleRank in a single clique which we call *community*. In large-scale networks, people are by default regrouped in communities. For example, considering the Dartmouth data set (described in detail in the previous section), users are regrouped according to this two community classifications:

- **Geographic Classification**: Since the dartmouth campus area is roughly 1300$\times$ 1300 square meters, people attending the campus every day are mostly visiting the same places. Usually, these places are selected in the way that minimizes the walking distance. To capture this classification, we looked at splitting the Dartmouth campus area into regions (Northwest *NW*, Northeast *NE*, Southwest

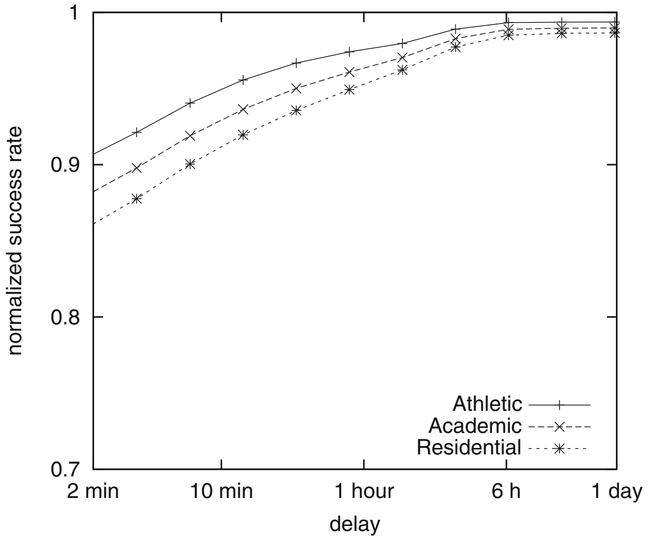


Fig. 13.17 Normalized success rate distribution PeopleRank relying on Activity-Based Classification

SW, and Southeast *SE*).[6] A node *i* belongs to a region *R* if it has been connected to more access points belonging to the corresponding region compared to the other regions.

- **Activity-Based Classification**: The Dartmouth College campus has over 160 buildings. Usually people visiting the campus are interested in few buildings. People could be classified relying on their activity interests. For example, the campus contains more than a dozen athletic facilities and fields. Most of them are located in the southeast corner of campus. Athletic people are more likely to meet each others and be classified in an athletic community. We consider people more connected to athletic building's access points as part of the athletic community. Similarly we define academic and residential communities.

Next, we plot the normalized success rate of PeopleRank according to the two community classifications described above: (1) Activity-Based Classification (in Fig. 13.17), and (2) Geographic Classification (in Fig. 13.18).

Obviously, the geographic classification leads to better PeopleRank performances. Indeed, PeopleRank achieves 92–97% of normalized success rate within 10 min timescales according to the geographic classification (in Fig. 13.18), and 90–94% within the same timescale according to the activity based classification in Fig. 13.17. That confirms that short distances (e.g., people leaving in the same neighborhood or region) leads to strong social ties, and relevant social classification.

[6]http://www.dartmouth.edu/~maps/campus/close-ups/index.html

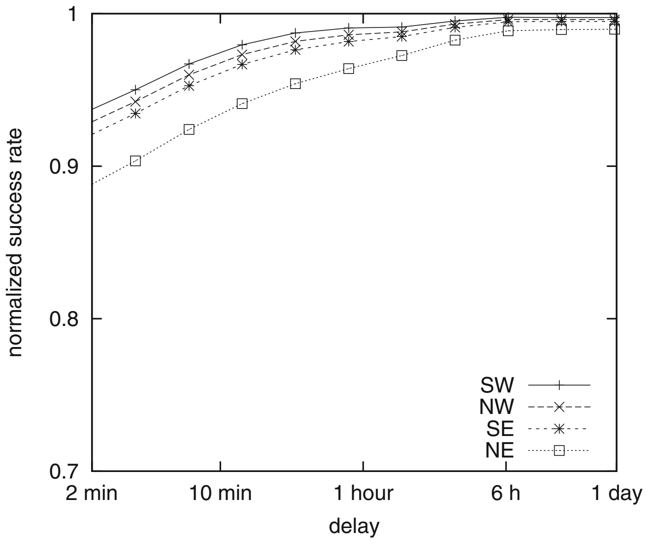


Fig. 13.18 Normalized success rate distribution of PeopleRank relying on Geographic Classification

Moreover, one could notice that, in Fig. 13.17, PeopleRank achieves better success rate performances among athletic users than others according to the activity based classification. Relying on the athletic community, PeopleRank outperforms its own success rate performances by roughly 3% and 5% within 10 min timescale compared to respectively the academic and the residential communities. Indeed, as described above, most of athletic buildings are located in the southeast corner of campus which leads to a combination of geographic and activity based classification.

To summarize, the previous results confirmed that PeopleRank achieved satisfactory performances within a single community. However, it was shown, in the previous section, that it suffers in very large communities. Next, we propose a framework extension which helps PeopleRank to deal with this issue.

13.7.2 The Framework Extension

We were motivated by satisfactory performances of social forwarding within single community, to propose a two step framework which could be easily integrated to most social forwarding algorithms in order to deal with large communities issues described above. This framework extension consists of:

- Step1: Social forwarding algorithms operate normally within the same community. Indeed, messages will be forwarded socially toward nodes which belong to the same community.

- Step2: Particular nodes will operate as a bridge and circulate the message to the other communities, and within these communities messages will be forwarded socially (Step1). Bridge nodes are characterized by their higher mobility. To capture mobility of nodes in the data sets, we assume that these bridge nodes belong to multiple communities (i.e., bridge nodes are moving from one community to another). We rank bridges according to the number of communities (BR) they belong to. For example, if we consider the geographic classification in the Dartmouth campus data set, bridges belonging to four communities are well ranked to bridges belonging to only two or three communities. We assume also that bridges carrying the message forward it to other bridges according to a non-decreasing BR (the bridge rank described above).

This two-step framework extension is easily integrated to most of social forwarding algorithms. Next, we evaluate an extended version of PeopleRank which integrates this framework.

13.7.3 Extended PeopleRank

We apply the previous framework extension to D-PeopleRank algorithm. Let us consider a node S which generates a message m to a destination node D. We assume no a priori knowledge of the destination's community. Message m will be forwarding to relay nodes $R_{i\,i=1..k}$ $(k \leq N-2)$ if and only if:

1. R_i belongs to the same community than R_{i-1}, and PeopleRank$(R_i) \geq$ PeopleRank (R_{i-1}).

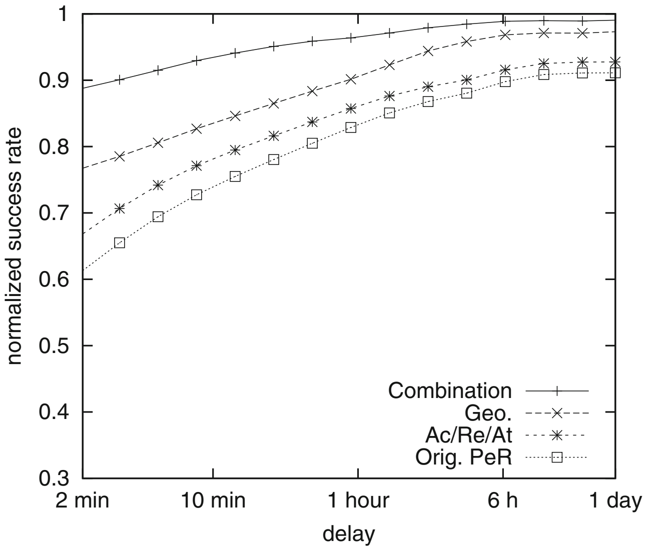


Fig. 13.19 Normalized success rate distribution of the Extended PeopleRank algorithm

2. R_i is a bridge node, and $\mathrm{BR}(R_i) \geq \mathrm{BR}(R_{i-1})$.

Figure 13.19 plots the normalized success rate of the extended D-PeopleRank algorithm in Dartmouth data set. One could notice that, extended PeopleRank outperforms the original one for all timescales (5–30% of success rate improvement). Furthermore, such improvement differs from one community classification to another. Geographic classification gives better performances than activity-based classification; indeed, the activity-based classification in the Dartmouth campus groups people belonging to specific buildings. However, theses buildings are not always geographically close to each others, and then, messages send from a specific building could take time to reach other members of the same community. Note that, combining the two definitions of communities leads to better success rate performance, and more than 30% improvement compared to the second definition (see Fig. 13.19).

13.8 Concluding Remarks

In this chapter, we have introduced a social opportunistic forwarding algorithm that uses PeopleRank metric to rank the relative "importance" of a node using a combination of social graph and contact graph information. We have developed centralized and distributed variants for the computation of PeopleRank . Analysis relies on real mobility traces and the social interactions between the corresponding participants. Evaluation and comparison to social-only and contact-only forwarding algorithms showed that PeopleRank achieves one of the best cost/delivery success rate trade-offs. PeopleRank algorithms achieve an end-to-end delay and a success rate close to those obtained by flooding while reducing the number of retransmission to less than 50% of the ones induced by flooding. Despite the fact that social information is used as a good predictor for human mobility, social forwarding algorithms in general, and PeopleRank in particular, suffer in large scale networks where the social graph consists of different communities.

We believe that PeopleRank algorithm is suitable to a wide range of mobile opportunistic networks for many reasons:

- PeopleRank is a distributed forwarding algorithm and is well suited for opportunistic networks; indeed, the lack of a central entity in such environment makes a global forwarding decisions challenging. In practice, nodes do not have a global view of the network, however they should rely on a local view to estimate future transfer opportunities.
- D-PeopleRank exchanges only a small amount of information: (1) the current PeopleRank value and (2) the number of social neighbors (as seen in Sect. 13.4).
- The damping factor used by PeopleRank is able to compensate for the mismatch between the social graph and the contact graph. To the best of our knowledge, this is the first work that proposes a mechanism to adjust the use of social information for forwarding.

- As opposed to contact-based algorithms which have to maintain updated contact properties between all nodes in the networks, PeopleRank is able to make forwarding decisions autonomously with local information only; it uses contact properties to augment the social information between neighbors. Indeed, the number of iterations performed by contact-based forwarding algorithms is proportional to the total size of the network. However, PeopleRank reduces this number and keeps it proportional to the social node degree.

References

1. Everett Anderson, Kevin Eustice, Shane Markstrum, Mark Hansen, and Peter Reiher. Mobile contagion: Simulation of infection and defense. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 80–87, Washington, DC, USA, 2005. IEEE Computer Society.
2. Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.
3. A. Chaintreau, A. Mtibaa, L. Massoulié, and C. Diot. The diameter of opportunistic mobile networks. In *Proceedings of ACM CoNext*, 2007.
4. Henri Dubois-Ferriere, Matthias Grossglauser, and Martin Vetterli. Age matters: efficient route discovery in mobile ad hoc networks using encounter ages. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 257–266, 2003.
5. Vijay Erramilli, Augustin Chaintreau, Mark Crovella, and Christophe Diot. Diversity of forwarding paths in pocket switched networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 161–174, New York, NY, USA, 2007. ACM.
6. Vijay Erramilli, Mark Crovella, Augustin Chaintreau, and Christophe Diot. Delegation forwarding. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 251–260, New York, NY, USA, 2008. ACM.
7. Theus Hossmann, Thrasyvoulos Spyropoulos, and Franck Legendre. Know thy neighbor: Towards optimal mapping of contacts to social graphs for dtn routing. In *Proceedings of IEEE INFOCOM*. IEEE, 2010.
8. Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: Social-based forwarding in delay tolerant networks. In *Proceedings of ACM MobiHoc '08*, New York, NY, USA, 2008. ACM. (also appeared as Cambridge University TR: UCAM-CL-TR-684).
9. Raj Jain, Dah-Ming Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *CoRR*, cs.NI/9809099, 1998.
10. S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proceedings of ACM SIGCOMM*, 2004.
11. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5): 604–632, 1999.
12. Massimo Marchiori. The quest for correct information on the web: hyper search engines. In *Selected papers from the sixth international conference on World Wide Web*, pages 1225–1235, Essex, UK, 1997. Elsevier Science Publishers Ltd.
13. Julie Morrison, Rainer Breitling, Desmond Higham, and David Gilbert. Generank: Using search engine technology for the analysis of microarray experiments. *BMC Bioinformatics*, 6(1):233, 2005.

14. Abderrahmen Mtibaa, Augustin Chaintreau, Jason LeBrun, Earl Oliver, Anna-Kaisa Pietilainen, and Christophe Diot. Are you moved by your social network application? In *WOSN'08: Proceedings of the first workshop on Online social networks*, pages 67–72, New York, NY, USA, 2008. ACM.
15. A-K Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. Mobiclique: Middleware for mobile social networking. In *WOSN'09: Proceedings of ACM SIGCOMM Workshop on Online Social Networks*, August 2009.
16. Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, New York, NY, USA, 2005. ACM.
17. Matteo Varvello, Ernst Biersack, and Christophe Diot. Is there life in second life? In *Proceedings of ACM CoNext*. ACM, 2007.
18. X. Zhang, J. Kurose, B. N. Levine, D. Towsley, and H. Zhang. Study of a bus-based disruption tolerant network: Mobility modeling and impact on routing. In *Proceedings of ACM MobiCom*, 2007.

Chapter 14
Discount Targeting in Online Social Networks Using Backpressure-Based Learning

Srinivas Shakkottai and Lei Ying

Abstract Online social networks are increasingly being seen as a means of obtaining awareness of user preferences. Such awareness could be used to target goods and services at them. We consider a general user model, wherein users could buy different numbers of goods at a marked and at a discounted price. Our first objective is to learn which users would be interested in a particular good. Second, we would like to know how much to discount these users such that the entire demand is realized, but not so much that profits are decreased. We develop algorithms for multihop forwarding of discount coupons over an online social network, in which users forward such coupons to each other in return for a reward. Coupling this idea with the implicit learning associated with backpressure routing (originally developed for multihop wireless networks), we show how to realize optimal revenue. Using simulations, we illustrate its superior performance as compared to random coupon forwarding on different social network topologies. We then propose a simpler heuristic algorithm and using simulations, and show that its performance approaches that of backpressure routing.

14.1 Introduction

The past few years have seen the rapid and global emergence of online social networks as a medium for community interaction [14,16,32,34,35]. Their meteoric adoption by millions of Internet users from all walks of life [7, 20], and the

S. Shakkottai (✉)
Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX
e-mail: sshakkot@tamu.edu

L. Ying
Department of Electrical and Computer Engineering, Iowa State University, Ames, IA
e-mail: leiying@iastate.edu

multifarious applications that have materialized on them suggest that they are likely to evolve into a platform for fundamental social change. Indeed, one can envision a future society in which communication, reputation, marketing, and the very molding of societal opinions transpire on an online social networking platform.

The continued success of the medium requires a sound economic foundation for sustainable growth – something that is missing in today's social networks. The method of choice to extract commercial value from an online social network is the advertising and sale of goods and services. However, by and large, the technique used to propagate messages is by simply flooding the network without regard to whether participants are actually interested in the products in question. Such *spam*, apart from wasting resources (e.g., bandwidth and storage space) worth billions of dollars every year [30], has the potential to kill the social network as a whole as it becomes difficult to distinguish relevant from irrelevant information.

Message flooding entirely ignores the fact that online social networks offer a unique perspective into the desires and tastes of the participants. In other words, it ignores the basic fact that makes these networks attractive to users in the first place – being able to interact with like-minded individuals with relatively little overhead. Further, since users of online social networks are constrained to *measurable and regulatable* interactions, there exists *the opportunity to implement systematically designed learning and targeting strategies at each individual user.*

Behavior of individuals in social networks has been studied qualitatively (e.g., [2, 8, 12, 15, 19, 27, 28]) to draw macroscopic inferences, and by using data mining and stochastic methods (e.g., [1, 3–5, 8, 11, 12, 17, 21, 37]) to draw microscopic inferences. However, these approaches employ centralized learning methods that are largely agnostic to both the *evolution* of individual preferences on the network, as well as the *applications* that the network can support. They also ignore the fact that *individuals possess information about their neighborhood*, and can promote sustainability if only they can be motivated to participate in applications. Thus, while there has been significant progress in learning methods, relatively little progress has been made in the joint optimization between end-user involvement, applications, and learning on online social networks.

In this chapter, we consider an alternate view of online social networks, one motivated by the idea that end-user involvement, learning, and applications must be strongly coupled to each other. We consider the issue of obtaining *preference awareness* in online social networks, which thrive on user-level contributions of both informational and computational resources. Thus, we have the social network provider that creates and runs the network, stores and advertisers that seek to identify the ideal individuals to target their products, and end-users who both seek information about goods and services that reflect their tastes, and possess information and computational resources that are valuable to the stores and advertisers. A successful sustainability model would ensure that incentives are aligned among all participants by appropriate remuneration. The following high-level goals seem suitable in such a scenario.

1. *Localized Information and Distributed Computation* Most learning algorithms employ centralized computation, which results in high computational complexity and a tremendous amount of information acquisition. A range of heuristics are utilized to simplify computation, which then results in suboptimal performance. The algorithms that we develop should be naturally distributed, so as to allow each individual to perform computations relatively independently. Participants should only require information pertaining to their immediate neighbors, which both reduces communication overhead, and allows a measure of privacy.
2. *Implicit Learning* Unless the purpose of the social network is to simply learn participant behavior, the objective of learning is to utilize perception of participants' preferences for some application. In other words, learning is not an end in itself, but needs to be closely coupled to the application in mind. The algorithms that we develop embrace this idea of implicit learning that is optimally targeted towards a specific application.
3. *Explicit Targeting* Learning algorithms merely obtain awareness of participants' preferences, they do not attempt to guide them in any particular direction. However, if the learning process is able to identify participants that are most influential in preference formation, it could potentially be used to direct the predilections of the majority (not just those targeted) toward some particular good or service by explicit targeting.

14.2 Optimization Decomposition and Backpressure

We are guided by ideas of *optimization decomposition* that effectively breaks down optimization problems into simple, distributed computations. Global objectives can then be achieved by properly linking local decisions based on neighborhood information and distributed computations. Optimization decomposition has been used in the design of communication networks [10, 18, 22–26, 29, 36, 38]. We briefly review some of the ideas presented in the work cited above, and illustrate how it satisfies the conditions that we are keen that our algorithms should possess.

Let us first consider a utility maximization framework for a network resource allocation problem. Suppose we have a set of traffic flows $\mathscr{F}$ and a set of links $\mathscr{L}$. Each flow f is defined by a traffic source s_f and a traffic destination d_f. A link from node m to node n is denoted by (m,n) and each link $(m,n) \in \mathscr{L}$ has a finite capacity $c_{m,n}$. The utility that the source obtains from transmitting data with rate x_f is denoted by $U_f(x_f)$. We assume that the utility function is continuously differentiable, non-decreasing and strictly concave. We will see later that similar ideas apply even when the utility function is linear. Our objective is to allocate capacity on the links such that the sum total utility is maximized. Thus, the problem that the network faces is the following optimization problem

$$\max_{x_f} \sum_{f \in \mathscr{F}} U_f(x_f), \tag{14.1}$$

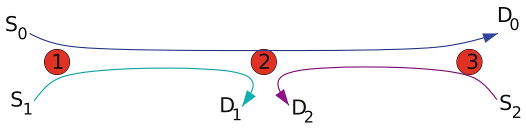


Fig. 14.1 A resource allocation problem. Satisfaction of constraints implies that packets are injected at a supportable rate. Routes need not be given a priori – each packet can find an appropriate route

subject to the constraints

$$\sum_{f:s_f=n} x_f + \sum_{m:(m,n)\in\mathscr{L}} y^f_{(m,n)} \le \sum_{i:(n,i)\in\mathscr{L}} y^f_{(n,i)},\ \forall n \tag{14.2}$$

$$\sum_f y^f_{m,n} \le c_{m,n} \forall (m,n) \in \mathscr{L}, \tag{14.3}$$

$$x_f \ge 0,\ \forall f \in \mathscr{F}, \tag{14.4}$$

$$y^f_{m,n} \ge 0,\ \forall (m,n), f, \tag{14.5}$$

where $y^f_{m,n}$ is the rate at which link (m,n) transmits packets belonging to flow f. Constraint (14.2) is the flow conservation constraint, i.e., all packets coming into node n should be served eventually. Constraint (14.3) indicates that the link capacities are finite, and the overall transmission rate of link (m,n) cannot exceed the link capacity. The constraints form a convex set, and since the utility functions are strictly concave, the problem has a unique solution. While there exist several different algorithms, both centralized and distributed, that are able to solve the above problem, we are interested in a particular method called "backpressure" [40] that is able to *learn* the routes to be taken, as well as the rates of packet injection into the network so as to maximize the total system throughput. We will illustrate the relevance of such an approach in social networks in the next section.

The resource allocation problem above has the objective of maximizing network utility subject to the constraint that the total transmission rate on each link is no more than the available capacity. We consider an illustration in Fig. 14.1.

There are 3 flows S_0, S_1, and S_2, with packet injection rates x_0, x_1 and x_2, respectively. Assume all have a capacity of 1 packet/time slot, and let $R^{(d)}_{ij}$ be the fraction of time that link (i,j) is dedicated to a flow destined for node d. The network utility maximization problem is

$$\max_{x,R\ge 0} \sum_{i=0}^{2} U_i(x_i)$$

$$
\begin{aligned}
x_0 &\le R_{12}^{(3)} \text{ (constraint at node 1 for destination node 3)}\\
x_1 &\le R_{12}^{(2)} \text{ (constraint at node 1 for destination node 2)}\\
R_{12}^{(3)} &\le R_{23}^{(3)} \text{ (constraint at node 2 for destination node 3)}\\
x_2 &\le R_{32}^{(2)} \text{ (constraint at node 3 for destination node 2)}\\
R_{12}^{(3)} + R_{12}^{(2)} &\le 1 \text{(capacity constraint for link (1,2)).}\\
R_{23}^{(3)} + R_{32}^{(2)} &\le 1 \text{(node capacity constraint for link (2,3)).}
\end{aligned}
$$

The Lagrange dual for the above problem can be written down with one Lagrange multiplier λ_{id} at each node i for each destination d (only the non-zero λ_{id} are shown below):

$$
\begin{aligned}
\max_{x,R\ge 0} \sum_{i=1}^{2} U_i(x_i) \,-\, \lambda_{13}\left(x_0 - R_{12}^{(3)}\right) - \lambda_{12}\left(x_1 - R_{12}^{(2)}\right)\\
-\, \lambda_{23}\left(R_{12}^{(3)} - R_{23}^{(3)}\right) - \lambda_{32}\left(x_2 - R_{32}^{(2)}\right)
\end{aligned}
\tag{14.6}
$$

subject to

$$
\begin{aligned}
R_{12}^{(3)} + R_{12}^{(2)} &\le 1\\
R_{23}^{(3)} + R_{32}^{(2)} &\le 1.
\end{aligned}
$$

Here, we see that the maximization over x and R can be divided into two distinct maximization problems:

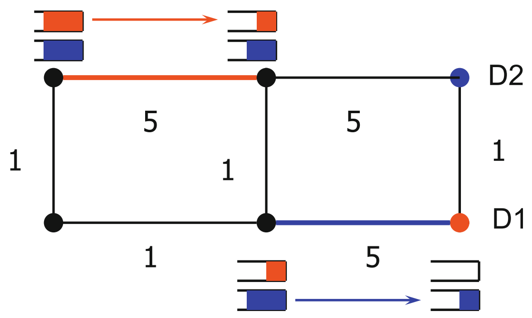
$$
\max_{x\ge 0} \sum_{i=0}^{2} U_i(x_i) - \lambda_{13}x_0 - \lambda_{12}x_1 - \lambda_{32}x_2 \tag{14.7}
$$

$$
\max_{R_{ij}^d \le 1} R_{12}^3\,(\lambda_{13} - \lambda_{23}) + R_{12}^2\,(\lambda_{12} - 0) + R_{23}^3\,(\lambda_{23} - 0) + R_{32}^2\,(\lambda_{32} - 0) \tag{14.8}
$$

The above is an illustration of a concept called *optimization decomposition,* wherein a global optimization naturally breaks up into different parts, each of which can be solved more easily.

Observe that the rate allocation problem (14.8) not only decides the fraction of time that each hop is scheduled, but also determines the fraction of time that a hop is dedicated to flows destined for a particular destination. In a general network topology, no pre-fixed routes are needed, routes *are implicitly determined by the rate allocated on each link for each destination.* The scheduling algorithm is a “max-weight” algorithm, with the weight for each hop for each flow as difference in the Lagrange multiplier at the transmitter and the Lagrange multiplier at the receiver

Fig. 14.2 Backpressure based routing and scheduling. Differences in queue length determine the packets sent on each link at each time



on that hop. This difference is called the *backpressure* due to the fact that a link is not scheduled if the Lagrange multiplier at the receiver is larger than the Lagrange multiplier at the transmitter of the hop. Hence, the rate allocation algorithm (14.8) is also called the backpressure algorithm.

Now, suppose that we have a system in which there is one buffer at each node devoted to each destination. Let the values of $\lambda_{i,d}$ indicate the queue length at node i for packets destined for node d. Then (14.8) above looks like a weighted maximization, with the difference in queue lengths (associated with a destination) between the two nodes constituting a hop acting as weights. This effect is clear in the first term of (14.8). The other terms relate to the final hop of the flows for which end in a sink with zero queue length, and we have explicitly indicated this with a 0. The maximization would result in the scheduling of hops (i, j) that have many packets backlogged at node i. In fact, there is queue lengths can be viewed as stochastic estimates of Lagrange multipliers. A detailed explanation of this connection can be found in [36].

We illustrate the backpressure idea in a wireless network in Fig. 14.2. Here, we have an additional scheduling constraint in that nodes may not transmit and receive simultaneously. Hence, the schedule at each time needs to be an independent set on the graph representing the wireless network. Notice that the source of each type of packet does not matter, since routes are not determined before hand. Simply scheduling the appropriate type of packet on each hop (subject to the scheduling constraint) results in throughput optimal routes at equilibrium, where the packets are classified according to their destinations.

Notice that under the backpressure algorithm, congestion at a particular node would be transferred closer and closer to the source (where ever it is), till the source feels the congestion due to packet buildup in its buffer. At this point we notice that from (14.7) that since the utility function is concave, the source would need to cut down its rate of packet generation in order to maximize system utility. Thus, the backpressure algorithm yields a natural means of regulating the rate of injection of packets into the system. The idea is illustrated in Fig. 14.3.

In summary, the main ideas that we take away from the network utility maximization approach that we wish to exploit in the social network context are the following:

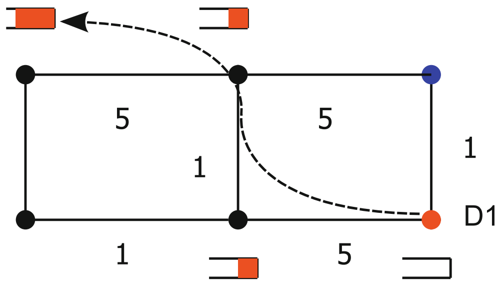


Fig. 14.3 Congestion effects are progressively transferred towards the source

- Implicit route determination using the backpressure algorithm.
- A natural source law that responds to the queue length one at the source.

In particular, we will consider the scenario where both users' preferences and the topology of the social network are not available at the coupon distributor. The coupon distributor needs to generate the coupons at proper rates and incentive the users to forward the coupons in a proper way so that it can extract the maximum profit from the network. Note that a user's preference (i.e., the number of goods a user would buy at a marked price and discounted price) can be viewed as the link capacities (marked and discounted) between the user and the product. Therefore, a back-pressure-type algorithm for coupon distribution can help the coupon distributor determine the optimal rates at which different types of coupons should be generated and the routes these coupons should be forwarded without knowing users' preferences and the network topology. However, as we will see in the next section, we need to considerably modify the algorithm in the context of social networks.

14.3 A Coupon Routing Problem in a Social Network

Consider goods and services that are consumed periodically (say on a weekly or monthly basis) such as movie tickets, car washes, fitness club visits and so on. Here, we could have a high displayed (marked) price that some consumers would be willing to pay. In order to extract maximum revenue, other consumers need to be subsidized to some extent by using discounts such as rebate coupons. In other words, discount coupons are used to create multiple tiers of prices for the same good or service. Two questions immediately arise, (1) which users should be given coupons? and (2) how many coupons should they be given? Further, the questions have to be answered in a system in which user preferences change over time.

Both questions are hard since the seller of the good is unlikely to be aware of the preferences of users, or possibly even of their existence. Even if the seller is aware of a user's interest, he must not give too many or too few discounts – too many would reduce profits and too few would mean that the entire demand would not be realized. As we saw in the Introduction, there are two classical methods of offering such incentives. The first is to flood communities of users in the hope that some of them would use the coupons. Here, the idea is to pre-identify communities that are not likely to buy the good without discounts. If identification is incorrect, either the users would not use the coupons, or the wrong set of users would be discounted. The second is to rely on self-identification of interested individuals. Here, the store gets the users to sign up for coupons, and then judiciously sends them coupons. Such a scheme would work only on users who do identify themselves to the store, and might not realize the entire demand.

Both the above solutions ignore the fact that users could belong to an online social network, and hence could obtain coupons by interacting with his or her friends. Thus, users could forward coupons from one to the next in a multihop fashion across the online social network. If a user is interested in the good that the coupon represents, he or she could use it. Otherwise, the user could forward it onwards. Allowing for coupon forwarding implies that the two questions raised have to be modified slightly: (1) given that a user has a coupon that he does not want, which friend should he forward it to and why? and (2) what rate should coupons be injected into the system? Hence, we need to design a signaling scheme that incentivizes users to somehow learn the preferences of users in such a way that the profits of the store are maximized. An example of such a system in practice is *mGinger* [31] that acts as a multihop advertising and discount distribution system using SMS messages, with rewards being paid in a pyramidal fashion. The motivation for multihop coupon distribution is that since user preferences change with time, and new products are continuously introduced, it is impossible for any store to be aware of all its potential customers. Hence, a system must *learn* user preferences, which then change after a while.

In this chapter, we develop *implicit distributed learning* schemes based on ideas of backpressure [40] that has been used as a throughput optimal scheme for packet routing in multihop wireless networks [9, 13, 33, 39]. We assume that the capacity for consumption of a good i by user j can be divided naturally into two parts – one at at the marked ("high") price $\hat{x}_j^{ih}$, and one at the discounted ("low") price $\hat{x}_j^{il}$. We assume that these values are fixed for some duration of time, and so can be learned. The number of coupons given to the user must be carefully regulated; if it is larger than $\hat{x}_j^{il}$, the store loses profits due to excessive discounting, while if it is less than $\hat{x}_j^{il}$, the entire demand is not realized. We combine ideas of self-identification by users and directed flooding through backpressure to achieve an optimal solution that realizes the entire demand by injecting the right number of coupons, and maximizes profit by ensuring that the users receive coupons at the optimal rate.

We then use *optimization decomposition* techniques in Sect. 14.6 to develop a coupon distribution scheme consisting of three entities: (1) a store at which goods may be purchased, (2) users connected by an online social network, and (3) a coupon source (or sources). The behavior of these entities is as follows:

- A store sells goods i at a marked price p^i, which it discounts to a price q^i upon presentation of a coupon. The store assigns a "goodness value" to each user j that makes a purchase from it. This value determines the probability with which neighbors of j are rewarded for forwarding a coupon to j.[1] However, all the other users (non-neighbors of j) that are involved in the forwarding path are guaranteed a reward. This artifice enables locality of information, as we show later. In other words, although the discount carried by each coupon is identical, the reward for forwarding coupons to each user j is not. The store uses a simple *up-down controller* to determine the reward probability for forwarding, based on the number of goods purchased by user j.

- All users in the system maintain a count of the number of coupons of each kind that their neighbors possess via communication over the social network. Users also maintain a count of the number of unrewarded coupons associated with their neighbors by polling the relevant store. We refer to the sum of these two as the *effective coupons*. Coupons can be transferred among users in a multihop fashion, and users are incentivized to forward coupons in the direction of lowest *effective pressure*, i.e., to a neighbor who has the smallest number of effective coupons. This controller is similar in nature to a backpressure controller.
- Finally, the coupon source generates coupons of different kinds (corresponding to different goods), and sends them to users that have identified themselves as interested in receiving particular kinds of coupons. The source chooses to send coupons using a *threshold controller*, which generates new coupons when the effective pressure is less than a certain threshold.

We prove that the system using this *backpressure-based* coupon distribution evolves with time to attain the maximum profits by ensuring that each potential consumer obtains exactly the right number of coupons. The system is distributed and each user only requires information associated with his or her neighbors. Thus, it succeeds in achieving light-weight learning framework, in which exploring for user capacity and exploiting existing capacity go hand-in-hand.

We then consider a simpler heuristic algorithm in Sect. 14.7, that is based on the delay in obtaining rewards. This *delay-based* algorithm does not require information exchange between users. At any time, users simply forward coupons to that neighbor for whom the delay experienced between forwarding a coupon and obtaining a reward for that coupon is the smallest. This algorithm inherently captures the idea of backpressure, although it is at a coarse level.

Our final scheme is even simpler, and consists of *random* coupon distribution. Here, each user randomly forwards coupons to its neighbors in the hope of finding correct paths. This system does not learn user preferences. We use this algorithm to test the efficacy of our other algorithms.

We simulate the distribution schemes in Sect. 14.8 on different topologies to compare their performance. We show that the backpressure-based scheme achieves near-optimal revenue, while the delay-based scheme performs acceptably

[1]Throughout this chapter, we use the word *reward* to denote remuneration for *forwarding* coupons, and the word *discount* to denote remuneration for redeeming coupons (when purchasing goods) at a store.

well. Further, both schemes significantly outperform the randomized scheme, thus making a strong case for backpressure based targeted coupon delivery in online social networks.

14.4 System Model

Network model: We consider an online social network structure as shown in Fig. 14.4. Denote by $\mathscr{N}$ the set of nodes and $\mathscr{L}$ the set of links. There are three different node types – a coupon distributor, users, and a store – in the network. The links represent social connections. A link from a coupon source to a user represents the idea that the user has registered with the source to receive its coupon periodically. A link from a user to a product means that the user buys that product periodically. The links between users are assumed to be bidirectional, and represent friendship between the connected users. In this chapter, we assume that the coupon sources and the store are managed by the same entity. We use s to denote the store and d to denote the coupon distributor. We define $\mathscr{S}$ to be the set of products and $\mathscr{B}_i$ is the set of users who will buy product i.

We consider a synchronized slotted-time system. We define μ_j to be the coupon transmission capacity of node j, which is the maximum number of coupons user j can send out in one time-step. We also impose the constraint that a user can buy a discounted product only if a coupon is presented.

Two-capacity model for user demands: We assume that users naturally have a maximum number of goods that they would buy at the marked price, $\hat{x}_j^{ih}$, and number of goods that users would buy at the discounted price, $\hat{x}_j^{il}$. Note that either of these quantities could be zero. We further define $b_j^i = \hat{x}_j^{ih} + \hat{x}_j^{il}$. These values can be thought of as the *capacities* associated with a user. We consider two different time scales in this chapter. The small time scale t is the one in which purchases are made and coupons are delivered. The large time-scale, consisting of T small time slots, is

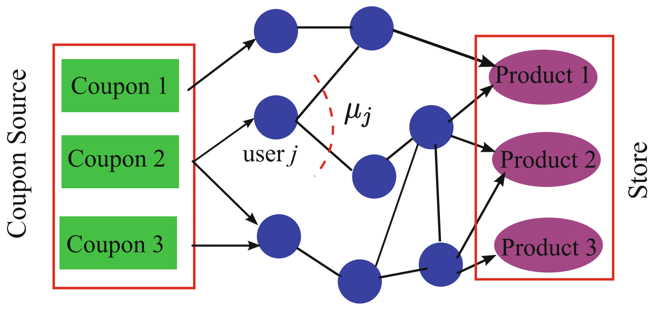


Fig. 14.4 A coupon distribution system. Coupons originate at the coupon source, follow a multihop forwarding path, and are finally spent at a store

the buying interval after which the customers start afresh. Since the users have the incentive to buy a product with a low price, we assume that the $\hat{x}_j^{ih}$, associated with the high price goods could be used to buy low priced goods as well. Specifically, during each large time scale, if the user were given no more than $b_j^i T$ coupons, he would use them all and buy $b_j^i T$ goods.

Note that if a user were given more than $\hat{x}_j^{il} T$ coupons, the store would not extract the maximum extractable revenue. If he were given less than $\hat{x}_j^{il} T$ coupons, he would not buy enough discounted goods, which reduces the profit of the store as well. A store that is unaware of these two capacities needs to probe customers in order to find their true potential, and neither supply too few or too many coupons. In what follows, we present a distributed solution that automatically explores for and attains the capacity of users, thus achieving the profit maximizing solution. The notations used in this chapter are summerized in the following table.

$\mathscr{N}$	Set of nodes
$\mathscr{L}$	Set of links
s	Store
d	Coupon distributor
$\mathscr{S}$	Set of products
$\mathscr{B}_i$	Set of users buying product i
μ_j	Transmission capacity of node j
p^i	Market price of produce i
q^i	Discounted price of product i
$\hat{x}_j^{ih}$	Number of type i goods that user j would buy at the marked price
$\hat{x}_j^{il}$	Number of type I goods that user j would buy at the discounted price
b_j^i	$\hat{x}_j^{ih} + \hat{x}_j^{il}$
$y_{(m,n)}^i$	The average number of valid type-i coupons sent from user M to user n in a time slot
γ_j^i	Target coupon usage rate at the large time scale
$Q_j^i[t]$	Number of type i coupons user j has at a finer time-step t
$\tilde{Q}_j^i[t]$	Number of unrewarded type i coupons corresponding to customer j
α^i	Reward for a type i coupon
$\Theta^i[t]$	Number of type i coupon generated at time slot t by the coupon distributor

14.5 Profit Maximization

Consider the profit made by the store. We say a coupon is *valid* if it is eventually used to purchase a product. We denote by $y_{(m,n)}^i$ the average number of valid type-i coupons sent from user m to user n in a time slot. The profit the store extracts from user j is

$$q^i y_{(j,s)}^i + p^i \min\left\{\hat{x}_j^{ih}, b_j^i - y_{(j,s)}^i\right\}.$$

Thus, the maximum profit the store can extract is defined by the following optimization problem:

OPT 1:

$$\max \sum_{i\in\mathscr{S}} \sum_{j\in\mathscr{B}_i} \left(q^i y^i_{(j,s)} + p^i \min\left\{ \hat{x}^{ih}_j , b^i_j - y^i_{(j,s)} \right\} \right) \tag{14.9}$$

$$s.t. \qquad \sum_{i\in\mathscr{S}} \sum_{j:(m,j)\in\mathscr{L}, j\neq s} y^i_{(m,j)} \le \mu_m, \forall m \in \mathscr{N} \tag{14.10}$$

$$\sum_{j:(m,j)\in\mathscr{L}} y^i_{(m,j)} = \sum_{n:(n,m)\in\mathscr{L}} y^i_{(n,m)} \forall m \in \mathscr{N} \tag{14.11}$$

$$y^i_{(m,j)} = 0 \text{ if } m \in \mathscr{B}_i \text{ and } j \neq s, \tag{14.12}$$

where (14.10) is the capacity constraint, which indicates node m cannot send more than μ_m coupons in a time-slot, (14.11) is the flow-conservation constraint for the coupons, and (14.12) indicates that user j will not forward type-i coupons to his/her neighbors if he/she uses type-i coupons.

To extract the maximum revenue, we need to distribute coupons to the users. There are two difficulties in distributing the right number of coupons to the users:

(1) The optimal $\left(\hat{x}^{ih}_j , \hat{x}^{il}_j\right)$ is unknown at the store, and needs to be identified.

(2) Since all users interested in a product may not be registered to directly receive coupons, they might need to receive such coupons via the social network. The store cannot directly control the number of coupons sent to such users.

To tackle these two difficulties we develop a two time-scale coupon distribution scheme in the next section.

14.6 Coupon Distribution

In this section, we develop an implicit distributed learning scheme based the idea of backpressure routing/scheduling [40]. Our algorithm consists of two control loops that operate at the small time scale and the large time scale. The purpose of the control loops is as follows:

1. **Choice of Coupon Forwarding Reward Rate:** At the large time scale, each store must adapt the target rate γ^i_j for the next buying interval using the information gathered about the customers' preferences over the past intervals. In our algorithm, γ^i_j is an estimate of $\hat{x}^{il}_j$. As discussed in Sect. 14.4, if γ^i_j is set too low, customer j may not purchase all the goods that he potentially could, and if γ^i_j is too high, customer j may be being discounted excessively and the store is not extracting the maximum extractable revenue.

2. **Coupon Routing at Target Rate:** At the small time scale a store assigns to a rate of coupon delivery γ_j^i to each product i and each customer j that purchases goods from it. The purpose of this control loop is to ensure that the customer would indeed receive discount coupons at this target rate. Mathematically, we will guarantee that the coupon distribution algorithm solves the following optimization problem:

   OPT 2:

$$\max \sum_{i \in \mathscr{S}} q^i \left(\sum_{j \in \mathscr{B}_i} y^i_{(j,s)} \right) \tag{14.13}$$

$$s.t. \quad \sum_{i \in \mathscr{S}} \sum_{j:(m,j)\in\mathscr{L}, j \neq s_i} y^i_{(m,j)} \leq \mu_m, \forall m \in \mathscr{N} \tag{14.14}$$

$$\sum_{j:(m,j)\in\mathscr{L}} y^i_{(m,j)} = \sum_{n:(n,m)\in\mathscr{L}} y^i_{(n,m)} \forall m \in \mathscr{N} \tag{14.15}$$

$$y^i_{(j,s)} \leq \gamma^i_j \quad \forall j, i \tag{14.16}$$

$$y^i_{(m,j)} = 0 \text{ if } m \in \mathscr{B}_i \text{ and } j \neq s. \tag{14.17}$$

To show the correctness of the proposed algorithm, we first need the following straightforward lemma.

Lemma 14.1. *Given that $\gamma_j^i = \hat{x}_j^{il}$ for all i and j, OPT 1 is equivalent to OPT 2.*

Proof. First, it is easy to verify that the optimal solution satisfies $y^i_{(j,s)} \leq x^{il}_j$ because otherwise, the store can extract more profit by reducing the number of coupons sent to user j. Based on that, OPT 1 can be re-written as

$$\textbf{OPT1}: \qquad \max \sum_{i \in \mathscr{S}} \sum_{j \in \mathscr{B}_i} \left(q^i y^i_{(j,s)} + p^i \hat{x}^{ih}_j \right)$$

$$s.t. \quad \sum_{i \in \mathscr{S}} \sum_{j:(m,j)\in\mathscr{L}, j \neq s_i} y^i_{(m,j)} \leq \mu_m, \forall m \in \mathscr{N}$$

$$\sum_{j:(m,j)\in\mathscr{L}} y^i_{(m,j)} = \sum_{n:(n,m)\in\mathscr{L}} y^i_{(n,m)} \forall m \in \mathscr{N}$$

$$y^i_{(j,s)} \leq x^{il}_j$$

$$y^i_{(m,j)} = 0 \text{ if } m \in \mathscr{B}_i \text{ and } j \neq s.$$

Since $\hat{x}^{ih}_j$ are constants, the objective is equivalent to

$$\max \sum_{i \in \mathscr{S}} \sum_{j \in \mathscr{B}_i} \left(q^i y^i_{(j,s)} \right) = \max \sum_{i \in \mathscr{S}} q^i \left(\sum_{j \in \mathscr{B}_i} y^i_{(j,s)} \right).$$

Thus, the lemma holds.

Next, we develop a distributed coupon routing algorithm that solves OPT 2.

14.6.1 Small Time Scale Control: Backpressure Coupon Routing

We first introduce the coupon management scheme which consists of three parts: (1) each user maintains a per-product queue, and monitors the lengths of the queues; (2) store rewards the neighbors that forwarded type i coupons used by each customer j at a rate γ_j^i, and monitors the number of unrewarded coupons at each customer[2]; (3) coupon distributor i monitors the number of coupons she has not yet sent out, and generates additional coupons based on this value.

A1: Coupon Management:

(1) Per-product queues are maintained at each user, and the number of type i coupons user j has at a finer time-step t is denoted by $Q_j^i[t]$. Thus, the dynamics of $Q_j^i[t]$ is as follows: If $j \notin \mathscr{B}_i$, then

$$Q_j^i[t+1] = \left(Q_j^i[t] + \sum_{m:(m,j)\in\mathscr{L}} y_{(m,j)}^i[t] - \sum_{n:(j,n)\in\mathscr{L}} y_{(j,n)}^i[t] \right)^+ ;$$

otherwise

$$Q_j^i[t+1] = Q_j^i[t] + \sum_{m:(m,j)\in\mathscr{L}} y_{(m,j)}^i[t] - y_{(j,s)}^i[t],$$

where

$$y_{(j,s)}^i[t] = \min\left\{ Q_j^i[t] + \sum_{m:(m,j)\in\mathscr{L}} y_{(m,j)}^i[t], \left(b_j^i T - \sum_{\tau=0}^{t-1} y_{(j,s)}^i[\tau] \right)^+ \right\},$$

i.e., user j will use up all available coupons unless she has already bought enough ($b_j^i T$) products.

(2) Store maintains a queue for unrewarded coupons corresponding to each of product i and its customers j. We may think of these as *virtual coupons* that are used to maintain a pressure on j's neighbors. Note that it is only the *neighbors* of j that are not rewarded for forwarding these coupons, the rest of the users involved in forwarding coupons would be guaranteed a reward (and, of course, j has already redeemed these coupons for a discount). The rewarding scheme will be describe in a detail later. This measure ensures that pressure against forwarding coupons to a particular customer is maintained adjacent to the customer, and not at arbitrary queues in the network. Denote by $\tilde{Q}_j^i[t]$ the number

[2]Recall that these are coupons that have been redeemed for a discount by j, but the neighbors of j who forwarded these coupons have not been rewarded.

of such unrewarded coupons corresponding to customer j. This is essentially a "virtual queue" that will be used to enforce constraint (14.16). We have

$$\tilde{Q}^i_j[t+1] = \left(\tilde{Q}^i_j[t] + y^i_{(j,s)}[t] - \gamma^i_j\right)^+,$$

where γ^i_j is the coupon forwarding reward rate for neighbors of customer j.

(3) Coupon distributor d maintains a separate queue for each type of coupons that have not been sent out. The length of the queue is denoted by $\tilde{Q}^i_d[t]$ for each product i. We have

$$Q^i_d[t+1] = \left(Q^i_d[t] + \Theta^i[t] - \sum_{j:(d_i,j)\in\mathscr{L}} y^i_{(d,j)}[t]\right)^+,$$

where $\Theta^i[t]$ is the number of new type i coupons generated by coupon distributor i at time t.

(4) We also assume that when user j receives a type i coupon such that $j \notin \mathscr{B}_i$, user j will insert her identity and the coupon queue length $Q^i_j[t]$ in the coupon before sending the coupon to her neighbor. This information allows the store to reconstruct path and reward the coupon relays based on $Q^i_j[t]$.

In our system, the stores need to reward coupon forwarding in order to motivate users to forward coupons to their friends. The efficiency of a coupon distribution scheme is determined by: (1) the incentive scheme that the store use and (2) the users' decisions under the incentive scheme. Next, we propose a coupon rewarding scheme, under which a rational user will distribute the coupons according to a backpressure policy. The optimality of the coupon distribution scheme will be proved in Theorem 14.1.

A2: Reward Scheme for Coupon Forwarding: Store rewards the users involved in forwarding each used type i coupon with a total of α^i dollars. We assume that α^i is fixed and is such that the store still makes a profit, i.e., we do not optimize over α^i. Consider a specific coupon associated with product i, and assume $\mathscr{R}$ is the path (consisting of the sequence of transmissions used to distribute the coupon) over which the coupon was transferred. Then node m gets a reward

$$\left(Q^i_m - Q^i_{n:(m,n)\in\mathscr{R}}\right)^+ \frac{\alpha^i}{\sum_{l\in\mathscr{R}}\left(Q^i_{s(l)} - Q^i_{r(l)}\right)^+}, \tag{14.18}$$

where l is a link on path $\mathscr{R}$, $s(l)$ is the sender, and $r(l)$ is the receiver. Note that this queue length information is inserted by the users before they forward the coupons to their neighbors. Furthermore, note that the amount of reward user m obtains is proportional to the queue difference. The idea is to motivate user m to send her coupon to a neighbor who has the least number of coupons and hence is most likely the one who needs the coupon. Under this scheme, the user has the motivation to follow the backpressure-like coupon distribution scheme.

A3: User Behavior:

(1) First, if node j is interested in buying product i, then user j uses all available type i coupons up to her purchasing limit b^i_j. Thus, at finer time-step t, user j purchases $y^i_{(j,s)}[t]$ products with coupons such that

$$y^i_{(j,s)}[t] = \min\left\{Q^i_j[t] + \sum_{m:(m,j)\in\mathscr{L}} y^i_{(m,j)}[t], \left(b^i_j T - \sum_{\tau=0}^{t} y^i_{(j,s)}[\tau]\right)^+\right\}.$$

We assume that user j never forwards type-i coupons to her neighbors if user j buys product i.

(2) If user j is not a customer buying product i, then user j needs to distribute type i coupons to her neighbors. We assume that at the beginning of finer time-step t, user j requests $Q^i_m[t]$ if user m is her neighbor, and also polls the store to obtain $\tilde{Q}^i_m[t]$. Since the amount of coupon forwarding reward is determined by the queue difference as described in (14.18), user j selects a coupon type i^* and neighbor m^* such that

$$(i^*, m^*) \in$$
$$\arg\max_{(i,m)\in\mathscr{L}} \left(Q^i_j[t] + \tilde{Q}^i_j[t] - Q^i_m[t] - \tilde{Q}^i_m[t]\right),$$

and transfers $\min\left\{\mu_j, Q^{i^*}_j[t]\right\}$ of type i^* coupons to node m^*.

Note that $\tilde{Q}^i_m[t]$ is the number of coupons used by user m but have not been rewarded yet, so $\tilde{Q}^i_m[t] = 0$ if user m is not a customer buying product i. A store maintains this unrewarded coupon queue to prevent a customer receiving too many coupons. When user j uses too many coupons, the unrewarded coupon queue becomes large. After a neighbor of user j finds a large $\tilde{Q}^i_j[t]$, the neighbor realizes that user j has received too many coupons and the store might not reward him for forwarding coupons to user j. Then the neighbor will stop forwarding more coupons to user j.

A4: Coupon Generation Scheme: The coupon distributor needs to decide the number of coupons to inject into the network. We assume that coupon distributor generates μ_d type-i coupons when $Q^i_d[t] \le Q_\text{T} q^i$, and zero type-$i$ coupon otherwise. Here, Q_T is a constant threshold value. In other words, $\Theta^i[t] = \mu_d$ if $Q^i_d[t] \le Q_\text{T} q^i$, and $\Theta^i[t] = 0$ otherwise.

In the following theorem, we analyze the performance of the backpressure coupon routing, and prove that

Theorem 14.1. *Assume that $\gamma^i_j \le b^i_j$ for all i and j. Under the coupon management, coupon rewarding and generating scheme, and user behavior defined above, we have*

$$\lim_{Q_\text{T}\to\infty} \lim_{T\to\infty} \frac{\sum_{t=1}^{T} \Theta^i[t]}{T} = \left(\sum_{j\in\mathscr{B}_i} \check{y}^i_{(j,s)}\right), \tag{14.19}$$

and

$$\lim_{Q_{\mathrm{T}}\to\infty}\lim_{T\to\infty}\frac{\sum_{t=1}^{T} y^i_{(j,s)}[t]}{T} = \check{y}^i_{(j,s)}, \tag{14.20}$$

where $\check{\mathbf{y}}$ *is the optimal solution of OPT 2.*

Proof. The analysis follows the Lyapunov drift used in [13, 33, 39]. Define $\mathbf{Q}[t] = \{Q^i_j[t], \tilde{Q}^i_j[t]\}_{j\in\mathscr{N}, i\in\mathscr{S}}$. It is easy to verify that $\mathbf{Q}[t]$ is a Markov chain. Further, given $\gamma^i_j \le b^i_j$ for all j and i, we can obtain that for any $j \in \mathscr{B}_i$, the following holds

$$Q^i_j[t+1] + \tilde{Q}^i_j[t+1] = \left(Q^i_j[t] + \tilde{Q}^i_j[t] + \sum_{m:(m,j)\in\mathscr{L}} y^i_{(m,j)}[t] - \gamma^i_j\right)^+ .$$

Next, consider a Lyapunov function such that

$$V[t] = \frac{1}{2}\sum_{i\in\mathscr{S}}\sum_{j\in\mathscr{B}_i}\left(Q^i_j[t] + \tilde{Q}^i_j[t]\right)^2 .$$

Following the analysis in [13], it can be shown that there exists $B > 0$, independent of $\mathbf{Q}[t]$, such that

$$\begin{aligned}\mathbf{E}\left[V[t+1] - V[t] | \mathbf{Q}[t]\right] &\le B + \sum_{i\in\mathscr{S}} Q^i_d[t]\mathbf{E}\left[\Theta^i[t] - \sum_{j:(d,j)\in\mathscr{L}} y^i_{(d,j)}[t]\middle|\mathbf{Q}[t]\right]\\ &+ \sum_{i\in\mathscr{S}, j\neq d}\left(Q^i_j[t] + \tilde{Q}^i_j[t]\right)\mathbf{E}\left[\left(\sum_{m:(m,j)\in\mathscr{L}} y^i_{(m,j)}[t] - \sum_{n:(j,n)\in\mathscr{L}} y^i_{(n,j)}[t]\right)\middle|\mathbf{Q}[t]\right].\end{aligned}$$

Letting $\check{\Theta}^i = \sum_{j:(d,j)\in\mathscr{L}} \check{y}^i_{(j,d)}$, where $\check{y}^i_{(j,d)}$ is the optimal solution to OPT 2, we further obtain that

$$\begin{aligned}\mathbf{E}\left[V[t+1] - V[t] | \mathbf{Q}[t]\right] &= B_1 + \sum_{i\in\mathscr{S}}\left(Q^i_d[t] - Q_{\mathrm{T}} q^i\right)\mathbf{E}\left[\Theta^i[t] - \check{\Theta}^i \middle| \mathbf{Q}[t]\right]\\ &+ \sum_{i\in\mathscr{S}} Q^i_d[t]\check{\Theta}^i - \sum_j \sum_{m:(j,m)\in\mathscr{L}}\sum_{i\in\mathscr{S}} y^i_{(j,m)}[t] \times \left(Q^i_j[t] + \tilde{Q}^i_j[t] - Q^i_m[t] - \tilde{Q}^i_m[t]\right)\end{aligned}$$

Next, we have that

$$\sum_{(j,m)\in\mathscr{L}}\sum_{i\in\mathscr{S}} y^i_{(j,m)}[t]\left(Q^i_j[t]+\tilde{Q}^i_j[t]-Q^i_m[t]-\tilde{Q}^i_m[t]\right)$$
$$\geq_{(a)} \sum_{(j,m)\in\mathscr{L}}\sum_{i\in\mathscr{S}} \check{y}^i_{(j,m)}\left(Q^i_j[t]+\tilde{Q}^i_j[t]-Q^i_m[t]-\tilde{Q}^i_m[t]\right) \geq \sum_{i\in\mathscr{S}} Q^i_d[t]\check{\Theta}^i,$$

where inequality (a) holds due to backpressure routing, and

$$\left(Q^i_{d_i}[t]-Q_{\mathrm{T}}q^i\right)\left(\Theta^i[t]-\check{\Theta}^i\right)\leq 0$$

holds according to the definition of the rate controller. Thus, according to the Foster's criterion, we can conclude that $\mathbf{Q}[t]$ is positive recurrent, which implies that $\lim_{t\to\infty}\mathbf{E}[V[t]]<\infty$.

Since $\mathbf{Q}[t]$ is positive recurrent, we further have

$$\frac{1}{T}\left(\mathbf{E}\left[V[T]\right]-\mathbf{E}\left[V[0]\right]\right)$$
$$=\frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{E}\left[\mathbf{E}[V[t]|\mathbf{Q}[t]]\right]-\mathbf{E}\left[\mathbf{E}\left[V[t-1]|\mathbf{Q}[t-1]\right]\right]\right)$$
$$\leq B_1+\sum_i Q_{\mathrm{T}}q^i\left(\frac{\sum_{t=0}^{T}\Theta^i[t]}{T}-\check{\Theta}^i\right),$$

which implies that

$$\sum_i q^i\left(\check{\Theta}^i-\frac{\sum_{t=0}^{T}\Theta^i[t]}{T}\right)\leq\frac{B}{Q_{\mathrm{T}}}+\frac{\mathbf{E}\left[V[T]-V[0]\right]}{T}.$$

Note that

$$\sum_i q^i\left(\check{\Theta}^i-\frac{\sum_{t=0}^{T}\Theta^i[t]}{T}\right)\geq 0,$$

because the network is stable and the $\check{\Theta}^i$ is the optimal solution to OPT 2. Thus, we have that

$$0\leq\sum_i q^i\left(\check{\Theta}^i-\lim_{T\to\infty}\frac{\sum_{t=0}^{T}\Theta^i[t]}{T}\right)\leq\frac{B}{Q_{\mathrm{T}}},$$

which leads to equality (14.19). Furthermore, since the queues are stable, so when $T\to\infty$, almost all coupons are consumed, which implies that equality (14.20).

The algorithm is similar to that proposed in [33]. Note that although the theorem is an asymptotic result, the algorithm itself works for any value of T. A finite value of T may result in a sub-optimal solution. In our simulations, we choose $T = 300$ and the final coupon allocation is very close to the optimal.

14.6.2 Large Time Scale Control: Coupon Rate Selection

We assume that the algorithm for coupon delivery at the small time scale converges quickly to the target rate, and now consider how to choose this target rate. Recall that our means of implementing coupon delivery at rate γ_j^i is to reward the neighbors of a customer j for forwarding coupons to j at rate γ_j^i. In this section all dynamics take place at the large time. Thus, we have the sequence of target rates $\gamma_j^i[0], \cdots, \gamma_j^i[k-1], \gamma_j^i[k], \gamma_j^i[k+1], \cdots$, and the large time scale algorithm needs to guarantee that $\lim_{k\to\infty} \hat{\gamma}_j^i[k] = \hat{x}_j^{il}$.

Denote by $z_j^{ih}[k]$ and $z_j^{il}[k]$ the number of goods i that user j buys from store in the interval $[k, k+1]$ at the marked price and the discounted price, respectively. Let the total number of goods purchased in the interval $[k, k+1]$ be denoted $z_j^i[k] = z_j^{ih}[k] + z_j^{il}[k]$. Further, denote the difference in purchases made by user j over intervals $[k, k+1]$ and $[k-1, k]$ by $\Delta z_j^i[k] = z_j^i[k] - z_j^i[k-1]$ corresponding to a difference in the coupon delivery rate $\Delta\gamma_j^i[k] = \gamma_j^i[k] - \gamma_j^i[k-1]$. We first intuitively understand the four possibilities associated with $\Delta\gamma_j^i[k], \Delta x_j^i[k]$ (assuming that $\Delta\gamma_j^i[k]$ is small):

- $\Delta\gamma_j^i[k] < 0$ and $\Delta z_j^i[k] = 0$: This implies that $\gamma_j^i[k] \geq \hat{x}_j^{il}$ and the user is receiving more coupons than he can use. We need to ensure $\gamma_j^i[k+1] < \gamma_j^i[k]$.
- $\Delta\gamma_j^i[k] < 0$ and $\Delta z_j^i[k] < 0$: This implies that $\gamma_j^i[k] < \hat{x}_j^{il}$ and the user is not receiving enough coupons to realize the maximum possible number of purchases. We need to ensure $\gamma_j^i[k+1] > \gamma_j^i[k]$.
- $\Delta\gamma_j^i[k] > 0$ and $\Delta z_j^i[k] = 0$: This implies that $\gamma_j^i[k] \geq \hat{x}_j^{il}$ and the user is receiving more coupons than he can use. We need to ensure $\gamma_j^i[k+1] < \gamma_j^i[k]$.
- $\Delta\gamma_j^i[k] > 0$ and $\Delta z_j^i[k] > 0$: This implies that $\gamma_j^i[k] < \hat{x}_j^{il}$ and the user is not receiving enough coupons to realize the maximum possible number of purchases. We need to ensure $\gamma_j^i[k+1] > \gamma_j^i[k]$.

Note that an increase in the coupon rate cannot cause a decrease in the number of purchases. A simple controller that takes into account all the four possible conditions is

$$\gamma_j^i[k+1] = (\gamma_j^i[k] + \delta)\chi_{\{\Delta\gamma_j^i[k]\Delta z_j^i[k]>0\}} + (\gamma_j^i[k] - \delta)\chi_{\{\Delta\gamma_j^i[k]\Delta z_j^i[k]=0\}}. \quad (14.21)$$

Here, $\delta > 0$ is a constant small amount by which we increase or decrease γ_j^i. We can now easily prove that the controller converges to within $\delta/2$ of the desired value of $\hat{\gamma}_j^i$.

Theorem 14.2. *Under the time scale separation assumption, using the controller (14.21) we have*

$$\lim_{k\to\infty} |\gamma_j^i[k] - \hat{x}_j^{il}| \leq \delta/2 \quad \forall\, i \in \mathscr{S},\ j \in \mathscr{R}.$$

Proof. We use a Lyapunov argument, with the Lyapunov function

$$J[k] = \left(\gamma_j^i[k] - \hat{\gamma}_j^i\right)^2.$$

Then we have

$$\begin{aligned} J[k+1] - J[k] = {} & (\gamma_j^i[k+1])^2 + (\hat{\gamma}_j^i)^2 - 2\hat{\gamma}_j^i\gamma_j^i[k+1] \\ & - (\gamma_j^i[k])^2 - (\hat{\gamma}_j^i)^2 + 2\hat{\gamma}_j^i\gamma_j^i[k] = \left(\gamma_j^i[k+1] - \gamma_j^i[k]\right)\left(\gamma_j^i[k+1]\right. \\ & \left. + \gamma_j^i[k] - 2\hat{\gamma}_j^i\right) \end{aligned}$$

We have two cases.

Case I: If $\Delta\gamma_j^i \Delta z_j^i[k] > 0$, i.e., $\gamma_j^i[k] < \hat{\gamma}_j^i$, we have from (14.21)

$$J[k+1] - J[k] = \delta(2(\gamma_j^i[k] - \hat{\gamma}) + \delta),$$

which is non-positive except in $\gamma_j^i[k] - \hat{\gamma}_j^i \in [-\delta/2, 0]$.

Case II: $\Delta\gamma_j^i \Delta z_j^i[k] = 0$, i.e., $\gamma_j^i[k] \geq \hat{\gamma}_j^i$, we have from (14.21)

$$J[k+1] - J[k] = -\delta(2(\gamma_j^i[k] - \hat{\gamma}) + \delta),$$

which is non-positive except in $\gamma_j^i[k] - \hat{\gamma}_j^i \in [0, \delta/2]$.

Thus, the system is globally asymptotically stable and $\gamma_j^i - \hat{\gamma}_j^i$ will converge to the interval $[-\delta/2, +\delta/2]$.

Note that when δ is smaller enough and the algorithm starts with a small $\gamma_j^i[0]$, we can guarantee that $\gamma_j^i[k] \leq b_j^i$ for all k. Combining Lemma 14.1, Theorem 14.1 and Theorem 14.2, we conclude that *the number of coupons consumed under the two time-scale algorithm converges to the optimal solution to OPT 1.*

14.7 Delay-Based Coupon Forwarding

Suppose that the store rewards relays only after a coupon has been used to make a purchase. The insight that we obtain from the optimality of backpressure is the following:

- If coupons are not used on a particular path, queues build up. This would cause the average delay in being rewarded to all relays on the path to increase.
- If a higher rate of coupons than that set by the store are transferred along a path, the store does not reward the relays for some fraction of coupons and virtual coupons build up. Again, this would mean that the average delay in being rewarded to all relays on the path would increase.

The observation immediately suggests that perhaps a simpler algorithm would be to replace the backpressure based user control of Sect. 14.6 A3 with a much simpler scheme. Users need only to keep track of the average delay experienced in obtaining rewards when they forward coupons to each of their neighbors. They choose to forward coupons to that neighbor who has the lowest such delay. The scheme is intuitively incentive compatible, since users might want to obtain rewards as soon as possible. Thus, we may replace the reward scheme of Sect. 14.6 A2 with an equal reward for all users in the path.

However, a few further additions are required to construct a workable heuristic algorithm. The first addition stems from the fact that under backpressure, if a user finds that all her neighbors have larger effective queue lengths than herself, she does not forward coupons to any of them. The equivalent in the delay based regime would be to simply choose a threshold value of delay (e.g., D_U), and refuse to forward coupons to any neighbor that yields a delay larger than this threshold.

The second addition is that while keeping track of delays, even small differences in delays could result in a particular user being ignored entirely. Hence, instead of a hard comparison between the delays of different options, we soften the comparison. For example, if neighbors 1 and 2 of a node yield delays d_1 and d_2, we consider both as equally lucrative options if $|d_1 - d_2| \leq D_\mathrm{T}$, where D_T is a constant delay threshold. Our expectation is that this simplified algorithm would perform almost as well as the backpressure-scheme.

Based on the observations above, we propose the following delay-based coupon forwarding to replace the user control of Sect. 14.6 A3 for all coupons in which user i is not interested.

Delay-based coupon forwarding: Consider product j that user i is not interested. Denote by $D^i_m(t)$ the average delay experienced in obtaining rewards when user j forwards type i coupons to neighbor m. User i keeps track $D^i_m(t)$ for all neighbors. At time step t, user j first selects type i^* coupon such that

$$i^* \in \arg\min_i \min_{m:(j,m)\in\mathscr{L}} D^i_m(t)$$

and a subset of neighbors associated with type i^* coupon

$$\mathscr{K}_j^{i^*} = \left\{ m : \substack{\left|D_m^{i^*}(t) - \min_{m:(j,m)\in\mathscr{L}} D_m^{i^*}\right| \leq D_{\mathrm{T}} \\ D_m^{i^*}(t) \leq D_U, (j,m)\in\mathscr{L}} \right\}.$$

Then user j sends

$$\frac{\min\left\{Q_j^{i^*}(t), \mu_j(t)\right\}}{|\mathscr{K}_j^{i^*}|},$$

number of type i^* coupons to each of the neighbors in $\mathscr{K}_j^{i^*}$.

Remark: Backpressure based user control requires a user to obtain the lengths of coupon queues from her/his neighbors and from the store. Delay-based coupon forwarding, on the other hand, does not require any information exchange among the users. Each user makes decisions based on her/his own information history, which results in a much smaller communication overhead as compared to backpressure based user control. Further, unlike backpressure, the reward given to every user in the path of a coupon can be identical.

14.8 Simulation Results

We simulate our coupon distribution system on different network topologies to study the validity of our schemes. For the sake of comparison, we also create a third coupon distribution system in which coupons are forwarded by relays randomly to their neighbors. This would indeed be the case if multihop coupon distribution were allowed without incentives for forwarding in any particular direction. Intuitively, such a distribution scheme should over-distribute coupons, since the distributor receives no feedback. Recall that each large-scale time slot consists of T small time slots.

14.8.1 Simple Tree Topology

A simple tree topology is illustrated in Fig. 14.5. There is a single coupon source, two relays, six leaf nodes (customers), and one store. Relays may choose one of their neighbors to forward coupons to at each time instant. Each customer j has a different value of $\hat{x}_j^l$ and $\hat{x}_j^h$. At each time instant t, users utilize all the coupons that they possess if the cumulative number of purchases made is less than $\left(\hat{x}_j^l + \hat{x}_j^h\right)T$. Once this is done, they purchase a random number of additional goods at the marked price, as long as it is rational to do so (i.e., either $\sum_{\tau=0}^{t} x_j^l[t] \leq \hat{x}_j^l T$ and $\sum_{\tau=0}^{t-1} x_j^h[\tau] \leq \hat{x}_j^h T$, or $\sum_{\tau=0}^{t} x_j^l[\tau] + \sum_{\tau=0}^{t-1} x_j^h[\tau] \leq b_j^i T$ and $\sum_{\tau=0}^{t-1} x_j^h[\tau] \leq \hat{x}_j^h T$). Users repeat this process

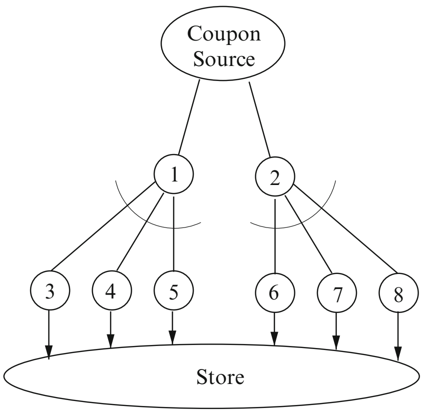


Fig. 14.5 Simple tree topology. This topology is used to verify that the algorithms perform as designed. Coupons are forwarded from top to bottom, from parent nodes to child nodes, and finally arrive at the store

until the end of the small time period $T = 300$. At the last instant $t = T - 1$, if $\sum_{\tau=0}^{T-2} x_j^h[\tau] \leq \hat{x}_j^h T$, user j purchases $\hat{x}_j^h T - \sum_{\tau=0}^{T-2} x_j^h[\tau]$ goods.

We first verify that the small time scale dynamics of backpressure is able to distribute the correct number of coupons to any user j. The capacities of all the relay links are set to 300 coupons per unit time. We illustrate the trajectory of purchases made by user 3 who has $\hat{x}_3^l = 50$ and $\hat{x}_3^h = 60$ over a time interval $T = 300$ units in Fig. 14.6. For purposes of illustration, we assume that $\gamma_3 = \hat{x}_3^l = 50$. In other words, we set the reward rate for coupon forwarding by neighbors of user 3 exactly equal to the average rate at which the user 3 should be given coupons in order to extract maximum revenue. The objective is to test whether the backpressure scheme would achieve this target. We see in Fig. 14.6 that the backpressure scheme indeed gives the right number (and rate) of coupons to the user, ensuring that $x_3^l[T] = 50$ and $x_3^h[T] = 60$.

We now simulate all three schemes (small and large time scales) and the results are as follows. Figure 14.7(a) shows the fractional error in high and low price purchases made by user 3, as compared to $\hat{x}_3^h$ and $\hat{x}_3^l$ for the delay based-scheme. We notice that there is a significant error, which is likely to impact the revenue generated by this scheme negatively. We plot the same quantities when we use the backpressure-scheme in Fig. 14.7(b). The scheme quickly converges, causing the errors to be small. Hence, we expect the revenue generated by this scheme to be close to optimum. Finally, we plot the same for the delay-based scheme in Fig. 14.7(c). For this scheme, we chose the cut-off threshold to be $T/8$ and the acceptable delay difference to be 15%. We observe that the error of this scheme lies in-between that of the other two schemes, which implies that its revenue generation potential is likely to be in-between the other two schemes.

Finally, we plot the total revenue obtained by the store for the three different schemes, and compare them to the maximum possible revenue in Fig. 14.8. The upper bound is the value $\sum_j p\hat{x}_j^h + q\hat{x}_j^l$, which is the maximum extractable revenue. We see that the randomized algorithm does significantly worse than backpressure as well as delay-based schemes. Even accounting for the fact that a constant part

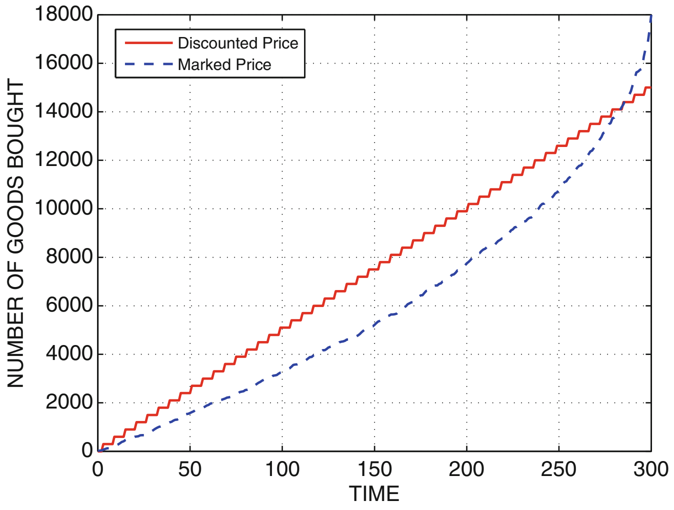


Fig. 14.6 An example trajectory for user 3 over the small time scale. The solid line indicates purchases made at the marked price, while the dashed line indicates discounted purchases. The user has $\hat{x}_j^l = 50$ and $\hat{x}_j^h = 60$. In this example, we have set (for illustration) $\gamma_j = \hat{x}_j^l = 50$, i.e., the reward rate for coupon forwarding is known exactly, and we see that the user receives exactly the right number of coupons

of the revenue would have to be used to incentivize the scheme, the performance improvement is still significant, although the delay-based scheme performs worse than backpressure.

14.8.2 Power Law Topology

We now consider a scale-free network where the node degree distribution follows a power-law topology, i.e., the fraction of nodes having degree k is approximately $ck^{-\gamma}$ for some constants c and γ, where γ is between 2 and 3 typically. Scale-free networks bear a closer resemblance to real-world social networks such as citation networks and collaboration networks. In this simulation, we evaluate the performances of the proposed algorithms using a scale-free network. The graph consists of 100 nodes, and is constructed using preferential-attachment [6] with each entering node connecting to two others with probability proportional to the current degrees of the target nodes, as illustrated in Fig. 14.9. Once the topology has been generated, nodes are connected to the coupon source independently with probability 0.2. Finally, nodes are labeled as relays or customers independently with probabilities 0.7 and 0.1, respectively. Customers have arbitrary spending capacities. We show the upper bound and the performance of the three schemes in

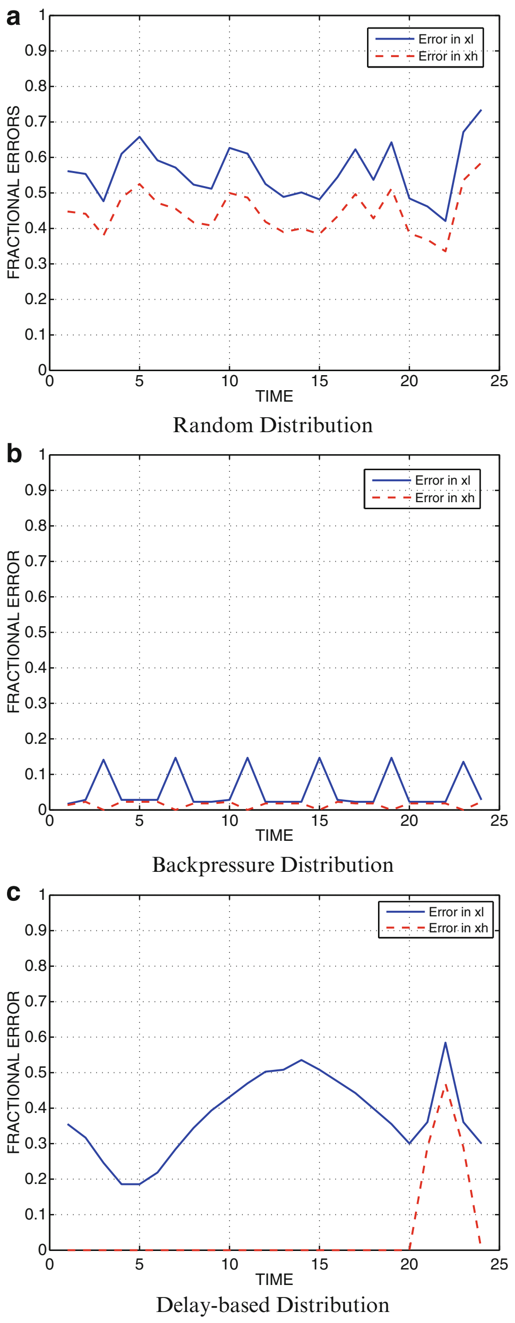


Fig. 14.7 Example trajectories of fractional errors (as compared to $\hat{x}_3^h$ and $\hat{x}_3^l$) in the number of goods bought by user 3 at the marked and discounted price. (**a**) Random Distribution (**b**) Backpressure Distribution (**c**) Delay-based Distribution

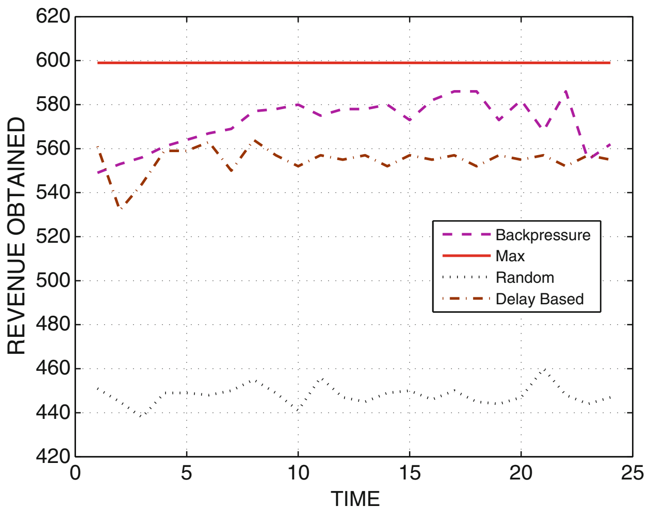


Fig. 14.8 Trajectory of revenue obtained by the store using different schemes. The upper bound is the maximum possible revenue, while the other trajectories correspond to our three schemes

Fig. 14.9 Preferential attachment: Entering nodes join two other nodes with probability proportional to the degrees of the target nodes

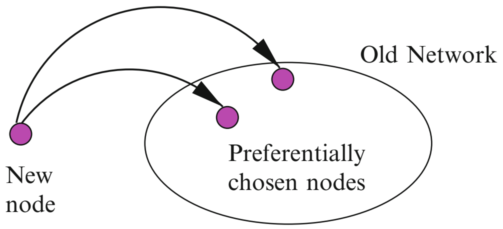


Fig. 14.10. Backpressure clearly performs the best, followed by the delay-based and random schemes. The results indicate that using such coupon distribution schemes could significantly increase the revenue obtained.

14.9 Conclusion

We developed distributed schemes for targeted coupon delivery using online social networks. The objective was to create a two-tier price structure for maximum revenue extraction by selective discounting. We designed systems that allow users to obtain coupons from their neighbors, and incentivize these neighbors by rewarding them for coupon forwarding. We proved how backpressure ideas could be used to achieve a optimal solution, and also how to use it to obtain a simpler (albeit less efficient) scheme. Future work includes dealing with potential malicious users and a testbed implementation.

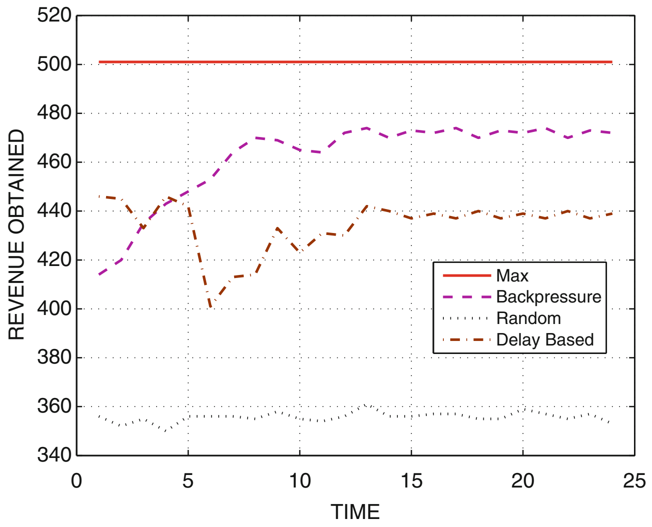


Fig. 14.10 Trajectory of revenue obtained by the store using different schemes for the power-law topology

Acknowledgements Research was funded in part by NSF grant CNS-0904520 and Qatar Telecom, Doha, Qatar.

References

1. Abe, N., Biermann, A., Long, P.: Reinforcement learning with immediate rewards and linear hypotheses. Algorithmica **37**(4), 263–293 (2003)
2. Armengol, A.C., Jackson, M.O.: The effects of social networks on employment and inequality. American Economic Review **94**(3), 426–454 (2004)
3. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.: The nonstochastic multiarmed bandit problem. SIAM Journal on Computing **32**(1), 48–77 (2003)
4. Bala, V., Goyal, S.: Learning from neighbors. Review of Economic Studies **65**, 595–621 (1998)
5. Banks, D., Carley, K.: Metric inference for social networks. Journal of Classification (Springer) **11**(1), 121–149 (1994)
6. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)
7. Boyd, D.M., Ellison, N.B.: Social network sites: Definition, history, and scholarship. Journal of Computer-Mediated Communication **13**(1) (2007)
8. Bramoulle, Y., Kranton, R.: A model of public goods: Experimentation and social learning, vol. 135 (2007)
9. Chen, L., Low, S.H., Chiang, M., Doyle, J.C.: Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks. In: IEEE Infocom. Barcelona, Spain (2006)
10. Chiang, M., Low, S.H., Calderbank, A.R., Doyle, J.C.: Layering as optimization decomposition: A mathematical theory of network architectures. In: Proceedings of the IEEE, pp. 255–312 (2007)

11. Choi, S., Gale, D., Kariv, S.: Behavioral aspects of learning in social networks: An experimental study. Advances in Applied Microeconomics **13** (2005)
12. Clauset, A., Moore, C., Newman, M.E.J.: Hierarchical structure and the prediction of missing links in networks. Nature **453**, 98–101 (2008)
13. Eryilmaz, A., Srikant, R.: Joint Congestion Control, Routing and MAC for Stability and Fairness in Wireless Networks. IEEE Journal on Selected Areas in Communications **24**(8), 1514–1524 (2006)
14. Facebook. http://www.facebook.com/(2009)
15. Fiore, A., Donath, J.: Homophily in online dating: When do you like someone like yourself? In: Proceedings of the ACM Conference on Human Factors in Computing Systems, pp. 1371–1374. New York, NY, USA (2005)
16. Friendster. http://www.friendster.com/(2008)
17. Gale, D., Kariv, S.: Bayesian learning in social networks. Games and Economic Behavior **45**(2), 329–346 (2003)
18. Georgiadis, L., Neely, M.J., Tassiulas, L.: Resource Allocation and Cross-Layer Control in Wireless Networks. Foundations and Trends in Networking, pp. 1–149 (2006)
19. Jackson, M.O., Wolinsky: A strategic model of social and economic networks. J. Economic Theory **71**(1), 44–74 (1996)
20. Joffe, B.: New business models in online communities. In: Proceedings of Media'08. Sydney, Australia (2008)
21. Kelly, F.P.: Multi-armed bandits with discount factor near one: The Bernoulli case. Adv. Appl. Prob. **9**, 897–1001 (1982)
22. Kelly, F.P.: Charging and rate control for elastic traffic. European Transactions on Telecommunications **8**, 33–37 (1997)
23. Kelly, F.P.: Models for a self-managed Internet. Philosophical Transactions of the Royal Society **A358**, 2335–2348 (2000)
24. Kelly, F.P.: Mathematical modelling of the Internet. In: Mathematics Unlimited - 2001 and Beyond (Editors B. Engquist and W. Schmid), pp. 685–702. Springer-Verlag, Berlin (2001)
25. Kelly, F.P., Maulloo, A., Tan, D.: Rate control in communication networks: Shadow prices, proportional fairness and stability. J. Operational Research Society. **49**, 237–252 (1998)
26. Lin, X., Shroff, N., Srikant, R.: A tutorial on cross-layer optimization in wireless networks. IEEE J. Sel. Areas Commun. (2006)
27. Liu, H.: Social network profiles as taste performances. Journal of Computer-Mediated Communication **13**(1) (2007)
28. Liu, H., Maes, P., Davenport, G.: Unraveling the taste fabric of social networks. International Journal on Semantic Web and Information Systems **2**(1) (2006)
29. Low, S.H., Lapsley, D.E.: Optimization flow control, I: Basic algorithm and convergence. IEEE/ACM Trans. Network. **7**(6), 861–875 (1999)
30. Lu, M.: Net group wants action on spam. Taipai Times. (2008). http://www.taipeitimes.com/News/taiwan/archives/2008/12/09/2003430651
31. mGinger. http://www.mginger.com/(2009)
32. MySpace. http://www.myspace.com/(2008)
33. Neely, M., Modiano, E., Li, C.: Fairness and optimal stochastic control for heterogeneous networks. In: Proc. IEEE Infocom., vol. 3, pp. 1723–1734. Miami, FL (2005)
34. Orkut. http://www.orkut.com/(2009)
35. Second Life. http://www.secondlife.com/(2008)
36. Shakkottai, S., Srikant, R.: Network Optimization and Control. Foundations and Trends in Networking. Now Publishes, Delft, The Netherlands (2008)
37. Spertus, E., Sahami, M., Büyükkökten, O.: Evaluating similarity measures: A large-scale study in the Orkut social network. In: Proceedings of 11th International Conference on Knowledge Discovery in Data Mining, pp. 678–684. New York, NY, USA (2005)
38. Srikant, R.: The Mathematics of Internet Congestion Control. Birkhauser, Boston, MA (2004)
39. Stolyar, A.: Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. Queueing Systems **50**(4), 401–457 (2005)

40. Tassiulas, L., Ephremides, A.: Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. IEEE Transactions on Automatic Control pp. 1936–1948 (1992)

Chapter 15
Social-Aware Data Diffusion in Delay Tolerant MANETs

Yang Zhang, Wei Gao, Guohong Cao, Tom La Porta, Bhaskar Krishnamachari, and Arun Iyengar

Abstract Most existing mobility-assisted data access techniques in delay tolerant mobile ad hoc networks (DT-MANETs) are designed to disseminate data to one or several particular destinations. Different from these works, we study the *data diffusion* problem which diffuses data among all moving nodes so that the nodes that are interested in this data item can get it easily either from their encountered friend nodes or stranger nodes. To reduce the data access delay, we introduce four social-aware data diffusion schemes based on the social relationship and data similarity of the contacts. We also provide solutions to quantify data/interest similarity and to determine whether two nodes are friends or strangers. Theoretical models are developed to analyze the data diffusion process and compare the performance of the four proposed diffusion schemes in terms of diffusion speed and query delay. We use real traces of human contacts to emulate data diffusion under different schemes. Both theoretical analysis and experimental results imply an interesting fact: to achieve better diffusion performance, each node should first diffuse the data similar to their common interests when it meets a friend, and first diffuse the data different to their common interests when it meets a stranger.

Y. Zhang (✉) • W. Gao • G. Cao • T.L. Porta
Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA
e-mail: yangzhan@cse.psu.edu; wxg139@cse.psu.edu; gcao@cse.psu.edu; tlp@cse.psu.edu

B. Krishnamachari
Department of Electrical Engineering, University of Southern California, Los Angeles, CA
e-mail: bkrishna@usc.edu

A. Iyengar
IBM T.J. Watson Research Center, Hawthorne, NY
e-mail: aruni@us.ibm.com

15.1 Introduction

With the rapid adoption of mobile hand-held devices (e.g., PDAs, Bluetooth enabled mobile phones, active RFID tags, etc.), more and more people use them to diffuse, query and share interesting data among themselves without any network infrastructure support. Such a community-wide (or even city-wide) network formed by the mobile hand-held devices is an example of a *delay tolerant mobile ad hoc network (DT-MANET)* [1].

Due to the low node density and unpredictable network topology, routing paths in DT-MANETs may be frequently disconnected. To deal with such problems, *mobility-assisted data access* techniques have been exploited, where a node physically carries data for some time until it moves within the communication range of some other node (i.e., *contact*) [2–4]. Then it decides whether to propagate the data to the new contact or not based on some algorithm. Existing algorithms such as *SimBet Routing* [5], *BUBBLE Rap* [6], *SocialCast* [7], *SOLAR* [8], and *MaxProp* [9] are designed to forward data to one given destination. Since the source and destination may be faraway from each other, the delay for the destination to get the data from the source may be long.

One way to reduce the query delay is through *data diffusion*, where data is diffused throughout the network and replicated in advance. Subsequent queries can be served by any node with the data instead of being sent to the source node, and thus reducing the query delay. However, data diffusion is not free. In DT-MANETs, nodes diffuse data when they are in contact. Because the contact time is pretty short and the buffer size of each node is limited, the diffused data has to compete for the limited bandwidth and buffer space. Therefore, the diffusion decisions made by each node such as which data should be propagated first and which data should be replaced out of the buffer, affect the diffusion speed and the data access delay.

In this chapter, we study the performance of different data diffusion schemes in DT-MANETs. Our classification of these data diffusion schemes is based on social networks. Social networks exhibit the “homophily” phenomenon [10] which comes from the observation that individuals often befriend others who have similar interests, and hence perform similar actions and have a higher possibility to meet with each other. For example, two individuals who own the same kind of video game console are more likely to become friends and meet at game shops due to the common interest in games. Students from the same department are more likely to take the same courses and appear in the same lab or building. Therefore, the contact frequencies are probabilistically different between two friends and two strangers, and this difference should be taken into consideration when designing data diffusion schemes.

To study the effects of social networking on data diffusion, we are interested in answering the following questions: *How does the data diffusion scheme used affect the diffusion speed and the data access delay? How to design better data diffusion schemes based on social networking results?* To answer these questions, we introduce four possible social-aware data diffusion schemes and develop theoretical models to analyze their performance in terms of diffusion speed and query delay. Based on the analysis, we discover an interesting result: to achieve better performance, when a node meets a new contact, if the new contact is a friend, it should first diffuse the data similar to their common interests; if the new contact is a stranger, it should first diffuse the data different from their common interest. We also provide solutions to determine if a new contact is a friend or stranger based on their common interests. To verify the theoretical results, we use two real social contact traces [11, 12] to emulate the data diffusion process under different schemes, and find that the experimental results are consistent with our theoretical analysis.

The rest of this chapter is organized as follows. Section 15.2 discusses the related works. Section 15.3 describes the application scenario, as well as four social-aware data diffusion schemes and their implementation techniques. In Sect. 15.4, we presents the theoretical analysis on diffusion speed and access delay of the four schemes. Performance evaluations are shown in Sect. 15.5. Finally, we conclude the chapter in Sect. 15.6.

15.2 Related Work

There have been several theoretical and empirical works on how social behavior can be used to improve the performance of data access in delay tolerant networks. PeopleNet [13] is a wireless virtual social network which mimics the way people seek information via social networking. It is simple and scalable for efficient information search in a distributed manner. However, it uses infrastructure to propagate data and queries, which is different from the peer-to-peer MANET scenario. *SimBet Routing* [5] studies the "small-world" phenomenon of human society and uses ego-centric centrality and its social similarity to guide data forwarding. Messages are forwarded towards the node with higher centrality. Similarly, *BUBBLE Rap* [6] focuses on community and social centrality, and nodes are structured into communities. High popularity nodes and community members of the destination are selected as relays. Ghosh et al. [8] have identified the orbital movement pattern of human being and relay nodes are chosen based on the places that they frequently visit. Similarly, Costa et al. [7] provide a routing framework using social interaction information in publish-subscribe systems and Gao et al. [14] study the social-aware multicasting issues in delay tolerant networks. Bai and Helmy [15] study the last encounter based routing protocol that utilizes encounter history to create time gradients for information diffusion in wireless networks. Furthermore, Gao and Cao [16] exploits transient contact patterns to improve the performance of data forwarding, and Li et al. [17] considers the selfishness property of social nodes in data forwarding. Although [5–8, 14–18] have applied sociological knowledge to data dissemination in DT-MANETs, these works consider the problem of disseminating data to one pre-determined destination node. Unlike these existing works, data diffusion is not for settings with a specific destination.

The aforementioned works aim to find the most suitable relay node to increase the possibility of reaching the final destination. Miklas et al. [19], Karagiannis et al. [20], Chaintreau et al. [4] and Wang et al. [21] study the social factor of delay tolerant networks in a different way. They analyze the distribution of inter-contact time between mobile devices and conclude that the inter-contact time follows the power law distribution or the exponential decay distribution. Further, Hsu et al. [22] analyze wireless users' behavioral patterns by extensively mining wireless network logs and discover that the size of distinct WLAN user group follows a power-law distribution. Besides, in the area of vehicular DT-MANETs, algorithms [23] and [24] have been proposed for finding the right relays for data forwarding and Kapadia et al. [25] consider the problem of optimizing the replication profile of content to minimize the aggregate average data access delay given knowledge of content popularity. In [26], the authors also present a cache replacement policy for finite buffers in a vehicular DT-MANET that takes into account the differing interests of vehicles in different geographic locations, but this work does not explicitly consider social interactions between users. Our work differs from these works in that we study how to use the social network results to improve data dissemination through social aware data diffusion schemes.

To the best of our knowledge, the closest studies to our work are ContentPlace [27] and PodNet [28]: when two nodes meet, they decide which data object to exchange based on the information gathered about nodes' interests. However, both [27] and [28] are designed for publish/subscribe services, they assume each node only fetches its interesting data (for single node in [28] or for single community in [27]). Our work, instead, studies the problem in a more general perspective where nodes can not only carry the data they are interested in, they can also buffer data they are not interested in, and forward them to other interested nodes in the future.

Data dissemination can be modeled as spreading of infectious disease. Disease spreading in fixed networks has been studied in the past [3]. [29] also analyzed the epidemic spreading in mobile networks. Different from the analysis in [3] and [29] where all nodes have the same moving and interest features, in our data diffusion scenario, both the interested nodes and uninterested nodes can help diffuse data and they have different meeting frequency and different interest preference. Therefore, different infection and immunization rates between interested and uninterested nodes are studied and the overall diffusion rates by both interested and uninterested nodes are investigated in our work, which adds complexity to the analysis.

15.3 Social-Aware Data Diffusion

In this section, we first present the target application scenario that our work relies on. After that, we introduce social-aware data diffusion as four possible schemes based on a two-dimensional classification that combines both interest similarity and data similarity, and then provide the necessary implementation techniques for the similarity classification, which is based on the predefined threshold.

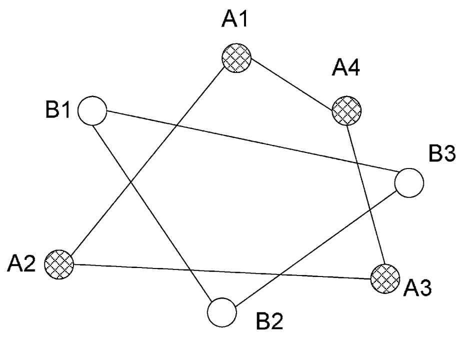


Fig. 15.1 Application scenario

15.3.1 Application Scenario

The application scenario we target is similar to the one used in ContentPlace [27] and PodNet [28] where we consider a number of mobile users whose devices cannot be encompassed by the conventional MANETs. Instead, communication is achieved by opportunistic pairwise contacts between users to exchange data objects. According to the homophily phenomenon, users with similar data interests have strong social relationships with each other. Further, people movements are governed by their social relationships, and by the fact that people with similar interests are also mostly bound to particular places (landmarks) that are associated with the interested topics [30,31]. Therefore, users will spend most of their time and meet more friends at the landmarks they are bound to, and will also visit some other places occasionally and meet some stranger nodes at other places. This application scenario fits many existing service environments. For example, sports fans will spend more time in the sports related stores instead of cosmetics related stores when they are visiting an outlet. As a result, they are easy to meet other people with the same interests at those sports stores. In other words, two people in contact at these stores have a high possibility to have the similar interests.

Figure 15.1 gives a possible scenario with seven landmarks. Landmark A1–A4 are associated with the similar interest topics (e.g., sports related) and B1–B3 focus on another group of interest topics (e.g., cosmetics related). Then, people interested in sports are more likely to visit landmarks A1–A4. It is possible that they meet some people with different interests when they visit other landmarks sometimes. But in general, the proposition still holds that their contact rates with the people of similar interests are higher than that with other stranger people, and most of their encounters at the interested landmarks have similar interests as they have.

15.3.2 Diffusion Schemes

When two nodes meet each other, they exchange two lists. One is called the *interests list* which is the list of the interesting data; the other is the *data list* which records the data they are buffering. Based on these two lists, each node decides whether the encountered node can serve its query request and make further data diffusion decisions.

From the data perspective, all nodes can be divided into two non-overlapped groups, depending on whether if they are interested in one particular data or not. If one node is interested in the data, then this node is called the interested node of the data; otherwise, its is the uninterested node. Meanwhile, from the social network perspective, each node has two kinds of contacts: *friends* and *strangers*, where two nodes are friends if they have more similar interests, and they are strangers otherwise. According to homophily, friends usually share more common interests while strangers have less common interest (more details in Sect. 15.3.3). Therefore, we always hope that data can be diffused to interested nodes quickly so that most nodes can access their interesting data easily. However, the contact time can be very short in DT-MANETs and thus some data cannot be diffused between the two contacts. Also, the memory constraint limits the number of data items that a node can hold. Thus, we should carefully choose the most suitable data to diffuse and buffer first.

Without considering sociological knowledge, nodes diffuse data based on their own interests. Each node fetches and buffers interesting data from its contacts. Due to the bandwidth and buffer limitations, this solution has slow diffusion speed since each node only helps diffuse its own interesting data while neglects others. Another approach is to diffuse data randomly, where all data have the same opportunity to be diffused. However, this solution may diffuse much uninteresting data to some nodes, thus wasting the limited bandwidth and buffer space, and increasing the query delay.

With sociological knowledge, contacts can be categorized as friends or strangers and data can be categorized as being interesting or uninteresting. Thus, we have four possible data diffusion schemes by combining nodes' relationship and their interests in the data (as shown in Fig. 15.2):

1. FsSd: When a node meets a new contact, if the new contact is a friend, it first sends the data of their common interest. These data items will be sorted based on their common interest (more details in Sect. 15.3.3). Each node sends the most similar data to its friend first, and then the second most similar data until the contact time is over. If the new contact is a stranger, it first sends the data different from their common interest. It will send the most different data first, and then the second most different data until the contact time is over.

   To summarize, it diffuses the most *S*imilar data between *F*riends, and diffuses the most *D*ifferent data between *S*trangers.
2. FsSs: it diffuses the most *S*imilar data between both *F*riends and *S*trangers.
3. FdSd: it diffuses the most *D*ifferent data between both *F*riends and *S*trangers.
4. FdSs: it diffuses the most *D*ifferent data between *F*riends and diffuses the most *S*imillar data between *S*trangers.

If friends first diffuse the data that is most close to their common interests (Fs), their interesting data will have priority to be propagated and buffered between themselves. However, if friends diffuse the most different data first (Fd), the

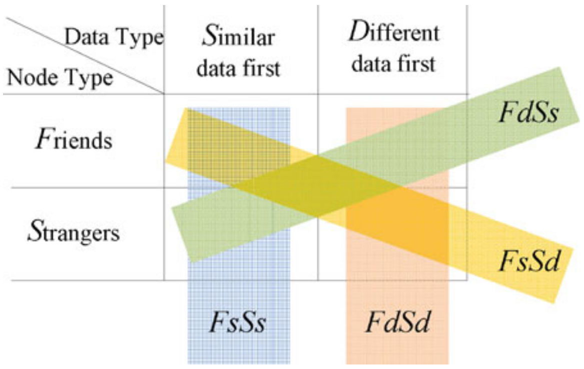


Fig. 15.2 Data diffusion schemes

diffusion probability of their common interesting data will be low. On the other hand, if strangers first diffuse the data most different from their common interests as they meet (Sd), for one specific data (notice that strangers share less interest similarity), the data still has a high probability to be diffused from its interested node to its uninterested node, and vice versa. Therefore, with FsSd, a node should be able to quickly diffuse data among its interested nodes, as well as between the interested nodes and its directly encountered uninterested nodes. Based on "homophily," friends have higher meeting frequency than strangers. If one data item can be buffered at more interested nodes, the query delay for this data item can be reduced. In this sense, FsSd may have the best diffusion performance. Before verifying this result through both theoretical analysis and experiments, we provide techniques for quantifying the interest/data similarity.

15.3.3 Measuring Similarity

To measure similarity, the first step is to formalize the description of data and query. Both data and query can be presented and indexed with resource representation techniques such as RDF (i.e., Resource Description Framework [32]) or WSDL (i.e., Web Services Description Language [33]) based on specific keyword attributes. In this chapter, to support complex data description, we associate each data with a sequence of keywords and define a mapping that preserves keyword similarity. The keywords are common words to describe data attributes such as "entertainment," "sport," "news," "travel," and etc. For example, music data may be labeled with "entertainment" and restaurant information can be indexed with "travel." Meanwhile, one data can have multiple attributes, thereby the same sport video might be labeled with both "entertainment" and "sport." Following this mapping method, all attributes form a multi-dimensional keyword space so that the data is indexed by a multi-dimensional binary vector. If the data has one attribute, its corresponding bit in the vector is marked "1"; otherwise, it is marked "0." For

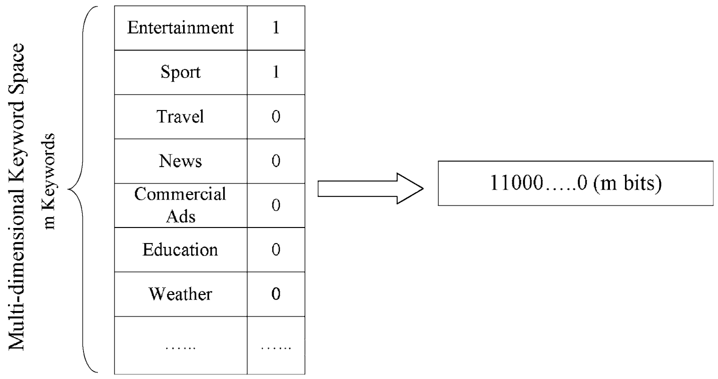
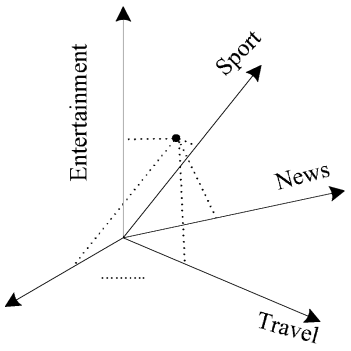


Fig. 15.3 Data description with m-dimensional attribute vector

Fig. 15.4 A m-dimensional keyword space



the simplicity of analysis and without loss of generality, we assume the attribute space is m-dimensional. Then each data can be described and indexed by a m-bit vector. Figure 15.3 demonstrates how to determine the vector of one data item, and Fig. 15.4 shows an example of a keyword space. Similarly, query messages can be described in a similar way.

Measuring interest similarity and data similarity for the classification of data diffusion schemes

The classification of our data diffusion schemes is based on the two-dimensional comparison of nodes' interest similarity and data similarity.

First, in social-aware data diffusion, nodes make diffusion decisions based on their relationship (i.e., friends or strangers). Two friends share more common interests while two strangers have less interest similarity. Therefore, we need to estimate the interest similarity of two nodes to decide their relationship. Since node interest follows a probability distribution on different attributes, the interest similarity between two nodes should be calculated with two distributions. The Kullback–Leibler (K-L) divergence method [34] is used here to measure the difference between two probability distributions. If we use P_1 and P_2 to denote the discrete interest distributions of two nodes, the K-L divergence of P_2 from P_1 is defined to be

$$D_{\mathrm{KL}}(P_1 \parallel P_2) = \sum_{i=1}^{m} P_1(i) \log \frac{P_1(i)}{P_2(i)}.$$

Therefore, the interest similarity of two nodes can be estimated as

$$\begin{aligned} \mathrm{SV}_{\mathrm{dd}} &= \frac{1}{D_{\mathrm{KL}}(P_1 \parallel P_2)} \\ &= \frac{1}{\sum_{i=1}^{m} P_1(i) \log \frac{P_1(i)}{P_2(i)}}. \end{aligned} \tag{15.1}$$

Suppose $\mathrm{FS}_{\mathrm{thres}}$ is the interest threshold to estimate the interest similarity of two nodes. If one node pair has a $\mathrm{SV}_{\mathrm{dd}}$ smaller than $\mathrm{FS}_{\mathrm{thres}}$, they share few common interests so that they are strangers; otherwise, they are friends.

Note that the K-L divergence is not symmetric, which means $D_{\mathrm{KL}}(P_1 \parallel P_2)$ is not necessarily equal to $D_{\mathrm{KL}}(P_2 \parallel P_1)$. In this chapter, we always use the interest distribution of the node with smaller ID as the first parameter (i.e., P_1) of the K-L divergence calculation and the node with larger ID as the second parameter (i.e., P_2).

Second, during each contact, nodes need to sort the data according to the data similarity to their common interests. Since the node's interests are presented by distributions and data objects are described by vectors. We need to compare the similarity between one vector and one discrete distribution. In this case, the similarity of one vector and one distribution can be calculated by their inner-product. Formally,

$$\begin{aligned} \mathrm{SV}_{vd} &= \parallel \mathbf{V} \cdot \mathbf{P} \parallel \\ &= \sum_{i=1}^{m} v^i \times p_{\mathrm{i}}, \end{aligned} \tag{15.2}$$

where $\mathbf{V} = \langle v_1^1, v_1^2, ...v_1^m \rangle$ is the description vector of data V, and $\mathbf{P} = \langle p_1, p_2, ...p_m \rangle$ is the distribution vector of the discrete interest distribution P. With the calculation of SV_{vd} and an interest threshold $\mathrm{IN}_{\mathrm{thres}}$, each node can distinguish the data that is most similar or different to nodes' common interests and choose the most proper ones to diffuse.

15.4 Theoretical Analysis of Data Diffusion

In this section, we develop theoretical models to analyze the performance of data diffusion. In Sect. 15.4.1, we first study the case in which nodes have infinite buffer. Without buffer limitation, nodes spread data to as many contacts as possible and never remove data from their buffers. However, due to the limitation of the

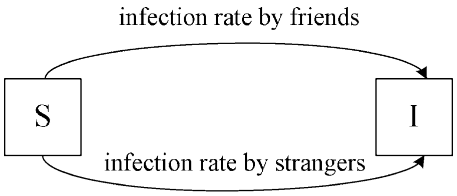


Fig. 15.5 Markov chain model of the S-I infectious disease with susceptible state and infected state (with infinite buffer)

contact time, not all data can be diffused during each contact. Different decisions on diffusing similar or different data between friends and strangers still affect the performance of data diffusion. In Sect. 15.4.2, we consider the finite buffer case where some data items have to be replaced when the buffer is full.

15.4.1 The Infinite Buffer Case

The diffusion of each data item can be modeled as spreading of infectious disease. Disease spreading in fixed networks has been studied in the past [3]. [29] also analyzed the epidemic spreading in mobile networks. In the infectious disease model, one node is "infected" if it has the data buffered in its memory. The node is "susceptible" to infection if it does not have the data, but could potentially get the data from other nodes. Different from the traditional "Susceptible–Infected–Recovered (S-I-R)" model [3, 29], in the infinite buffer data diffusion scenario, data is never deleted as long as it is buffered at some node. Therefore, all nodes follow a two-state compartmental S-I model. Meanwhile, both the interested nodes and the uninterested nodes can help diffuse the data. Therefore, the different infection rates between interested and uninterested nodes should be considered.

First, for the interested nodes, as shown in Fig. 15.5,

$$\begin{aligned}&\text{total infection rate of interested node}\\&\quad = \text{infection rate by friends} + \text{infection rate by strangers}\end{aligned}$$

We use susceptible state $S(t)$ and infected state $I(t)$ represents the number of nodes which are "susceptible" and "infected" in the system at time t, respectively. Then, $I(t) = I_{\mathrm{i}}(t) + I_{\mathrm{u}}(t)$ and $S(t) = S_{\mathrm{i}}(t) + S_{\mathrm{u}}(t)$ where $I_{\mathrm{i}}(t)$ and $S_{\mathrm{i}}(t)$ are the numbers of "infected" and "susceptible" interested nodes, and $I_{\mathrm{u}}(t)$ and $S_{\mathrm{u}}(t)$ are the numbers of "infected" and "susceptible" uninterested nodes. β is the contact rate of one node to meet any other node,[1] which consists of the contact rate among friends

[1]The contact rate does not mean the pairwise contact times for two specific nodes. Instead, it is the average number of contact for one node to meet any other node in the system.

(β_f) and the contact rate among strangers (β_s). Further more, γ_f and γ_s are used to denote the data diffusion probability between two interested friends and from one interested node to any other uninterested stranger. Suppose there are N_i interested nodes in the system, then an interested node contacts $\beta_f(N_i - 1)$ other friends per unit time, of which $\frac{S_i}{N_i-1}$ do not yet have the data. The probability that the data will be exchanged to the encountered friend is γ_f. Therefore, the infection rate by friends can be estimated as

$$\begin{aligned}
&\text{infection rate by friends}\\
&\quad = (\sharp \text{ of infected nodes})(\text{contact rate of friends})\\
&\qquad \times(\text{infect probability of friends})(\sharp \text{ of susceptible nodes})\\
&\quad = I_i(\beta_f \times (N_i - 1)) \times \gamma_f \times \frac{S_i}{N_i - 1}\\
&\quad = I_i \beta_f S_i \gamma_f.
\end{aligned}$$

Similarly, we can get the infection rate by strangers as $I_i\beta_s S_i \gamma_s$. Then, for a particular data item, the transition rate of any interested node from state S to state I becomes

$$\begin{aligned}
&\text{total infection rate of interested node}\\
&\quad = \text{infection rate by friends} + \text{infection rate by strangers}\\
&\quad = I_i\beta_f S_i\gamma_f + I_i\beta_s S_i\gamma_s\\
&\quad = I_i S_i(\beta_f\gamma_f + \beta_s\gamma_s).
\end{aligned}$$

We are interested in the transient solution to the Markov chain in Fig. 15.5. We can get $I_i(t)$ by solving the following first-order differential equation,

$$\begin{aligned}
\frac{dS_i}{dt} &= -I_i S_i(\beta_f\gamma_f + \beta_s\gamma_s)\\
\frac{dI_i}{dt} &= I_i S_i(\beta_f\gamma_f + \beta_s\gamma_s)\\
&= I_i(N_i - I_i)(\beta_f\gamma_f + \beta_s\gamma_s)\\
&= (\beta_f\gamma_f + \beta_s\gamma_s)N_i I_i - (\beta_f\gamma_f + \beta_s\gamma_s)I_i^2.
\end{aligned}$$

This differential equation is separable and can be solved with the initial conditional $I_i(0) = 1$ to get the solution

$$I_i(t) = \frac{N_i}{1 + e^{-(\beta_f\gamma_f+\beta_s\gamma_s)N_i t}(N_i - 1)}. \tag{15.3}$$

Similarly, based on the same S-I model, we can get the total infection rate of uninterested nodes as

$$\begin{aligned}
&\text{total infection rate of uninterested node}\\
&\quad = \text{infection rate by friends} + \text{infection rate by strangers}\\
&\quad = I_u \beta_f S_u \gamma_f' + I_u \beta_s S_u \gamma_s'\\
&\quad = I_u S_u (\beta_f \gamma_f' + \beta_s \gamma_s'),
\end{aligned}$$

where γ_f' and γ_s' are the diffusion probabilities between two uninterested friends and uninterested strangers.

If we use N_u to represent the number of uninterested nodes in the system, then the first infected uninterested node is expected to appear at time $\frac{1}{\beta_s \cdot N_u \cdot \gamma_s}$. Therefore, $I_u(t)$ can be approximated in the same way as $I_i(t)$ with a time offset of $\frac{1}{\beta_s \cdot N_u \cdot \gamma_s}$, i.e.,

$$I_u(t) = \begin{cases} 0 & t \leq \frac{1}{\beta_s \cdot N_u \cdot \gamma_s} \\ \frac{N_u}{1 + e^{-(\beta_f \gamma_f' + \beta_s \gamma_s') N_u (t - \frac{1}{\beta_s \cdot N_u \cdot \gamma_s})} (N_u - 1)} & \text{else.} \end{cases} \tag{15.4}$$

We use P_i and P_u to denote the probabilities of interested nodes and uninterested nodes to initiate the query. Then for one specific data, its expected query delay at time t, $E_Q(t)$, can be estimated as

$$\begin{aligned}
E_Q(t) = {} & E(query\ delay\ of\ interested\ node) \cdot P_i\\
& + E(query\ delay\ of\ uninterested\ node) \cdot P_u
\end{aligned} \tag{15.5}$$

In particular, if the query is initiated by one interested node, this node can get the data either from its friends (according to "homophily," they are also the interested nodes) or from its strangers (they are uninterested nodes).

First, if the data is from a friend node, because there are N_i interested nodes in the system and $I_i(t)$ interested nodes have the data at time t, the probability that the query node meets one friend and the friend has the data is $\frac{I_i(t)}{N_i - 1}$. Meanwhile, as the query node can contact $\beta_f(N_i - 1)$ interested friends per time unit, the average query delay of this case can be estimated as $\frac{N_i - 1}{I_i(t)} \cdot \frac{1}{\beta_f(N_i - 1)} = \frac{1}{I_i(t) \cdot \beta_f}$. Second, if the data is from a stranger node, the query node contacts $\beta_s(N_u)$ stranger per time unit and the probability that the contact nodes has the data is $\frac{I_u(t)}{N_u}$. Then we can get its average query delay as $\frac{N_u}{I_u(t)} \cdot \frac{1}{\beta_s(N_u)} = \frac{1}{I_u(t) \cdot \beta_s}$.

Therefore, the expectation of the query delay of the interested node is the minimum query delay from either interested nodes or uninterested nodes, i.e.,

$$E(\text{query delay of interested node}) = \min\left\{ \frac{1}{I_i(t) \cdot \beta_f}, \frac{1}{I_u(t) \cdot \beta_s} \right\} \tag{15.6}$$

Table 15.1 The setting of $(\gamma_\mathrm{f}, \gamma_\mathrm{s})$ and $(\gamma'_\mathrm{f}, \gamma'_\mathrm{s})$

	γ_f	γ_s	γ'_f	γ'_s
FsSd	Large	Large	Small	Large
FdSs	Small	Small	Large	Small
FsSs	Large	Small	Small	Small
FdSd	Small	Large	Large	Large

However, if the query is initiated by one uninterested node, there are three kinds of nodes to serve the query: the friends of the node (who are also uninterested nodes), the uninterested strangers, and the interested strangers. Similarly, the expectation of the query delay of the uninterested node can be estimated as

$$\begin{aligned}
&E(\text{query delay of uninterested node})\\
&\quad= \min\left\{\frac{N_\mathrm{i}}{I_\mathrm{i}(t)}\cdot\frac{1}{\beta_\mathrm{s}N_\mathrm{i}}, \frac{N_\mathrm{u}-1}{I_\mathrm{u}(t)}\cdot\frac{N_\mathrm{i}}{N_\mathrm{u}-1}\cdot\frac{1}{\beta_\mathrm{f}N_\mathrm{i}},\right.\\
&\qquad\left.\times\frac{N_\mathrm{u}-1}{I_\mathrm{u}(t)}\cdot\frac{N_\mathrm{u}-N_\mathrm{i}}{N_\mathrm{u}-1}\cdot\frac{1}{\beta_\mathrm{s}(N_\mathrm{u}-N_\mathrm{i})}\right\}\\
&\quad= \min\left\{\frac{1}{I_\mathrm{i}(t)\beta_\mathrm{s}}, \frac{1}{I_\mathrm{u}(t)\beta_\mathrm{f}}, \frac{1}{I_\mathrm{u}(t)\beta_\mathrm{s}}\right\}. \qquad (15.7)
\end{aligned}$$

Therefore, we can get

$$\begin{aligned}
E_Q(t) &= \min\left\{\frac{1}{I_\mathrm{i}(t)\beta_\mathrm{f}}, \frac{1}{I_\mathrm{u}(t)\beta_\mathrm{s}}\right\}\cdot P_\mathrm{i}\\
&\quad+\min\left\{\frac{1}{I_\mathrm{i}(t)\beta_\mathrm{s}}, \frac{1}{I_\mathrm{u}(t)\beta_\mathrm{f}}, \frac{1}{I_\mathrm{u}(t)\beta_\mathrm{s}}\right\}\cdot P_\mathrm{u}. \qquad (15.8)
\end{aligned}$$

Different data diffusion schemes have different combinations of $(\gamma_\mathrm{f}, \gamma_\mathrm{s})$ and $(\gamma'_\mathrm{f}, \gamma'_\mathrm{s})$. For example in FsSd, suppose a node carries its interesting data. When this node meets its friend, most likely it will diffuse the data to its friend. When it meets a stranger the probability to diffuse this data between them is still high. This is because stranger nodes first choose the data that is most different to their common interests to diffuse. Also, two strangers have less interest similarity. If one node is interested in the data, its stranger might not be interested in it. Therefore, both γ_f and γ_s are set to large values in FsSd.

Suppose a node is carrying an uninteresting data item. When it meets a friend, its friend may also have no interest in this data, decreasing the diffusion possibility. If this node meets a stranger that is also not interested in the data, instead, it might diffuse this uninteresting data because the data is still different from the common interests of these two uninterested strangers and should have high diffusion priority according to FsSd. Consequently, γ'_f becomes small and γ'_s is still large. Similarly, we can set the values of $(\gamma_\mathrm{f}, \gamma_\mathrm{s})$ and $(\gamma'_\mathrm{f}, \gamma'_\mathrm{s})$ for the other three schemes as shown in Table 15.1.

Fig. 15.6 Numerical results based on the S-I analysis model ($\beta_f = 0.01$, $\beta_s = 0.002$, $N_i = 20$, $N_u = 80$). (**a**) Infected Nodes (**b**) Query Delay

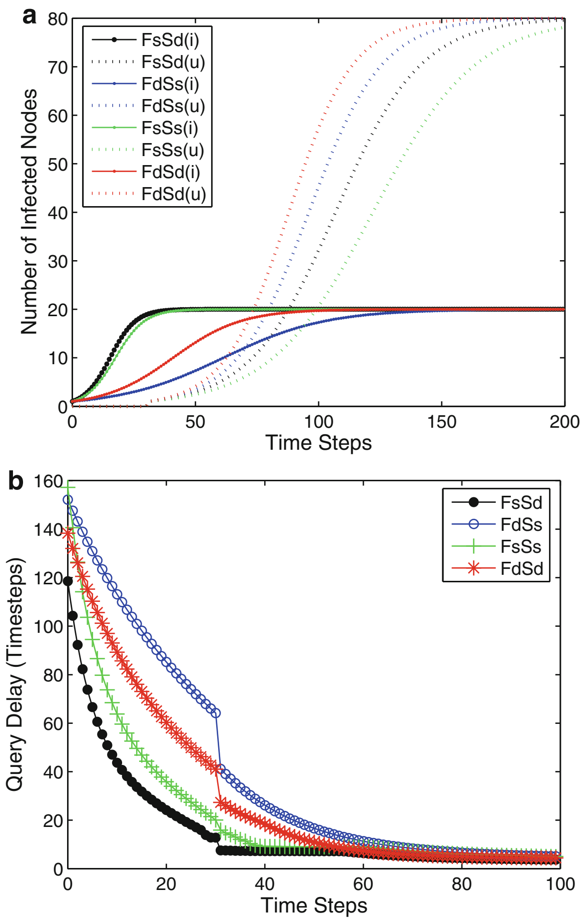


Figure 15.6 shows some numerical results according to the analysis. Figure 15.6(a) depicts the number of infected nodes as a function of time. We use FsSd(i) and FsSd(u) to denote the number of infected interested nodes and infected uninterested nodes, respectively, under the FsSd scheme. The infected nodes of other three schemes are denoted similarly. As there is infinite buffer, all nodes should be infected after some amount of time. From the figure, we can see that at time 200, almost all nodes (20 interested nodes and 80 uninterested nodes) are infected. However, different diffusion schemes have different data diffusing speed. For example, at time 25, all the 20 interested nodes in FsSd(i) are infected, but it takes 130 time units for the 20 interested nodes to be infected in FdSs(i). Note that the diffusion speeds of FsSd and FsSs are slower than FdSs and FdSd among uninterested nodes. This is because both FsSd and FsSs give high diffusion priority to the interested friends while sacrificing the diffusion opportunity of their uninteresting data.

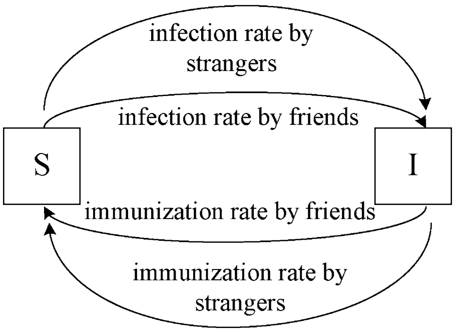

infection rate by strangers
infection rate by friends
S
I
immunization rate by friends
immunization rate by strangers

Fig. 15.7 Markov chain model of the S-I-S infectious disease with susceptible state and infected state (with finite buffer)

Figure 15.6(b) investigates the query delay as a function of time for different diffusion schemes. We can see that FsSd has the shortest query delay. This is because FsSd diffuses the interesting data among its friends quickly. Homophily suggests that friends share more common interests and have a high meeting probability. Therefore, quickly diffusing interesting data among friends results in lower query delay. FdSs has the lowest diffusion priority between interested friends and strangers, and thus it has the slowest diffusion speed and longest query delay. Notice that there is a sudden drop at about time 26. The sudden drop is due to the piecewise function (5) that is used to estimate the appearance time of the first infected uninterested node. After an uninterested node gets the data, many queries might be served, and thus reducing the delay.

15.4.2 The Finite Buffer Case

With finite buffer, the analysis becomes more complicated since data may be removed from the buffer. In this case, the S-I model should be replaced by the S-I-S model in which infected nodes return to the susceptible state on recovery because they are not against reinfection.

Figure 15.7 illustrates the Markov chain model of S-I-S. Similar to the infinite buffer case, we can get the infection rate and the immunization rate and have the mass balance equations for $I_i(t)$ and $I_u(t)$:

$$\frac{dI_i}{dt} = I_i S_i(\beta_f \gamma_f + \beta_s \gamma_s) - (N_i \beta_f \alpha_f + N_u \beta_s \alpha_s) I_i,$$

$$\frac{dI_u}{dt} = I_u S_u(\beta_f \gamma_f' + \beta_s \gamma_s') - (N_i \beta_f \alpha_f + N_u \beta_s \alpha_s) I_u$$

With $S_i = N_i - I_i$ and $S_u = N_u - I_u$ we get:

$$\frac{dI_i}{dt} = I_i(N_i - I_i)(\beta_f \gamma_f + \beta_s \gamma_s) - (N_i \beta_f \alpha_f + N_u \beta_s \alpha_s) I_i$$

$$= ((\beta_f\gamma_f + \beta_s\gamma_s)N_i - (N_i\beta_f\alpha_f + N_u\beta_s\alpha_s))I_i$$
$$-(\beta_f\gamma_f + \beta_s\gamma_s)I_i^2$$
$$= ((\beta_f\gamma_f + \beta_s\gamma_s)N_i - (N_i\beta_f\alpha_f + N_u\beta_s\alpha_s))I_i$$
$$\times \left(1 - \frac{I_i}{N_i - \frac{N_i\beta_f\alpha_f + N_u\beta_s\alpha_s}{\beta_f\gamma_f + \beta_s\gamma_s}}\right) \tag{15.9}$$

and

$$\frac{dI_u}{dt} = I_u(N_u - I_u)(\beta_f\gamma_f' + \beta_s\gamma_s') - (N_i\beta_f\alpha_f + N_u\beta_s\alpha_s)I_i$$
$$= ((\beta_f\gamma_f' + \beta_s\gamma_s')N_u - (N_i\beta_f\alpha_f + N_u\beta_s\alpha_s))I_u$$
$$\times \left(1 - \frac{I_u}{N_u - \frac{N_i\beta_f\alpha_f + N_u\beta_s\alpha_s}{\beta_f\gamma_s' + \beta_s\gamma_f'}}\right) \tag{15.10}$$

where α_f and α_s are the purging rates of friends and strangers (i.e., the probability that one data will be purged out from the buffer at each contact).

For the logistic differential (15.9) and (15.10), since $\beta_f\gamma_f + \beta_s\gamma_s$, $N_i\beta_f\alpha_f + N_u\beta_s\alpha_s$, $\beta_f\gamma_f' + \beta_s\gamma_s'$, and $N_i\beta_f\alpha_f + N_u\beta_s\alpha_s$ are larger than 0, as long as $\frac{(\beta_f\gamma_f + \beta_s\gamma_s)N_i}{N_i\beta_f\alpha_f + N_u\beta_s\alpha_s}$ and $\frac{(\beta_f\gamma_f' + \beta_s\gamma_s')N_u}{N_i\beta_f\alpha_f + N_u\beta_s\alpha_s}$ exceeds one, the endemic equilibrium of (15.9) and (15.10) can be reached when $\frac{dI_i}{dt} = 0$ and $\frac{dI_u}{dt} = 0$, respectively.

Therefore, in these two cases, $1 - \frac{I_i}{N_i - \frac{N_i\beta_f\alpha_f + N_u\beta_s\alpha_s}{\beta_f\gamma_f + \beta_s\gamma_s}}$ is equal to 0 and $1 - \frac{I_u}{N_u - \frac{N_i\beta_f\alpha_f + N_u\beta_s\alpha_s}{\beta_f\gamma_s' + \beta_s\gamma_f'}}$ is equal to 0, which means,

$$I_i = N_i - \frac{N_i\beta_f\alpha_f + N_u\beta_s\alpha_s}{\beta_f\gamma_f + \beta_s\gamma_s} \tag{15.11}$$

and

$$I_u = N_u - \frac{N_i\beta_f\alpha_f + N_u\beta_s\alpha_s}{\beta_f\gamma_f' + \beta_s\gamma_s'} \tag{15.12}$$

Table 15.2 shows some numerical results based on our analysis. The results indicate that FdSs and FdSd tend to diffuse and buffer data among nodes that are not interested. Therefore, nodes use more buffer space to hold uninteresting data. However, in FsSd and FsSs, as data has high priority to be diffused between interested nodes, most data copies are at the interested nodes. FsSd differs from FsSs in that it also has high probability to diffuse one particular data item between any two strangers, which brings in more data copies at its uninterested nodes. Since most

Table 15.2 Numerical results of data distribution based on the S-I-S analysis model ($N_i = 20$, $N_u = 80$)

	$\alpha_f/\alpha_s = 0.1/0.9$			$\alpha_f/\alpha_s = 0.2/0.8$			$\alpha_f/\alpha_s = 0.3/0.7$			$\alpha_f/\alpha_s = 0.4/0.6$			$\alpha_f/\alpha_s = 0.5/0.5$		
	I_i	I_u	Delay	I_i	I_u	Delay	I_i	I_u	Delay	I_i	I_u	Delay	I_i	I_u	Delay
FsSd	19	48	4.63	17	52	5.09	14	57	5.57	11	62	6.07	7	67	6.27
FdSs	0	75	5.60	0	73	5.75	0	70	6.00	0	68	6.18	7	67	6.27
FsSs	18	0	10.00	15	27	6.07	12	49	7.07	10	60	6.60	7	67	6.27
FdSd	2	79	5.32	3	77	5.46	5	74	5.68	6	71	5.92	7	67	6.27

queries are initiated by the interested nodes and friends are easier to meet with each other, as reported in Table 15.2. Even though FsSd results in fewer data copies than FdSs and FdSd, it can still serve queries faster. Since FsSd has more data copies at the uninterested nodes than FsSs, it can serve the query from uninterested nodes more quickly. We can also observe that as the purging rate becomes skewer, the advantage of FsSd becomes more obvious. This is because when the purging rates become unequal between interested nodes and uninterested nodes, most queries can be served quickly by the interested nodes according to (8), (11), and (12).

15.5 Performance Evaluations

In this section, we evaluate the proposed diffusion schemes with real traces. We first study the infinite buffer case and then study the finite buffer case.

15.5.1 Experiment Setup

To evaluate different diffusion schemes, we use two well-known traces: the Cambridge Haggle Trace [11] and the MIT Reality Mining Trace [12]. In the Cambridge trace, 41 Intel iMotes were distributed to students attending the Infocom student workshop in Miami, 2005. They collected information such as when they meet with each other or any other external new devices. The trace covers 3 days. The MIT trace consists of 100 users carrying Nokia 6600 smart phones over more than nine months. The details of the two experimental traces are briefly summarized in Table 15.3. We extract the contact information from both traces to identify direct contacts between nodes where data diffusion could have taken place. The trace files are divided into discrete sequential contact events which are fed into our experiments. Each time a contact is observed, the node makes a diffusion decision based on the diffusion scheme.

The interest distribution of each node is generated based on its contact rate with other nodes. We assure that each pair of nodes share more common interests if they have higher contact frequency. Due to the randomness of node activity, in each

Table 15.3 Characteristics of the two experimental traces

Experimental trace	*Cambridge* Infocom'05	*MIT* Reality
Device	iMotes	Smart Phones
Network type	Bluetooth	Bluetooth
Duration	3/7/2005∼3/10/2005 (3 days)	7/10/2004∼5/5/2005 (9 months)
Granularity	120 s	300 s
# of devices	41 internal, 233 external	97 internal
# of contacts	28,216	113,902

experiment we characterize the contact rate of each node pair with the first half of the trace. We count the contact times of each node pair and adjust their interest distributions so that two nodes can have more common interests when they meet more often. After that we use the second half of the trace to evaluate the proposed schemes. More specifically, in the Cambridge trace, we use the trace from time 21,703 to time 108,098 as the training dataset and run the experiments with the trace from time 108,106. In the MIT trace, the first half of the trace (from 2004/7/10, 15:57 to 2004/11/19, 11:52) is used to predict node relationship and the remainders are fed into the diffusion schemes. We generate 1,000 queries which are uniformly distributed in the whole experiment period. Following Pareto's Rule [35], 80% queries are initiated by interested nodes and other 20% are initiated by uninterested nodes. We also divide the whole experiment period into n ($n = 15$ in *Cambridge* trace and $n = 14$ in *MIT* trace) statistical sessions to record the query delay results. We use the average delay of all queries in each session to represent the query delay of that statistical session. In order to overcome the finiteness of the traces, if the initiated query is not served before the end of the trace, the same meeting pattern is applied by re-feeding the trace from the beginning. Each experiment is repeated 10 times with different random seeds to eliminate randomness.

15.5.2 The Infinite Buffer Case

Figure 15.8 compares the data diffusion speed and data access delay as a function of time for the four schemes. As shown in the figure, with infinite buffer, the number of infected nodes increases and the query delay decreases as time goes. However, the diffusion speed and query delay under different scheme is different. As shown in Fig. 15.8(a), data can be diffused to its interested nodes more quickly in FsSd and FsSs than that in FdSs and FdSd. This result is consistent with our numerical analysis in which FsSd and FsSs have faster diffusion speed among interested nodes. Since FsSd and FsSs assign high diffusion priority among interested nodes and nodes sharing similar interests are more likely to meet with each other, the data can easily spread out among its interested nodes. Instead, FdSs and FdSd diffuse data slowly among interested nodes but they have a faster diffusion speed

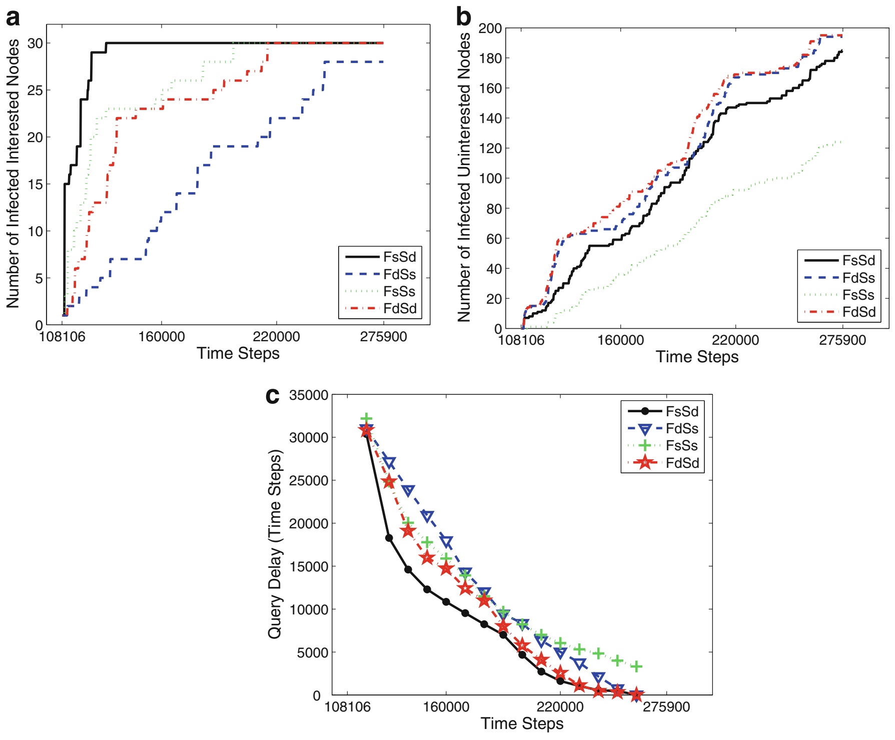


Fig. 15.8 Results for the Cambridge Infocom'05 trace (with infinite buffer). (**a**) Number of Infected Interested Nodes (**b**) Number of Infected Uninterested Nodes (**c**) Query Delay

among uninterested nodes as shown in Fig. 15.8(b). This is because friends diffuse different data first in FdSs and FdSd, and the data has more opportunity to be diffused between two uninterested friends. Further, FdSd can speed up data diffusion between two strangers who are not interested in the data because two strangers diffuse different data first in FsSd.

As shown in Fig. 15.8(c), as the number of data copies increases, queries for these data can be served more quickly. Since FsSd diffuses data faster among interested nodes, it has the shortest query delay compared with other schemes. For example, at time 160,000, the query delay of FsSd is about 26% less than FdSd, 32% less than FsSs, and about 40% less than FdSs. At time 220,000, the performance difference is more obvious. The query delay of FsSd is about 37% and 73% less than FdSd and FsSs, respectively, and 68% less than FdSs. Note that FsSs has the longest delay because more than 100 uninterested nodes are still uninfected in FsSs when the experiment finishes.

Figure 15.9 presents comparison results based on the MIT Reality trace. Again, FsSd achieves the best performance in terms of data diffusion speed and query delay. It has the fastest diffusion speed among interested nodes and the shortest query

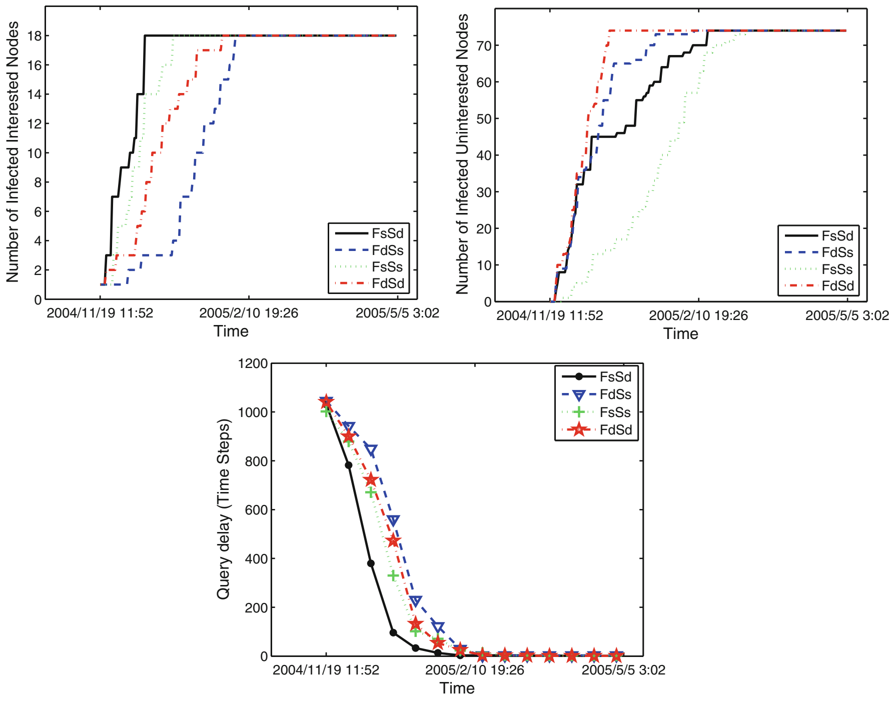


Fig. 15.9 Results for the MIT Reality Mining trace (with infinite buffer). (**a**) Number of Infected Interested Nodes (**b**) Number of Infected Uninterested Nodes (**c**) Query Delay

delay, which is consistent with the results of the Cambridge trace, and the analytical results in the last section. We notice that the diffusion speed is much slower in the Cambridge trace than that in the MIT trace. This is because most nodes in the Cambridge trace are external nodes, which do not appear regularly in the network.

15.5.3 The Finite Buffer Case

With finite buffer, some data may be replaced if the buffer is full. As shown in Fig. 15.10(a) and Fig. 15.10(b), the number of infected interested nodes and the number of infected uninterested nodes fluctuate at different time. This fluctuation comes from the fact that the data can be diffused among nodes and can be removed from the buffer as well. When the data item is buffered, the number of infected nodes increases. If the node's buffer is full, some data item has to be removed. Then, the node returns to the susceptible status and the number of infected nodes decreases.

In FsSd, each node prefers buffering its interesting data rather than uninteresting data. As long as a data item is buffered at its interested node, it will not be removed

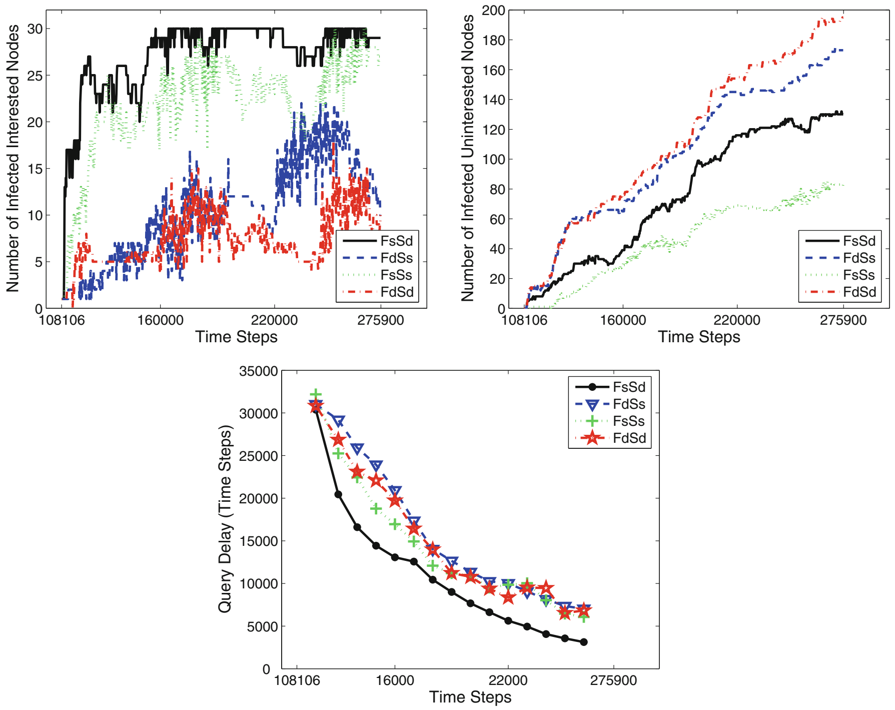


Fig. 15.10 Results for the Cambridge Infocom'05 trace (with finite buffer). (**a**) Number of Infected Interested Nodes (**b**) Number of Infected Uninterested Nodes (**c**) Query Delay

most likely. Thus, there are always more infected interested nodes in FsSd. FdSs and FdSd are different. They diffuse different data and remove similar data first between friends. Consequently, there will not be many data copies at the interested nodes. However, FdSs and FdSd give high priorities to diffuse and buffer data in the uninterested nodes. Hence, they have more data copies in uninterested node than FsSd and FsSs. Even though there are fewer infected uninterested nodes in FsSd, FsSd still outperforms FdSs and FdSd in terms of query delay because it helps diffuse data to the interested nodes. As shown in Fig. 15.10(c), FsSd can reduce up to 60% query delay compared to the other three schemes.

Figure 15.11 shows comparisons based on the MIT trace. The results are similar to that of the Cambridge trace. Because the MIT trace logs fewer nodes, but with more activities, the prediction on the contact rate with the first half of trace data is more accurate, and thus making FsSd perform better.

By comparing Fig. 15.8(c) to Fig. 15.10(c), we can see that the query delay in the infinite buffer case is much shorter than that in the finite buffer case, and the difference becomes more obvious as time goes. Similar results can be seen by comparing Fig. 15.9(c) and Fig. 15.11. This is because data may be purged out when

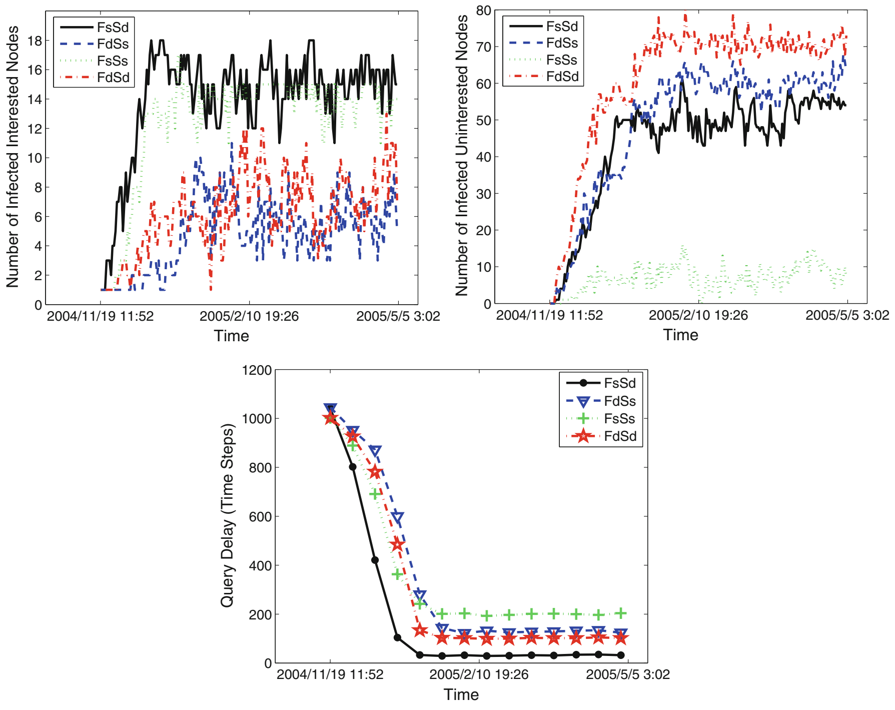


Fig. 15.11 Results for the MIT Reality Mining trace (with finite buffer). (**a**) Number of Infected Interested Nodes (**b**) Number of Infected Uninterested Nodes (**c**) Query Delay

the buffer is full in the finite buffer case. As a result, the delay for the finite buffer case is longer than that in the infinite buffer case. Similarly, the data fusion speed is also higher in the infinite buffer case than that in the finite buffer case.

15.5.4 Discussion

It is worth noticing that although FsSd has the best performance among the four schemes, its diffusion probability between two uninterested friends is still low, which slows down the diffusion speed among uninterested nodes. This is because in FsSd, friends first choose the data that is more similar to their common interests to diffuse, which prevents the diffusion of their uninteresting data (γ_f' is set to a small value in Table 15.1). To diffuse one data item quickly between uninterested friends, some changes have to be made in FsSd. For example, each pair of friends have to make different diffusion decisions on their interesting data and uninteresting data, i.e., to diffuse similar data first for the interesting data and to diffuse different data first for uninteresting data.

However, this modified scheme may not be practical, because it is impossible to tell whether the interesting data or uninteresting data is more important and a node cannot treat interesting data and uninteresting data separately. Further, according to the modified scheme, all data items have the same diffusion priority. Then, suppose one data item could have the diffusion privilege at all nodes, all data will have high diffusion priority, which is also impossible in a competition system. Although FsSd does not have the fastest diffusion speed among all nodes, it can diffuse data to the interested nodes quickly, which helps reduce the overall query delay.

15.6 Conclusions

In this chapter, we studied the performance of different data diffusion schemes in delay tolerant mobile ad hoc networks (DT-MANETs). We introduced four possible social-aware data diffusion schemes and developed theoretical models to analyze their performance in terms of data diffusion speed and query delay. Based on the analysis, we found an interesting result: to achieve better performance, a node should first diffuse the data most similar to their common interest when it meets a friend, and it should first diffuse the data most different to their common interest when it meets a stranger. To verify the theoretical result, extensive experiments have been carried out based on real traces of human contacts, and the experimental results are consistent with our theoretical analysis.

To the best of our knowledge, our work is the first to study data diffusion instead of data forwarding/routing using sociological knowledge. In this initial effort, of course, we have not addressed all relevant problems. In the future, we will look into other techniques to measure interest similarity. We will also investigate how to integrate data forwarding and data diffusion.

Acknowledgements This work was supported in part by Network Science CTA under Grant W911NF-09-2-0053.

References

1. S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *SIGCOMM*, pages 145–158, 2004.
2. W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc*, pages 187–198, 2004.
3. T. Small and Z. J. Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *MobiHoc*, pages 233–244, 2003.
4. A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6): 606–620, 2007.

5. E. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc*, pages 32–40, 2007.
6. P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social based forwarding in delay tolerant networks. In *MobiHoc*, 2008.
7. P. Costa, C. Mascolo, M. Musolesi, and G.P. Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 26(5):748–760, June 2008.
8. J. Ghosh, S. J. Philip, and C. Qiao. Sociological orbit aware location approximation and routing (solar) in manet. *Ad Hoc Netw.*, 5(2):189–209, 2007.
9. J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *INFOCOM*, 2006.
10. M. McPherson, L. Smith-Lovin, and J. Cook. Birds of a feather: Homophily in social networks. In *Annual Review of Sociology*, pages 15–44, 2001.
11. Cambridge Haggle Project. http://www.haggleproject.org/.
12. MIT Realisty Mining Project. http://reality.media.mit.edu/.
13. M. Motani, V. Srinivasan, and P. S. Nuggehalli. Peoplenet: engineering a wireless virtual social network. In *MobiCom*, pages 243–257, 2005.
14. W. Gao, Q. Li, B. Zhao, and G.Cao. Multicasting in delay tolerant networks: A social network perspective. In *MobiHoc*, 2009.
15. F. Bai and A. Helmy. Impact of mobility on last encounter routing protocols. *SECON*, pages 461–470, June 2007.
16. W. Gao, and G. Cao. On Exploiting Transient Contact Patterns for Data Forwarding in Delay Tolerant Networks. In *IEEE International conference on network protocols(ICNP)*, 2010.
17. Q. Li, S. Zhu, and G. Cao. Routing in socially selfish delay tolerant networks. In *INFOCOM*, 2010.
18. Y. Zhang, J. Zhao, G. Cao, and C. Das. On interest locality in content-based routing for large-scale manets. In *IEEE 6th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 178–187, 2009.
19. A. Miklas, K. Gollu, K. Chan, S. Saroiu, K. Gummadi, and E. Lara. Exploiting social interactions in mobile systems. In *UbiComp*, 2007.
20. T. Karagiannis, J. Boudec, and M. Vojnović. Power law and exponential decay of inter contact times between mobile devices. In *MobiCom*, pages 183–194, 2007.
21. Y. Wang, B. Krishnarnachari, and T. Valente. Findings from an empirical study of fine-grained human social contacts. In *The Sixth International Conference on Wireless On-Demand Network Systems and Services (WONS)*, pages 141–148, 2009.
22. W. Hsu, D. Dutta, and A. Helmy. Mining behavioral groups in large wireless lans. In *MobiCom*, pages 338–341, 2007.
23. J. Zhao and G. Cao. VADD: vehicle-assisted data delivery in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 57(3):1910–1922, May 2008.
24. Y. Zhang, J. Zhao, and G. Cao. Roadcast: a popularity aware content sharing scheme in vanets. In *IEEE ICDCS*, pages 223–230, 2009.
25. S. Kapadia, B. Krishnamachari, and S. Ghandeharizadeh. Static replication strategies for content availability in vehicular ad-hoc networks. *Journal of Mobile Network and Applications (MONET)*, 14(5):590–610, 2009.
26. S. Ghandeharizadeh and S. Kapadia. An evaluation of location-demographic replacement policies for zebroids. In *IEEE Consumer Communications and Networking Conference (CCNC)*, Jan 2006.
27. C. Boldrini, M. Conti, and A. Passarella. Contentplace: social-aware data dissemination in opportunistic networks. In *MSWiM*, pages 203–210, 2008.
28. V. Lenders, G. Karlsson, and M. May. Wireless ad hoc podcasting. In *SECON*, pages 273–283, 2007.
29. X. Zhang, G. Neglia, J. Kurose, and D. Towsley. Performance modeling of epidemic routing. *Comput. Netw.*, 51(10):2867–2891, 2007.

30. K. Lee, M. Le, J. Haerri, and M. Gerla. Louvre: Landmark overlays for urban vehicular routing environments. *IEEE WiVeC*, 2008.
31. Q. Yuan, I. Cardei, and J. Wu. Predict and relay: an efficient routing in disruption-tolerant networks. In *MobiHoc*, pages 95–104, 2009.
32. RDF Core Working Group http://www.w3.org/RDF/.
33. Web Services Description Language (WSDL) Version 2.0 http://www.w3.org/TR/wsdl20.
34. S. Kullback and R. A. Leibler. On information and sufficiency. In *Annals of Mathematical Statistics 22: 79-86.*, 1951.
35. W. J. Reed. The pareto, zipf and other power laws. In *Economics Letters*, pages 15–19, 2001.

Chapter 16
Security and Privacy in Online Social Networks: Optimization Perspectives

Ling Ding, Hongjie Du, and Weili Wu

Abstract Recently, Online Social Networks (OSNs) becomes one of the most remarkable technologies in the twenty-first century since it has been extraordinarily popular with over 200 million users. Security and privacy problems are the most important issues in OSNs. In this chapter, we introduced the optimization of security and privacy problems in OSNs. We characterized three existing works with different targets to give a view of this problem.

16.1 Introduction

Online Social Networks (OSNs) is one of the most remarkable technologies in the twenty-first century since it has been extraordinarily popular with over 200 million users. Through OSN applications (e.g. Facebook, Myspace, and Twitter), users can share their information such as photographs, phone numbers, with their friends. OSNs have already attracted much attention by some very popular Web sites [19]. As the technology matures, more applications are likely to emerge. It is also likely that social networking will play an important role in the future personal and commercial online interaction, as well as the location and organization of information and knowledge. Examples include browser plug-ins to discover information viewed by friends [20, 24], and social network based, cooperative Web search tools [18]. Even major Web search companies are deploying services that leverage social networks, like Yahoo!s MyWeb 2.0 [26] and Google Co-op [8].

Unlike the Web [12], which is largely organized around content, OSNs are organized around users. Participating users join a network, publish their profile and any content, and create links to any other users with whom they associate.

L. Ding (✉) • H. Du • W. Wu
Department of Computer Science, University of Texas at Dallas, USA
e-mail: ling.ding@utdallas.edu; hongjiedu@utdallas.edu; weiliwu@utdallas.edu

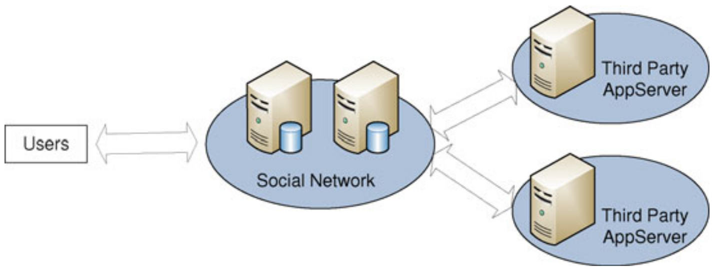


Fig. 16.1 A typical social network architecture: third party apps. are hosted on remote servers, and are accessed via the social network

The resulting social network provides a basis for maintaining social relationships, for finding users with similar interests, and for locating content and knowledge that has been contributed or endorsed by other users.

It is well known that the commercial success of OSNs relies heavily on the number of users they attract [6]. Thus, there is pressure on OSNs providers to encourage design and behaviour which increase the number of users and their connections. Sociologically, the natural human desire to connect with others, combined with the multiplying effects of Social Network (SN) technology, can make users less discriminating in accepting "friend requests". Users are often not aware of the size or nature of the audience accessing their profile data and the sense of intimacy created by being among digital "friends" often leads to disclosures which are not appropriate to a public forum. Moreover, recent work has proposed the use of social networks to mitigate email spam [5], to improve Internet search [18], and to defend against Sybil attacks [11].

Figure 16.1 depicts the architecture of a typical OSN like Facebook, which supports third-party applications run on remote servers. The social graph and user data are stored at the OSN site in a cluster or a "cloud". Third party applications run on their own servers using the API provided by the OSN, store and process application content locally, but interact with users through the OSN. Facebook Application Platform and OpenSocial are two popular examples of platforms with this architecture. The threats to privacy they identify in this section are common to these centralized architectures, as well as distributed architectures used by Tribler [22], FTN [10], and SocialSearch [18].

Recently, as more and more social network data has been made publicly available [1,2,13,14], preserving privacy in publishing social network data becomes an important concern.

A social network is a special graph structure made of entities and connections between these entities. These entities, or nodes, are abstract representations of either individuals or organizations that are connected by one or more attributes. The connections, or edges, denote relationships or interactions between these nodes. Connections can be used to represent financial exchange, friend relationships, conflict likelihood, web links, sexual relationships, disease transmission (epidemiology), etc. Although studying social networks has wide applications and attracted more and more attentions in recent years, they still face the challenge of achieving a reasonable tradeoff between securing the confidential information associated with the social networks and maximizing the benefits from the social network analysis. These threats against privacy of the social networks promote us to develop social network privacy oriented -preserving techniques.

A fundamental feature of social networks is the relationship graph that connects users. This graph enables two individuals to find the relationship paths that connect them. These paths are useful to express trustworthy users: nearby people (with short relationship paths connecting them) often deserve a higher level of trust. The path discovery mechanism can be used as a building block for many social networking applications: (1) Discovering a relationship path to a recruiter may boost the chances of a job applicant to get the position; vice-versa, discovering a relationship path to an applicant could help the recruiter get a more trusted judgment on the applicant. (2) Relationship path discovery can provide a basis for access control mechanisms suitable for Social Networks, where users determine the authorized users based on their distance to themselves in the social network. (3) A path to a person submitting an online review can boost confidence in the review. (4) Ensuring the receiver of an email that the sender is nearby in her social network can help avoid falsely flagging the email as spam. Although the relationship graph is at the core of the usefulness of social networks, personal relationships represent sensitive, private information that can also be misused. A primary concern is the unwelcome linkage among users. For example, two professionals employed by rival companies that have a connection may trigger suspicion. Or, connections of innovators and venture capitalists could alert the competition by giving leads to upcoming technological developments. Or, simply, a social relationship can correspond to a sensitive personal real-world relationship. Of greater concern is the discovery of entire relationship paths and, in the end, of the entire graph. A significant negative consequence of this discovery is the large-scale targeting, tracking, and monitoring of multiple individuals in real life based on discovered relationship paths. Other privacy concerns can arise from graph operations; e.g., user de-anonymization through merging of relationship graphs [15].

So, when we try to optimize the security and privacy issues in OSNs, we usually map the model to a graph, and use the graph theory to solve the problem. There are four parts in OSNs (Sect. 16.2), and current research mainly focuses on protecting data or profile of Data owner. In this chapter, we introduce some existing works with good citation. Each of them focuses on different target and has different model. We put an summary section at the front of each problem to show the pros and cons between their works and other works.

16.2 Online Social Network Model

In [25], OSN is composed by four parts – Credential authority, Storage site, Data Owner, and Member.

16.2.1 Credential Authority

Credential authority is in charge of cryptographically initializing an OSN domain and issuing a public/private key pair to each user in the domain. An OSN domain consists of a credential authority and all the registered users. It is necessary for users to possess legitimate credentials (i.e., key pairs) in order to perform security operations in the domain. A service provider acts as the credential authority in the OSN, e.g., the administrator in the Facebook social network. The credential authority is generally trusted by the OSN users and is provided with the users identity information (e.g., email address) upon registration.

16.2.2 Storage Site

Storage site is a third-party provider that offers free or priced mass storage space to accommodate user data possibly from multiple OSN applications or domains. The storage site is not trusted by the OSN users because it is not directly run by the OSN. The reason that we assume the untrusted third-party storage in favor of trusted proprietary storage (owned by the OSNs), is to take a more hostile and challenging environment into account when carrying out our security design.

16.2.3 Data Owner

Data owner or group manager, is an OSN user who shares personal or private data within his/her groups of contact, controls access of the group members to the private data, and adds/removes users from his/her groups. Hereafter, we use group to represent all contacts of a data owner who classifies these contacts (or the group) into different subgroups, based on the contacts social relationships with the data owner.

16.2.4 Member

Member is an OSN user and a contact of one or more data owners subgroup. The member may take on a different role in each data owners subgroup (e.g., ones classmate and anothers family). The member is meanwhile the data owner of his/her own group. The trust relationship between the data owner and a member is based on the social relationship of the two. For example, one trusts the family but may not trust a friend made in the online chatting room.

16.3 Privacy-Preserving Graph Algorithms in the Semi-honest Model

16.3.1 Summary

In Sect. 16.3, they introduced privacy-preserving protocols that enable two honest but curious parties to compute All Pairs Shortest Distance (APSD) and Single Source Shortest Distance (SSSD) on their joint graph. A related problem is how to construct privacy-preserving protocols for graph comparison. Many of these problems (e.g., comparison of the graphs respective maximum flow values) reduce to the problem of privacy-preserving comparison of two values, and thus have reasonably efficient generic solutions.

Their algorithm for APSD was new when they proposed, while the SSSD algorithm is a privacypreserving transformation of the standard Dijkstras algorithm. They also show that minimum spanning trees can be easily computed in a privacy-preserving manner.

They show that how the graph theory like shortest path tree and minimum spanning tree can be used in privacy issues in OSN.

16.3.2 Definition of Privacy

They [3] use a simplified form of the standard definition of security in the static semi-honest model due to Goldreich [7] (this is the same definition as used, for example, by Lindell and Pinkas [15]).

Definition 16.1 (computational indistinguishability). Let $S \subseteq \{0,1\}^*$. Two ensembles (indexed by S), $X = X_{\omega\{\omega \in S\}}$ and $Y = Y_{\omega\{\omega \in S\}}$ are computationally indistinguishable (by circuits) if for every family of polynomial-size circuits, $D_{nn \in N}$, there exists a negligible (i.e., dominated by the inverse of any polynomial) function $\mu : N \mapsto [0,1]$ so that

$$|Pr[D_n(\omega, X_\omega) = 1] - Pr[D_n(\omega, Y_\omega) = 1]| < \mu(|\omega|)$$

In such a case, they write $X \equiv Y$.

Suppose f is a polynomial-time functionality (deterministic in all cases considered in this section), and π is the protocol. Let x and y be the parties respective Privacy-Preserving Graph Algorithms in the Semi-honest Model 239 private inputs to the protocol. For each party, define its *view* of the protocol as $(x, r^1, m_1^1, \ldots, m_k^1)$ (respectively, $(y, r^2, m_1^2, \ldots, m_l^2)$), where $r^{1,2}$ are the parties internal coin tosses, and m_j^i is the jth message received by party i during the execution of the protocol. They will denote the ith parties view as $view_i^\pi(x,y)$, and its output in the protocol as $output_i^\pi(x,y)$.

Definition 16.2. Protocol π securely computes deterministic functionality f in the presence of static semi-honest adversaries if there exist probabilistic polynomialtime simulators S_1 and S_2 such that

$$S_1(x, f(x,y))_{x,y\in\{0,1\}^*} \equiv view_1^{\pi}(x,y)_{x,y\in\{0,1\}^*}$$

$$S_2(x, f(x,y))_{x,y\in\{0,1\}^*} \equiv view_2^{\pi}(x,y)_{x,y\in\{0,1\}^*}$$

where $|x| = |y|$.

Informally, this definition says that each parties view of the protocol can be efficiently simulated given only its private input and the output of the algorithm that is being computed (and, therefore, the protocol leaks no information to a semi-honest adversary beyond that revealed by the output of the algorithm).

16.3.3 Privacy-Preserving Algorithms on Joint Graphs

They now present their constructions that enable two parties to compute algorithms on their joint graph in a privacy-preserving manner. Let G_1 and G_2 be the two parties respective weighted graphs. Assume that $G_1 = (V_1, E_1, \omega_1)$ and $G_2 = (V_2, E_2, \omega_2)$ are complete graphs on the same set of vertices, that is, $V_1 = V_2$ and $E_1 = E_2$. Let $\omega_1(e)$ and $\omega_2(e)$ represent the weight of edge e in G_1 and G_2, respectively. To allow incomplete graphs, the excluded edges may be assigned weight ∞. They are interested in computing algorithms on the parties joint minimum graph $gmin(G_1, G_2) = (V, E, \omega_{min})$ where $\omega_{min}(e) = \min(\omega_1(e), \omega_2(e))$, since minimum joint graphs seem natural for application scenarios.

16.3.3.1 Private All Pairs Shortest Distance

The APSD problem is the classic graph theory problem of finding shortest path distances between all pairs of vertices in a graph (see, e.g., [4]). They will think of APSD(G) as returning a complete graph $G' = (V, E', \omega')$ in which $\omega'(e_{ij}) = d_G(i, j)$and V is the original edge set of G. Here, $d_G(i, j)$ represents the shortest path distance from i to j in G. This problem is particularly well suited to privacy-preserving computation because the solution leaks useful information that can be used by the simulator. To motivate the problem, consider two shipping companies who are hoping to improve operations by merging so that they can both take advantage of fast shipping routes offered by the other company. They want to see how quickly the merged company would be able to ship goods between pairs of cities, but they do not want to reveal all of their shipping times (and, in particular, their inefficiencies) in case the merger does not happen. In other words, they wish to compute $APSD(G)$ where $G = gmin(G_1, G_2)$. The basic idea behind their construction is to build up the solution graph by adding edges in order from shortest to longest. The following algorithm takes as input the parties complete graphs G_1 and G_2. The graphs may be directed or undirected, but they must have strictly positive weight functions.

1. For notational convenience they introduce a variable k, initially set to 1, that represents the iteration count of the algorithm. Color each edge in E "blue" by *letting* $B^{(k)}$ denote the set of blue edges in the edge set E at iteration k, and *setting* $B^{(0)} = E$. Let $R^{(k)}$ denote the set of "red" edges, $R^{(k)} = E - B^{(k)}$. The lengths of red edges have reached their final values and will not change as the algorithm proceeds, while the lengths of blue edges may still decrease.
2. A public graph $G_0^{(0)} = (V, E, \omega_0^{(0)})$ is created. Its edges are all initially weighted as $\omega_0^{(0)} = \infty$. When the algorithm terminates after n iterations, they will have $\omega_0^{(n)}(e_{ij}) = d_G(i, j)$ and $B^{(n)} = \emptyset$.
3. The parties compute the following public value
$$m_0^{(k)} = \min_{e \in B^{(k-1)}} \omega_0^{(k-1)}(e) \tag{16.1}$$
and the respective private values
$$m_1^{(k)} = \min_{e \in B^{(k-1)}} \omega_1(e), and \tag{16.2}$$
$$m_2^{(k)} = \min_{e \in B^{(k-1)}} \omega_2(e) \tag{16.3}$$
4. Now the parties privately compute the length of the smallest blue edge among all three graphs, $m^{(k)} = \min(\min(m_1^{(k)}, m_0^{(k)}), \min(m_2^{(k)}, m_0^{(k)}))$, using a generic protocol for private minimum. This protocol does not reveal the larger value.
5. The parties form the following public set
$$S_0^{(k)} = \{e | w_0^{(k-1)}(e) = m^{(k)}\} \tag{16.4}$$
and the respective private sets
$$S_1^{(k)} = \{e | w_1(e) = m^{(k)}\}, and \tag{16.5}$$
$$S_2^{(k)} = \{e | w_2(e) = m^{(k)}\} \tag{16.6}$$
By construction, $S_0^{(k)}$, $S_1^{(k)}$, and $S_2^{(k)}$ contain only blue edges.
6. First, the parties *privately* compute the set union $S^{(k)} = S_0^{(k)} \bigcup S_1^{(k)} \bigcup S_2^{(k)}$. This is done using the privacy-preserving set union algorithm from section. Next, the

color of each edge $e \in S^{(k)}$ is changed from blue to red by setting $B^{(k)} = B^{(k-1)} - S^{(k)}$. Define a weight function $\omega_0^{'(k)}$ by

$$w_0^{'(k)}(e) = \begin{cases} m^{(k)}, & if\ e \in S^{(k)}, \\ w_0^{(k-1)}(e), & otherwise. \end{cases} \tag{16.7}$$

7. Examine triangles with an edge $e_{ij} \in S^{(k)}$, an edge $e_{jk} \in R^{(k)}$, and an edge $e_{ik} \in B^{(k)}$. Define the weight function $\omega_0^{(k)}$ by fixing these triangles if they violate the triangle inequality under $\omega_0^{'(k)}$. More precisely, if $\omega_0^{'(k)}(e_{ij}) + \omega_0^{'(k)}(e_{jk}) < \omega_0^{'(k)}(e_{ik})$, then define $\omega_0^{(k)}(e_{ik}) = \omega_0^{'(k)}(e_{ij}) + \omega_0^{'(k)}(e_{jk})$. Do the same for triangles with an edge $e_{ij} \in R^{(k)}$, an edge $e_{jk} \in S^{(k)}$, and an edge $e_{ik} \in B^{(k)}$.
8. If there are still blue edges, go to step 3. Otherwise stop; the graph $G_0^{(k)}$ holds the solution to $APSD(G)$.

16.3.3.2 Private All Pairs Shortest Path

While there is only a single APSD solution for a given graph, there may be many all pairs shortest path solutions, because between a pair of points there may be many paths that achieve the shortest distance. As a side effect of engaging in the protocol described in Sect. 16.3.3.1, the two participants learn an APSP solution. When defining the weight function w(k) 0 by fixing violating triangles in $\omega_0^{'(k)}$ during step 7, a shortest path solution may be associated with the fixed edge. Specifically, if $\omega_0^{'(k)}(e_{ij}) + \omega_0^{'(k)}(e_{jk}) < \omega_0^{'(k)}(e_{ik})$, then the shortest path from i to k is through j.

In step 6 of subsequent iterations, when adding an edge $e_{ij} \in S^{(k)}$ to the set of blue edges, they can conclude that the shortest path from i to j is the edge e_{ij} itself if $e_{ij} \in S_0^{(k)}$, or is the shortest path solution as computed above if $e_{ij} \in S_0^{(k)}$.

Note that learning this APSP solution does not imply any violation of privacy, as it is the APSP solution implied by the APSD solution.

16.3.3.3 Private Single Source Shortest Distance

The SSSD problem is to find the shortest path distances from a source vertex s to all other vertices [4]. An algorithm to solve APSD also provides the solution to SSSD, but leaks additional information beyond that of the SSSD solution and cannot be considered a private algorithm for SSSD. Therefore, this problem warrants its own investigation. Similar to the protocol of Sect. 16.3.3.1, the SSSD protocol on the minimum joint graph adds edges in order from smallest to largest. This protocol is very similar to Dijkstras algorithm, but is modified to take two graphs as input.

1. Set $\omega_1^{(0)} = \omega_1$ and $\omega_2^{(0)} = \omega_2$. Color all edges incident on the source s blue by putting all edges e_{si} into the set $B^{(0)}$. Set the iteration count k to 1.

2. Both parties privately compute the minimum length of blue edges in their graphs.

$$m_1^{(k)} = \min_{e_{si} \in B^{(k-1)}} \omega_1^{(k-1)}(e_{si}),$$

$$m_2^{(k)} = \min_{e_{si} \in B^{(k-1)}} \omega_2^{(k-1)}(e_{si})$$

3. Using the privacy-preserving minimum protocol, compute

$$m^{(k)} = \min\left(m_1^{(k)}, m_2^{(k)}\right)$$

4. Each party finds the set of blue edges in its graph with length $m^{(k)}$.

$$S_1^{(k)} = \left\{e_{si} | \omega_1^{(k-1)}(e_{si}) = m^{(k)}\right\}, \textit{ and}$$

$$S_2^{(k)} = \left\{e_{si} | \omega_2^{(k-1)}(e_{si}) = m^{(k)}\right\}$$

5. Using the privacy-preserving set union protocol, compute

$$S^{(k)} = S_1^{(k)} \bigcup S_2^{(k)}$$

6. Color the edges in $S^{(k)}$ red by setting $B^k = B^{(k-1)} - S^{(k)}$. Define a weight function $\omega_1^{'(k)}$ by

$$w_1^{'(k)}(e) = \begin{cases} m^{(k)}, & if\ e \in S^{(k)}, \\ w_1^{(k-1)}(e), & otherwise. \end{cases} \tag{16.8}$$

and a weight function $\omega_2^{'(k)}$ by

$$w_2^{'(k)}(e) = \begin{cases} m^{(k)}, & if\ e \in S^{(k)}, \\ w_2^{(k-1)}(e), & otherwise. \end{cases} \tag{16.9}$$

7. Similar to the APSD algorithm, form the weight function $\omega_1^{(k)}$ by fixing the triangles in w$\omega_1^{'(k)}$ that violate the triangle inequality and contain edges in $S^{(k)}$. $\omega_2^{(k)}$ is likewise formed from $\omega_2^{'(k)}$.

If there are still blue edges remaining, go to step 2. Otherwise stop; both parties now have a graph with each edge incident on s colored red, and with the weight of these edges equal to the shortest path distance from s to each vertex.

16.3.3.4 Minimum Spanning Tree

Suppose that two frugal telephone companies wish to merge. Each company has a cost function for connecting any pair of houses, and they want to connect every house as cheaply as possible using the resources available to the merged company.

In other words, they wish to compute $MST(gmin(G_1, G_2))$. If they can perform this computation privately, then both companies can see the final result without revealing their entire cost functions. Both Kruskals and Prims algorithms for MST are easily turned into private protocols using their techniques, because the algorithms already consider edges in order from smallest to largest. At each iteration, Kruskals algorithm adds the shortest edge such that its addition does not form a loop. It is a simple task for each party to compute the set of edges which would not form loops, and then to privately compute the length of the shortest edge in this set. One problem arises when there are multiple edges that share this length. In the shortest path algorithms, they addressed this issue by adding all edges of appropriate length at the same time using the private set union protocol, but this will not work for MST. Instead, they can assign a canonical ordering to the edges, and at each step find the shortest length edges that are canonically first. This will allow a simulator to determine, given the final MST, in what order the edges arrived.

16.3.4 Complexity Analysis

For each algorithm considered in this section, they calculate the number of rounds, the total communication complexity, and the computational complexity, and compare them with the generic method. Using Yaos method on a circuit with m gates and n inputs requires $O(1)$ rounds, $O(m)$ communication, and $O(m+n)$ computational overhead. Lindell and Pinkas note in [15] that the computational overhead of the n oblivious transfers in each invocation of Yaos protocol typically dominates the computational overhead for the m gates, but for correct asymptotic analysis they must still consider the gates.

Complexity of privacy-preserving APSD. For their analysis they will assume that the edge set E has size n, and that the maximum edge length is l. The generic approach to this problem would be to apply Yaos Method to a circuit that takes as input the length of every edge in G_1 and G_2, and returns as output $G = APSD(gmin(G_1, G_2))$. Clearly, such a circuit will have $2n \log l$ input bits. To count the number of gates, note that a circuit to implement Floyd–Warshall minimums and $O(n^{3/2})$ additions. For integers represented with $\log l$ bits, both of these functionalities require $\log l$ gates, so they conclude that Floyd–Warshall requires $O(n^{3/2} \log l)$ gates. To compute gmin requires $O(n \log l)$ gates, but this term is dominated by the gate requirement for Floyd–Warshall. They conclude that the generic approach requires $O(1)$ rounds, $O(n^{3/2} \log l)$ communication, and $O(n^{3/2} \log l)$ computational overhead.

The complexity of their approach depends on the number of protocol iterations k, which is equal to the number of different edge lengths that appear in the solution graph. In iteration i, they take the minimum of two $(\lg l)$-bit integers, and compute a set union of size s_i. Because each edge in the graph appears in exactly one of the set unions, they also know that $\Sigma_{i=1}^{k} s_i = n$.

First, they will determine the contribution to the total complexity made by the integer minimum calculations. If they use Yaos protocol, then each integer minimum requires a constant number of communication rounds, $O(\lg l)$ inputs, and $O(\lg l)$ gates, so the k calculations together contribute $O(k)$ rounds, $O(k \lg l)$ communication complexity, and $O(k \lg l)$ computational complexity.

Complexity of privacy-preserving SSSD. Complexity of SSSD is similar to that of APSD, except that the number of rounds is $k = O(v)$ and the total number of set union operations is v, where v is the number of vertices $(O(e^{1/2}))$. They conclude that their protocol requires $O(v)$ rounds, $O(v(\log v + \log l))$ oblivious transfers, and $O(v(\log v + \log e))$ gates. A generic solution, on the other hand, would require $O(v^2 \log l)$ oblivious transfers.

16.4 Privacy Preserving in Social Networks Against Sensitive Edge Disclosure

16.4.1 Summary

In this section, they [17] emphasize edge weight privacy instead focus on preserving either node or edge privacy like other researchers did. Data owners may not want to release the exact weight of each edge, but would like to keep the shortest paths of a set of nodes and the lengths of the corresponding shortest paths as unperturbed as possible, for the data analysis purpose.

In this section, they consider preserving weights (data privacy) of some edges, while trying to preserve close shortest path lengths and exactly the same shortest paths (data utility) of some pairs of nodes without adding or deleting any edge and node.

In fact, edge weights, reflecting affinity between two nodes in many cases, relate the expenses or frequency between two persons or similarity between two organizations. The edge weights in the network are more realistically assigned on a practical scale. The shortest path is important to be preserved in a social network for the following reasons. (1) Previous work is mostly on the unweighted graph. Their work is mostly focused on de-identification of nodes or edges. (2) The weighted graph is quite popular. One of the things people care about in this type of graphs is the shortest path between every pair of nodes. The shortest path is a major data utility which has a wide application such as physical location search in GIS, min-delay path problem in telecommunications midset, and optimal Analog circuits in VLSI (very large scale integration). (3) In essence, a weighted graph is a generalization of the unweighted graph. Their algorithms might be generalized and extended to unweighted graph cases.

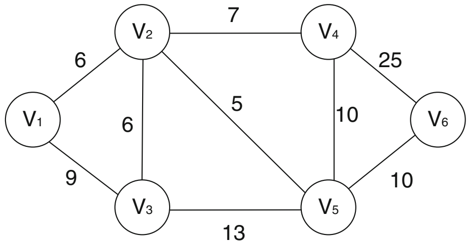


Fig. 16.2 A simple social network G

16.4.2 Notation

A social network in this section is defined as an undirected and weighted graph $G = (V, E, W)$. Figure 16.2 is a simple social network. The nodes of the graph, V, may denote meaningful entities from the real world such as individuals, organs, organizations, communities, and so on (in Fig. 16.2, $V = v_1, v_2, v_3, v_4, v_5, v_6$). E is the set of all undirected but weighted edges. The edge weight between node i and node j is $w_{i,j}$ (the value beside an edge is the weight in Fig. 16.2). All $w_{i,j}$ compose the set W. The cardinalities of V and E, $||V||$ and $||E||$, are the number of nodes and edges in this social network, respectively, (in the example, $||V|| = 6$ *and* $||E|| = 9$). They assume that $n = ||V||$, $m = ||E||$. Since the graph G is undirected, $w_{i,j}$ is equal to $w_{j,i}$. So the adjacency weight matrix of G is symmetric. Although the following perturbation strategies are all based on the undirected graph and symmetric adjacency weight matrix, they can be easily modified with respect to directed graphs and the corresponding nonsymmetric adjacency weight matrices.

Let $w^*_{i,j}$ be the perturbed weight of the edge between node i and node j, $d_{i,j}$ and d$d^*_{i,j}$ be the shortest path lengths between node i and node j before and after a perturbation strategy, respectively, $p_{i,j}$ and $p^*_{i,j}$ be the shortest paths between node i and node j before and after a perturbation strategy.

16.4.3 Greedy Perturbation Algorithm

In a static social network, they may easily collect some necessary information about this social network for their analysis and privacy-preserving purpose.

They assume that not all shortest paths of node pairs in a social network are considered to be significant (in the real world, it is not reasonable that all information is considered as confidential).

Then, in a social network $G = V, E, W(||V|| = n)$, they generate a shortest path matrix P and the corresponding length $n * n$ matrix D. In the matrix P, each entry p_{s_1,s_2} is a linked list representing the shortest path between v_{s_1} and v_{s_2}. For example, $p_{1,6} = (1 \to 2 \to 5 \to 6)$, it shows that the shortest path $p_{1,6}$ successively passes through v_1, v_2, v_5 *and* v_6. In the matrix D, each $d_{s1,s2}$ is the length of the shortest path connecting v_{s_1} and v_{s_2}. In the following contents, all node pairs $(s_1; s_2)$ of p_{s_1,s_2} and d_{s_1,s_2} are in the set H unless otherwise stated explicitly.

So, their goal is to generate a perturbed graph $G^* = V^*, E^*, W^*$ which satisfies the following conditions:

1. $V^* = V$ and $E^* = V$
2. maximize $\sum_{i,j} |w_{i,j} - w^*_{i,j}|$
3. $p^*_{s_1,s_2} = p_{s_1,s_2}$, for every pair (s_1, s_2) in H
4. $d^*_{s_1,s_2} \approx d_{s_1,s_2}$, for every pair (s_1, s_2) in H.

16.5 StarClique: Guaranteeing User Privacy in Social Networks Against Intersection Attacks

16.5.1 Summary

Recently, several social graph anonymization algorithms are proposed to enable public release of social graphs without compromising user privacy [9, 16, 27]. The main goal here is to prevent attackers from identifying a user or a link between users based on the graph structure. There are, however, some key differences that set apart their work. First, in the graph anonymization problem also, similar to the work on databases, the attacker is outside the system. In this section, they consider a stronger attacker that is an active participant of the network (or online) with abilities to perform multiple queries and use the results to improve the attack. Second, the definition of privacy breach is different in the two cases. In graph anonymization, a user privacy is breached if either a user is identified in the anonymized graph, or a link between two users is established. Their goal, however, is to prevent attackers from linking the data transmitted by applications with the users. Given the abundance of the application data as well as the social graph, it is more challenging to provide anonymity guarantees. Finally, the solutions proposed by prior graph anonymization work [9, 27] provide global privacy properties (as in, create k identical neighborhoods, or k identical degree nodes in the graph, etc.). These global properties do not ensure that each node in the network has sufficient degree to defend against the intersection attack.

In this section, they studied privacy risks involved in sharing data in todays social content-sharing applications due to compromised user accounts. They identify the social intersection attack, a low-cost privacy attack that can be used by two or more compromised users to identify the source of shared data objects in all content-sharing applications. It effectively links data objects with their owners relying only the social graph topology and the data shared by the applications. This attack invalidates naive solutions to mitigate privacy risks.

Social networks can provide their users with privacy guarantees in the form of k-anonymity by adding new edges to the social graph. They identify a graph structure we call Star-Clique, and prove that it is the minimal structure necessary to provide each user with k-anonymity. A privacy-conscious OSN provider can build StarCliques around each user, and utilize several optimizations to dramatically reduce the cost of new edges. This type of “graph evolution” is practical for todays social content-sharing networks, and provides sufficient flexibility for OSN operators to make local decisions about the privacy and overhead tradeoff.

16.5.2 Background

Naturally formed social graphs tend to exhibit power-law degree distributions and high skew in node connectivity. Local clustering is limited, and the lack of common friends makes users vulnerable to the social intersection attack. Their solution to this problem is to “evolve” the graph by adding “privacy buddies” to users such that all users have *k*-anonymity, for some value of k chosen by the OSN operator. Adding these buddies creates latent edges between buddies and users.The real and latent edges together provide *k*-anonymity. The evolved graph with privacy guarantees is used by the applications to transfer data between users, but this evolved graph is never revealed to the users directly. As a result, attackers do not know the list of friends sending data to them and cannot identify the exact source of the data they receive. The only change existing social networks need to do, to use their solution, is to evolve the graph, and send the evolved graph to the application servers instead of the real social graph.

16.5.3 Assumptions, Goals and Attacker Model

This section lists their [23] assumptions, goals, and the attackermodel for this section.

16.5.3.1 Assumptions

They make two simple assumptions in their design. First, they assume that the OSN operators and third-party application servers are secured by the owners and do not compromise their users privacy. These sites have significant financial incentives to keep their service secure: To attract and retain their users. The end users, on the other hand, may be lax in applying security patches and hence be compromised due to various malware attacks. Second, their privacy mechanisms are irrelevant if user identities can be deduced directly from shared data. So they assume that all data is scrubbed to remove identifiable user information. This scrubbing can happen before the data leaves a trusted endpoint. Similarly, they assume that the attackers cannot cross correlate application data with out-of-band information to identify its owner, as was done in recent NetFlix privacy attack [21].

16.5.3.2 Goals

Their goal is to provide three key properties to all users in the network irrespective of their social connectivity.

1. *Provable k-Anonymity*. They aim to provide k-anonymity to social application users. k-anonymity provides source anonymity and the data receiver cannot tell the source even with social intersection attack. Formally, k-anonymity is:

   Definition 16.3. The system provides k-anonymity to the source (x) of an event ξ, if the probability that the attackers assign for x to be the source of ξ is less or equal to $1/k$. In other words, the attackers suspect at least k different nodes to be the likely sources of ξ, with equal probability.

2. *LowOverhead.* It is necessary to add *minimal* number of latent edges to reduce the additional overhead on the social infrastructure due to processing and data transfer of cover traffic along the latent edges.
3. *Preserve Relevance of Cover Traffic.* The latent edges added should connect nodes that are *close in social distance*, so that the cover traffic is still relevant to users. Nodes that are farther apart have fewer "similarities" in interests and connecting them might send highly irrelevant data to users.

16.5.3.3 Attacker Model

In the social application setting they consider, they assume the following attacker model:

1. A fraction (p) of the one-hop friends of a given user x are compromised. They can work both independently, and in collusion to compromise honest users privacy.
2. The attackers have the entire social graph. They know their local graph, and can crawl the rest of the graph.
3. They assume that only the attackers within one hop from a user x collude together to break xs privacy via passive intersection attacks.

This is a stronger attack model compared to prior work on graph anonymization [9, 16, 27] as the attackers here use both the application data and the social graph to attack. In addition, passive attacks are harder to detect compared to active attacks. For example, an active attacker can delete all but one of her friends, and assign the new data received to that friend. However, such attacks will be easily detected. Finally, note that the actual number of malicious nodes around a node x depends on its degree (d_x) and the fraction p. They use f to represent the number of malicious neighbors of a node throughout the section, but f is node specific, and $f = [d_x p]$.

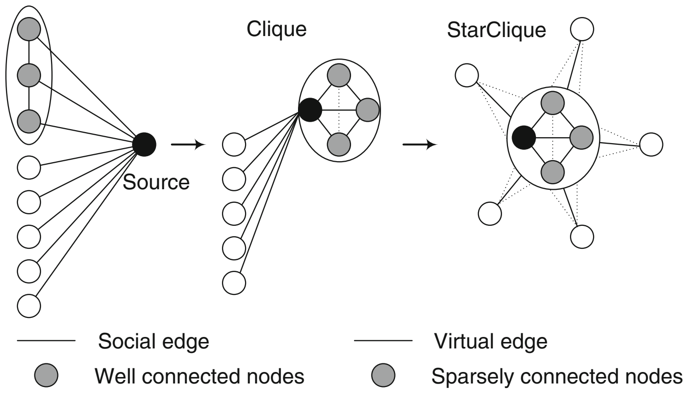


Fig. 16.3 The graph evolution process for a node. The node first selects a subset of its neighbors. Then it builds a clique with the members of this subset. Finally, it connects the clique members with all the non-clique members in the neighborhood. Latent or virtual edges are added in the process

16.5.4 Graph Evolution

This section shows the details of their social graph evolution mechanisms. First they introduce the StarClique graph structure, and then present a simple algorithm to evolve the graph.

16.5.4.1 StarClique Structure

Figure 16.3 depicts the StarClique structure and its formation. There are two main parts in the structure: The portion to the right (in the central sub-figure) is the clique, and the portion to the left is called the Star. Let the node that is evolved be x, and let f be the number of attackers around x. A StarClique is built around x using its neighborhood nodes. The clique for x consists of x along with its $(k+f-1)$ neighbors, that together form a $(k+f)$-clique. The Star consists of the one-hop friends of x that do not belong to the clique. Each member of the Star is connected with all $(k+f-1)$ members of the clique.

StarClique provides two key properties: (a) StarClique provides provable k-anonymity to node x against f one-hop colluders, and (b) StarClique has the minimal number of edges necessary to provide k-anonymity against f one-hop colluders.

Table 16.1 Table (Notation used in this paper)

$G=(V,E)$	Graph definition
x, y, z	Nodes $\in G$
$N(x)$	Set of nodes in the neighborhood of the node x
$g.neighbors(x,i)$	Set of all neighbors of x at most i hop away in g
d_x	Degree of the node x
p	Fraction of malicious one-hop neighbors
f	Number of malicious neighbors of x ($f=\lceil d_x p \rceil$)
C	Subset of nodes in $N(x)$

16.5.4.2 Evolution Algorithm

With this background in mind, they describe the evolution algorithm. The evolution algorithm works on one node (say x) at a time, and it works in three steps. The first step in evolving a node x is to identify the closest neighborhood of x that has at least $(k+f-1)$ nodes in it. Evolution starts with one-hop neighborhood, and moves to two-hop, etc. Selecting nearest neighbors first ensures that the privacy buddies are closer in social distance. The second step is to select a subset of $(k+f-1)$ nodes from the neighborhood, and build a $(k+f)$-clique out of them by adding latent edges. These clique members are selected at random in this simple algorithm. The final step is to connect the members of xs neighborhood that do not belong to the clique with all members of the clique by adding latent edges. This process forms a structure as shown in Fig. 16.3. As a result, xs k-anonymity is preserved. They analyze the security properties of this algorithm in more detail later. The Evolve Graph algorithm is presented in Algorithm 1, and the notations used are listed in the Table 16.1.

16.5.5 Optimizing the Evolution Algorithm

Here, we present several optimizations that significantly reduce the number of new latent edges added to the graph during evolution. They apply their optimizations to *Evolve_Graph*, and present an optimized algorithm called *Optimized_Evolve_Graph* (shown in Algorithm 1) that is annotated to show where each optimization is applied. The intuition behind the optimization and analysis are introduced.

Optimization 1: Select Clique. While selecting the clique members from the neighborhood, choosing the most well-connected $(k+f-1)$ nodes, instead of random nodes, reduces the latent edges added significantly. Select Clique function in the Algorithm 2 implements this optimization, where the well-connected nodes are chosen based on the number of friends shared between the nodes in the neighborhood and x. This selection leads to clique reuse in the neighborhood, reducing the new edges added.

Algorithm 1 Evolution Algorithm: evolves the input graph g to produce an evolved graph g'.

```
    Graph $g'$ = Evolve Graph (Graph $g$)
 1: $g' = g$
 2: /* Copy the original graph $g$ into the evolved graph $g'$ */
 3: for $x \in V$ do
 4:     $N(x) = \phi; i = 1$
 5:     while $|N(x)| < (k+f-1)$ do
 6:         $N(x) = N(x) \bigcup g.neighbors(x,i)$
 7:         /* Neighborhood is selected in the original graph */
 8:         $i = i+1$
 9:     end while
10:     $C$ = select $(k+f-1)$ random nodes from $N(x)$
11:     $C = C \bigcup \{x\}$
12:     $Build_Clique(g',C)$
13:     /* Edges added in $g'$ to build the clique structure around $x$ */
14:     $Build_Star(g',N(x),C)$
15:     /* Edges added in $g$ to build the star structure around $x$ */
16: end for
17: Return $g'$
```

Algorithm 2 Optimized Evolution Algorithm: evolution algorithm annotated with the optimizations.

```
    Graph $g'$ = Evolve_Graph_Optimized (Graph $g$)
 1: $g' = g$
 2: $S(x) = \{V' nodes sorted in the decreasing order of degree\}$
 3: Applying Optimization 3: Ordered Evolution Above
 4: for $x \in S$ in decreasing order of degree do
 5:     $N(x) = g.neighbors(x,1); i = 1$
 6:     $g$ (instead of $g'$) is used above to handle the side-effects of Edge Reuse
 7:     while $|N(x)| < (k+f-1)$ do
 8:         $N(x) = N(x) \bigcup g'.neighbors(x,i)$
 9:         Applying Optimization 2: Edge Reuse above
10:         if $|N(x)| > (k+f-1)$ then
11:             Applying Optimization 4: Limit to $k$ Friends here
12:         end if
13:         $i = i+1$
14:     end while
15:     $C = Select_Clique(N(x))$
16:     Applying Optimization 1: Select Clique above
17:     $C = C \bigcup \{x\}$
18:     $Build_Clique(g')$
19:     $Build_Star(g',N(x),C)$
20: end for
21: Return $g'$
```

Optimization 2: Edge Reuse. Before evolving a node x, this optimization considers xs most recent and evolved state, instead of xs connectivity in the original graph, which includes the latent edges of x. This reduces the new edges added: as evolution progresses, more and more latent edges are added, and the connectivity of nodes around x increase. This means that xs neighborhood is more connected than in the original graph and hence the number of new edges necessary to evolve x is reduced significantly. This optimization implies that Algorithm 1 should use the evolved graph g in the loop instead of the original graph g.

Optimization 3: Ordered Evolution. Instead of evolving the nodes in random order, evolving nodes from highest degree to lowest degree leads to fewer latent edges overall. The intuition here is that if a supernode is evolved first, the edges added to provide k-anonymity to this supernode, and the clique formed, can be potentially reused by a *majority* of the low-degree nodes attached to the supernode.

Optimization 4: Limit to k Friends. If a node x has $< k + f - 1$ nodes, the unoptimized algorithm considers the larger neighborhood incrementally one hop at a time. It is quite likely that when the neighborhood increases by one hop, the neighborhood size goes significantly beyond $k + f - 1$. However, all the nodes in this y-hop are not necessary to provide k-anonymity: we need only $l = (k + f - |g.neighbors(x, 1)|)$ additional nodes. Thus, we select only the l most well-connected nodes from outside the one-hop neighborhood in this optimization.

Optimized Evolve Graph. The evolution process is depicted in Fig. 16.3, and Algorithm 1 shows the pseudo-code for *Optimized_Evolve_Graph*. In this algorithm, first, the nodes are sorted by their degree, and evolved in the order of their degree, starting from the highest. Second, evolution is applied on the evolved graph repeatedly C this applies the edge reuse optimization. Indeed, we use the original graph g to get the original degrees, and the evolved graph g to maximize the number of reused edges during the neighbor selection. This is necessary to handle the side-effects of edge reuse optimization, as described before. When the node has less than $k + f - 1$ friends, its neighborhood in the evolved graph is selected. They apply the Limit to k Friends optimization at this step. Finally, Select Clique optimization is applied in this optimized algorithm while choosing the clique members out of the neighbors. StarClique is built for each evolved node as in *Evolve_Graph*.

16.5.6 Anonymity Analysis

This section has two main parts. First, we introduce formal notations and identify the conditions under which k-anonymity is preserved. Second, we present the properties of StarClique that are necessary to provide k-anonymity.

16.5.6.1 Formal Notations and *k*-Anonymity

They represent the social network as a graph, $G = (V, E)$, where each user is mapped to a unique vertex $\in V$ and the friend relationship between two users x and $y \in V$ is represented as an edge (x, y). V is the set of all vertices, and E is the set of all edges. Each edge is undirected, as it represents the friendship between two user. And an undirected edge $(x, y) \in E$ is equivalent to two directed edges (x, y) and (y, x), where (x, y) represents that x is a friend of y and (y, x) represents that y is a friend of x.

16.5.6.2 Privacy via the StarClique Structure

The evolution algorithm protects privacy by building Star-Clique around nodes. The first step to prove the k-anonymity property of evolution is to identify the necessary conditions that the StarClique structure has to satisfy in order to provide k-anonymity for a particular source (x). StarClique (Fig. 16.3) is constructed around x in two steps as follows: (1) Clique: Build a clique C of $k + f - 1$ nodes $\in N(x)$ around x. Note that the edges in C are bidirectional. (2) Star: The remaining nodes $\in N(x) \setminus C$ are connected to $k + f - 1$ nodes in the clique. Each edge in this step is directed from the clique nodes to the Star nodes. Directed edges are necessary only to prove the structure minimality. They next show a theorem, that StarClique guarantees k-anonymity, using the locally minimal connectivity between a source and its one-hop neighbors.

Theorem 16.1. *The StarClique has the minimal connectivity, in the one-hop neighborhood of a node, necessary to provide k-anonymity against f one-hop colluding neighbors.*

16.6 Discussion and Future Works

There is no doubt that OSNs benefit too much to the society – no matter providers or users. OSNs providers can get many benefits based on information they collect. This may lead to privacy problem to OSNs' users. As we said before, only a few works have been done in Credential authority, Storage site and Member parts. Most works focus on Data Owner part and barely optimization. Security and privacy are always important in different networks. How to optimize the solution of the security and privacy problems is still a good topic for us. We can put our eyes on other parts instead of Data Owner part to improve the security in OSNs.

References

1. L. Adamic and E. Adar (2005) How to search a social network. Social Networks, vol. 27, no. 3, pp. 187C203
2. L. Backstrom et al. (2006) Group formation in large social networks: membership, growth, and evolution. KDD
3. J. Brickell and V. Shmatikov (2005) Privacy-preserving graph algorithms in the semi-honest model. In ASIACRYPT, LNCS, pages 236C252
4. T. Cormen, C. Leiserson, and R. Rivest (1990) Introduction to Algorithms. MIT Press
5. S. Gariss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu (2006) Re: Reliable email
6. R. Gross and A. Acquisti (2005) Information revelation and privacy in online social networks. In WPES 05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society, pages 71C80, ACM.
7. O. Goldreich: Foundations of Cryptography (2004) Volume II (Basic Applications). Cambridge University Press
8. Google co-op. http://www.google.com/coop/.
9. M. Hay, G. Miklau, D.Jensen, D. Towsely and P. Weis (2008) Resisting structural re-identification in anonymized social networks. Proc. of VLDB
10. Q. Huang, H.J. Wang, and N. Borisov (2005) Privacy-Preserving Friends Troubleshooting Network. Proc. of NDSS
11. H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman (2006) Sybilguard: defending against sybil attacks via social networks. In SIGCOMM 06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, pages 267C278, New York, NY, USA, ACM.
12. J. M. Kleinberg (1999) Authoritative sources in a hyperlinked environment. J. ACM, 46(5):604C632
13. G. Kossinets and D. J. Watts (2006) Empirical analysis of an evolving social network. Science, vol. 311, no. 5757, pp. 88C90
14. R. Kumar et al. (2006) Structure and evolution of online social networks. KDD
15. Y. Lindell and B. Pinkas: Privacy preserving data mining (2002) J. Cryptology, 15(3):177–206
16. K. Liu, and E. Terzi:Towards identity anonymization on graphs (2008) Proc. of SIGMOD
17. L. Liu, J. Wang, J. Liu, and J. Zhang: Privacy preserving in social networks against sensitive edge disclosure (2008) Technical Report Technical Report CMIDA-HiPSCCS 006-08, Department of Computer Science, University of Kentucky, KY
18. A. Mislove, K.P. Gummadi, and P. Druschel (2006) Exploiting social networks for internet search. Proc. of HotNets
19. A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee (2007) Measurement and analysis of online social networks. In IMC 07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pages 29C42, New York, NY, USA, ACM.
20. Mozillacoop. http://www.mozilla.com.
21. A. Narayanan and V. Shmatikov (2008) Robust de-anonymization of large sparse datasets. Proc. of IEEE S&P.
22. J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. Van Steen, and H. Sips (2008) Tribler: a social-based peer-to-peer system. Concurrency And Computation
23. K. P. N. Puttaswamy, A. Sala and B.Y. Zhao (2009) Starclique: Guaranteeing user privacy in social networks against intersection attacks. Proc. of ACM CoNEXT
24. Stumbleupon. http://www.stumbleupon.com
25. J. Sun, X. Zhu, and Y. Fang (2010) A privacy-preserving scheme for online social networks with efficient revocation. Proc. of IEEE INFOCOM 2010, pages 1C9
26. Yahoo! myweb. http://myweb2.search.yahoo.com.
27. B. Zhou, and J. Pei (2008) Preserving privacy in social networks against neighborhood attacks. Proc. of ICDE

Chapter 17
A Social Network Based Patching Scheme for Worm Containment in Cellular Networks

Zhichao Zhu, Guohong Cao, Sencun Zhu, Supranamaya Ranjan, and Antonio Nucci

Abstract Recently, cellular phone networks have begun allowing third-party applications to run over certain open-API phone operating systems such as Windows Mobile, Iphone and Google's Android platform. However, with this increased openness, the fear of rogue programs written to propagate from one phone to another becomes ever more real. This chapter proposes a counter-mechanism to contain the propagation of a mobile worm at the earliest stage by patching an optimal set of selected phones. The counter-mechanism continually extracts a social relationship graph between mobile phones via an analysis of the network traffic. As people are more likely to open and download content that they receive from friends, this social relationship graph is representative of the most likely propagation path of a mobile worm. The counter-mechanism partitions the social relationship graph via two different algorithms, balanced and clustered partitioning and selects an optimal set of phones to be patched first as those have the capability to infect the most number of other phones. The performance of these partitioning algorithms is compared against a benchmark random partitioning scheme. Through extensive trace-driven experiments using real IP packet traces from one of the largest cellular networks in the US, we demonstrate the efficacy of our proposed counter-mechanism in containing a mobile worm.

Z. Zhu (✉) • G. Cao • S. Zhu
Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA
e-mail: zzhu@cse.psu.edu; gcao@cse.psu.edu; szhu@cse.psu.edu

S. Ranjan • A. Nucci
Narus Inc., Mountain View, CA 94043, USA
e-mail: soups@narus.com; anucci@narus.com

17.1 Introduction

Cellular phone networks are increasingly receptive to open-API operating systems such as Windows Mobile, Iphone and Google's Android running over mobile phones in the networks. While this openness would allow richer applications to run over mobile phones, it also makes it easier for hackers to write malicious software that can take control of a mobile device by exploiting its vulnerabilities or that of the applications running on top of it. In this regard, cellular networks may witness a similar evolution of worms as has been seen in the wired world. These mobile worms could impose unwarranted bandwidth charges to customers, deterioration in quality of service, and ultimately loss of revenue for service providers. Moreover, although it took over a decade for wireline worms to evolve to the current stage, it might take much less time for hackers to adapt existing techniques to mobile environments.

The usual ways for mobile worms to propagate include `Bluetooth` [14] interface and `Multimedia Messaging Service` (MMS) [13] interface. One Bluetooth based mobile worm is *Cabir* [9], which can spread through Bluetooth connection to other Bluetooth-enabled devices it can find. As its name suggests, MMS messages are intended to contain media content such as photos, audios or videos, but they can also contain infected malicious codes. One noteworthy example is *Commwarrior* [10], which is the first worm that can propagate via MMS. It searches through a user's local address book for phone numbers and sends MMS messages containing infected files to other users in the address book.

The increasing popularity and unique property of MMS worms draws our focus on dealing with MMS worms in this chapter. MMS worms could be sent out in just one click and travel to any mobiles all over the world with a larger chance of success in propagation, thus are potentially more virulent in terms of speed and area of propagation than Bluetooth worms. Note that worms that exploit plain-text Short Messaging Service (SMS) can not carry malicious payload, and hence usually only carry a URL in the message, from where the victim is lured to download the payload, e.g., the worm *Symbos/Feak* [8]. We consider worms that exploit SMS as similar to MMS in the way they spread (via address books or call records) and hence our methodology developed here would be applicable to both types of worms.

Due to characteristics of slow start and exponential propagation exhibited by mobile worms, it is challenging to detect a worm outbreak at the early stage while it is hard to mitigate it at a later stage. The heterogeneity of cellular networks also makes worm propagation speed at different spots variable. Given the extremely large scale and the distributed nature of mobile cellular networks, it is difficult to place monitors everywhere. Regularly reporting the traffic records to a central server by individual phones is also undesirable as users are not willing to be disturbed by any unrelated traffic. However, even if network operators are unable to detect a worm propagation during the earliest stage, they still have a window of opportunity to react before the worm spreads to a larger population. This is especially true for MMS worms in which users' interactions are required to download and install the malicious files on mobile devices. Therefore, unlike automatic Internet worms [25]

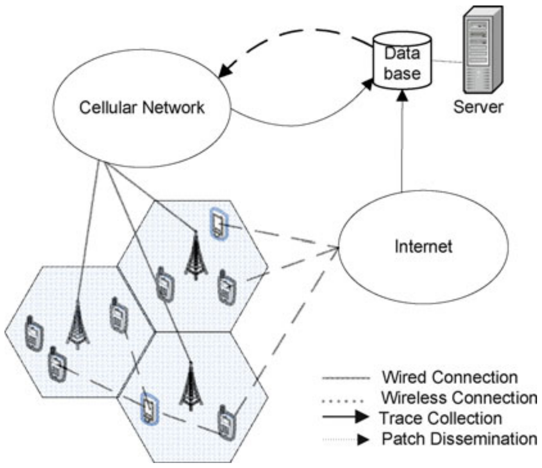


Fig. 17.1 The architecture of our social-based worm containment system

which only take hours to infect millions of users, it usually takes much longer for mobile worms to spread to a severe level. Defense techniques against Internet worms usually include rate limiting, filtering or patching [33] [26] [29]. However, they are not appropriate for mobile worms as they are prone to both false positives and false negatives. Moreover, filtering allows non-blacklisted infected phones to spread worms even faster. In this chapter, we focus on the methodology by which a mobile network operator would distribute a patch to arrest a worm's propagation before it causes complete network infection.

Patch propagation techniques have been developed for delivering worm signatures in the wired Internet [29]. However, such solutions are not directly applicable to mobile networks which have a unique constraint of lower data rates. In such a bandwidth-constrained environment, patches can not be propagated by a network operator to *all* devices at the same time. Moreover, patches would have to compete with the bandwidth already being consumed by a propagating worm. Existing work on modeling and containment of worms in a mobile network [2,3,11,23], do not take into account the unique capability of mobile worms which exploit social network of users by exploiting their address book or recent call records. In lieu of above observations, we take a hierarchical approach towards patching mobile devices such that those devices which act as a "bridge" between social clusters within the network are patched first. The intuition is that such devices once infected have the ability to infect entire social clusters and hence they must be patched first.

In this chapter, we propose a new approach to contain MMS worms within a limited range at the earliest stage. We divide the mobiles in cellular networks into multiple partitions based on the social relationships between mobiles retrieved from a real cellular network trace. Mobiles in each partition closely interact with each other while mobiles across different partitions are less related. Security patches are distributed to key nodes that separate individual partitions to block the worm propagation from one partition to another. The architecture of our social-based worm containment system is shown in Fig. 17.1. The trace, including both voice traffic and Internet data traffic, is collected through cellular networks and stored in a database for analysis. After processing, the generated security patches are disseminated through cellular networks to selected mobiles. More specifically, the contributions of our work are three-fold:

- We construct a social relationship graph of mobile devices by extracting their communication patterns based on a network trace. This graph describes the social relationships between mobile phones which are usually exploited by mobile worms for spreading.
- We propose a new containment strategy for MMS worms by partitioning the mobiles appropriately based on the social relationship graph. Two partitioning algorithms: *balanced partitioning* and *clustered partitioning* are proposed and their performance is evaluated.
- We experimentally compare our targeted patching algorithms (balanced and clustered) against a random patching strategy. Our experiments show the efficacy of targeted patching: both balanced and clustered patching algorithms achieve a lower infection rate than the random strategy while patching a significantly smaller number of nodes.

The rest of this chapter is organized as follows. Section 17.2 reviews the related work on mobile worms containment in cellular networks. Section 17.3 presents motivations behind the trace-driven partitioning approach. Section 17.4 describes how this social relationship graph can be built by using a network traffic trace. Section 17.5 introduces the graph partitioning theory and two corresponding patching schemes. Section 17.6 evaluates the performance of our worm containment strategy. Section 17.7 gives an extensive discussion of related issues. Finally, Sect. 17.8 concludes our work and provides future research directions.

17.2 Related Work

Defense techniques against Internet worms include rate limiting [33] or filtering [26]. Vojnovic et al [29] studied the efficacy of automatic patching countermeasure in protecting the Internet against scanning worms. Zou et al [38] used a Kalman filter to detect Internet worm's propagation at its early stage in real-time. However, these techniques are not directly applicable to the mobile network scenario.

There is limited work on mobile viruses/worms modeling and containment in the literature. Yang et al [34] applied a software diversity approach to deal with worm attacks in wireless sensor networks. Mickens and Noble [23] proposed a probabilistic queuing framework to model the propagation of mobile viruses over short-range wireless interfaces. Fleizach et al [11] evaluated the effects of malware propagating using communication services like VOIP and MMS in mobile phone networks. However, they did not use real traffic data in their worm propagation model. Khouzani et al [19] developed optimal decision rules to quantify the damage that the malware can inflict on the network as well as an intelligent defense strategy to limit the damage. Bose and Shin [2] applied two commonly used mechanisms: rate limiting and quarantine to the dynamically generated list of vulnerable clients in the mobile messaging network. Van Ruitenbeek et al [28] also investigate propagation of MMS/SMS malware and various responses. Zyba et al [39] studied the dynamics of proximity mobile phone malware that propagates by Bluetooth interface, and evaluated potential defenses against it. Li et al [21] proposed CPMC scheme which integrates short-term coping components and long-term evaluation components to deal with proximity malware. Miklas et al [24] used a trace-driven simulator to study the interactions between Bluetooth devices. They conclude that Bluetooth based worms would spread more widely by exploiting contacts between "strangers" instead of "friends". While our focus here is on worms which spread via MMS or SMS, the hypothesis driving our work is analogous – that to contain a worm, we must first detect and patch the devices which bridge social clusters. Meng et al [22] investigated the reliability of SMS by analyzing traces collected from a nationwide cellular network over a period of three weeks. Here, we exploit the social relationship graph from a real cellular network trace that includes a variety of services and use it to develop a worm containment mechanism.

Recently, Bose et al and Kim et al have proposed two techniques for using behavioral signatures [1] and power signatures [20] for locally detecting malware on mobile devices. Some other work tried to detect mobile virus at the network-level such as SmartSiren [3]. The aforementioned work is complimentary to our approach in that these mobile worm detection systems can detect a worm within a reasonable latency and hence could serve as the initial trigger for our worm containment via patch distribution mechanism. A preliminary version of this work has been presented in [37], and an introductory version can be found in [36]. Other security issues such as DoS attacks in the 3G network scenario have been studied in [5, 35].

17.3 Motivation

Mobile worms that spread using MMS [10] or SMS [8] typically exploit the social network of users to propagate from one mobile device to another. These worms search through a user's local address book and recent call records for phone numbers and send messages to other users. Note that randomly scanning does not work on mobile worm environment, as any malicious message from an untrusted stranger would not be opened and activated. In the case of MMS, the message itself could be the malicious payload, while in the case of SMS, the user would be lured to download the payload from a URL. A victim mobile receiving this message will most likely open and download the message since he believes it comes from someone he knows and trusts. Thus, an effective worm containment approach must take into account the social relationship graph between mobile devices in a cellular network. By figuring out the social interactions between mobile devices, i.e. which devices are more likely to exchange messages with each other, we can predict the propagation path of such mobile worms. In this way, the vulnerable mobiles or connections could be marked and be protected.

Given that there has not been any instance of mobile worm that has propagated far and wide across a cellular network "in the wild" as yet, there is limited knowledge of propagation paths of mobile worms. In this regard, we assume that the propagation path of a mobile worm can be approximated by the social network of mobile devices. Given that a user *Joe* has a higher probability to open and download a message from *Jane* with whom he periodically exchanges messages, this pair of users, *Joe–Jane* would be considered more vulnerable. In contrast, if *Joe* doesn't exchange messages with *Mary*, he is unlikely to be infected by a worm sent by *Mary* and hence the pair of users, *Joe–Mary* is considered less likely to be included in the worm's propagation path. In summary, we use the amount of traffic exchanged between two mobile devices as an indicator of whether this pair of devices would be present in a worm's propagation path. This propagation model would be reflective of worms that spread by exploiting the call records of infected hosts. Such a social relationship graph can be accurately built by a mobile network operator by looking at the calling and messaging records at which the operator stores for billing purposes. Even for mobile worms which spread by using the address book of an infected host, the social relationship graph built by using the calling and messaging records would be reflective of the propagation path of the worm, which is similar for worms which spread by randomly generating a hit list of potential devices. This is on account of the fact that humans are much more likely to open and download a message from someone with whom they have communicated in the past.

Our worm containment strategy would be implemented at a mobile service provider's messaging gateways or base-station controllers. Service providers typically store records of all traffic generated by a user per session for billing and accounting purposes. We use an anonymized trace from one of the largest cellular network providers over a two-week period in April 2008. The trace summarizes the total amount of traffic generated by every user for a variety of applications such as SMS, MMS, SIP based VoIP, Push-To-Talk and so on. We use all traffic exchanged between a pair of devices regardless of application types, as an indicator of their likelihood to infect each other.

We use the traffic trace to simulate the relational topology graph. In this graph, each vertex represents a mobile in the cellular network and each edge between two vertices represents that the two mobiles have communicated with each other in the past. This topology graph gives us an overview of how mobiles are related with each other and how worms might use these social relationships to propagate themselves.

With a knowledge of the social relationship graph, the next question is how to prevent a worm from propagating once it starts to breakout. Here, we use the social relationship graph to find an effective patch distribution strategy. A mobile that receives a patch becomes immune to the worm and could then be used to propagate the patch further. However, as we will discuss in Sect. 17.5, disseminating patches to all mobiles may not be a practical method due to the time and bandwidth limits. Thus, a faster way of patch dissemination, or an appropriate order of patch distribution is needed. Intuitively, the one with the highest risk to be infected or the one with the highest probability to infect others should have the highest priority for security upgrades. Under our partitioning based approach, security patches need not reach all the mobiles if the worms could be contained in each small partition. Therefore, only those key nodes that separate the graph into individual partitions should be patched in the first place. We next discuss how to determine this set of key nodes.

17.4 Trace-Driven Social Relationship Graph

In this section, we describe how a service provider can construct a social relationship graph by using an example traffic trace collected at the network layer at one of the largest mobile phone networks in the US. The endpoints present in the trace were anonymized while preserving the uniqueness of the identifiers of ip-addresses and phone numbers involved. The trace provides session-level information for traffic (bytes and packets) exchanged between two endpoints per application over a two-week period in April 2008. The trace contains information about 2 million users across 65000 base station cells all over the US. According to this trace, about 35% of users in this network exchange about 0.4 million MMS messages every day. Besides MMS, the trace also contains traffic volume information for SIP based VoIP sessions exchanged between users, SIP based Push-To-Talk and SMS.

Definition 17.1 (Cellular-Social Relationship Graph). An undirected weighted graph $G = (V, E)$ consists of a set of vertices V and a set of edges E, such that each vertex $u \in V$ denotes a mobile in the cellular network, while each edge $e(u, v)$ denotes that at least one traffic flow was exchanged between mobile u and v. Let d_u denote the degree of vertex u, $u \in V$ (the number of mobiles or vertices having a link with u). Let $m(u, v)$ denote the amount of traffic initiated from u to v. If there are functions f and g that map each vertex $u \in V$ and each edge $(u, v) \in E$ to a real number, then the graph is considered to be weighted with f and g determining the vertex-weights and edge-weights, respectively. The weight-mapping functions are as following:

$$f(u) = d_u \tag{17.1}$$

$$g(u, v) = m(u, v) + m(v, u) \tag{17.2}$$

An example of social relationship graph is shown in Table 17.1. In this example, we pick 9 mobile phones who interact with each other more or less from the trace, anonymize them as A to I. We use the number of sessions exchanged between two mobiles u and v over one week as our weight $m(u, v)$. Alternatively, the total number of bytes or packets exchanged between two mobiles could also be used as the weights. For the sake of generalization, we count all sessions exchanged between two mobiles regardless of the application type, as all types contribute to the worm propagation patterns. Each entry in the table shows how many times any

Table 17.1 Communication traffic records

Between mobile phones	WAT[a]	Normalized WAT[b]
A and *B*	4	1
A and *G*	12	3
A and *H*	12	3
B and *C*	8	2
B and *H*	4	1
B and *I*	4	1
C and *D*	40	10
C and *I*	4	1
D and *E*	6	1.5
D and *I*	6	1.5
E and *F*	20	5
F and *G*	4	1
F and *I*	20	5
G and *H*	8	2
G and *I*	4	1

[a] WAT: Weekly Averaged Traffic
[b] Divided by the minimum WAT over the week

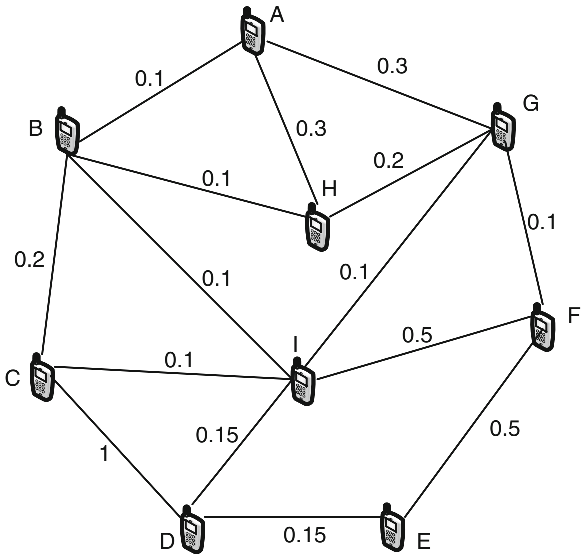


Fig. 17.2 Example of Cellular Social Relationship Graph. Each vertex in the graph denotes a mobile phone and the weight of each edge between two vertexes represents normalized WAT between the two mobile phones

two mobiles communicated with each other every week on an average. We note this metric as WAT (weekly averaged traffic). If we abstract each mobile as a vertex and normalize WAT between any two mobiles by dividing the minimum WAT over the week, we get a relationship graph as Fig. 17.2.

The weights of vertices and edges together contribute to a `significance level` which represents the chance of being infected by worms. As can be seen from (17.1), the weights of vertices depend on the node degrees. Intuitively, the mobile with the highest risk to be infected or infect others is the key node that a worm can use to spread and thus has the highest priority for containment. For MMS worms, a mobile with a higher in-degree means that it is more likely to be infected while a mobile with a higher out-degree is more likely to infect other mobiles. Therefore, those high-degree mobiles, either in-degree or out-degree, should be assigned a higher vertex weight and get higher priority for patching consideration. The in-degree and out-degree of a mobile are not necessarily dependent, but may be correlated. The phone number of the mobile that has large address book tends to appear in the address books of many others.

The social interactions [24] between mobiles can be used to explain (17.2). Whenever there is a traffic record between two mobiles, they have a chance to be friends and therefore a larger probability to open and activate a worm message received from each other. The more frequently they communicate with each other, they would be closer to each other which means a higher vulnerability. This social relationship graph gives us an overview of how mobiles are related with each other and how worms might use these social relationships to propagate themselves.

We use weekly averaged traffic (WAT) to measure the relationship between two mobiles. According to what we have observed from the trace, although the number of interactions between two individuals behaves differently for weekday and weekend, the number of interactions across the two weeks remains similar. This result which is also confirmed by [24] shows that people's interaction rate is predictable on a weekly basis. Therefore, it is reasonable to use a weekly averaged traffic information to represent the interaction rate through a long period.

17.5 Containing Worms by Graph Theory

Most security patch providers such as F-Secure [7] use push-based strategy for patch distribution, that is, as soon as a new security patch is available, the notification of updates is sent to all subscribed users. Upon receiving the notification, users authenticate and verify the message, and then connect to a centralized database to download the patch updates promptly. This can be achieved by short messages through control channels. However, the time to disseminate patches to entire cellular networks could be in the order of hours or days, which is much longer than the worm propagation period. Moreover, the bandwidth bottleneck of the control channel prevents all the mobiles from reaching the system and downloading the patches at the same time. According to [5], the total number of messages per second needed to saturate the cellular network capacity for a metropolitan area such as Washington D.C. is 240 msgs/sec and for the entire United States is 525,325 msgs/sec. Therefore, any larger traffic volume would cause congestion or even crash the network.

17.5.1 Uniform Patching vs. Targeted Patching

Therefore, an appropriate order or scheduling of patch distribution is needed. Intuitively, the one with the highest risk to be infected or the one with the highest probability to infect others should have the highest priority for security patches. Our goal is to find a small set of nodes with the highest priority for patching, while keeping the infection rate as low as possible. We call it *targeted patching*. Under the partition based scenario, security patches do not have to reach all the mobiles if the worm could be contained in each small partition. A small set of nodes which separate all the nodes into multiple partitions is enough for our targeted patching.

With the knowledge of the network topology, we partition the graph into as many separate pieces as possible and contain the worm propagation within each partition. These partitions are separated by a minimum set of key nodes called `separators`. The separators are chosen and patched by the network with the highest priority. As a result, the worm propagation can be blocked since an infected node inside its partition has to go through a separator to reach other partitions. Then, the worm containment problem becomes a graph problem and we can use graph-partition techniques to solve it. Now the question is what criteria should be used to partition the graph.

Based on the following two different partitioning strategies, there are two kinds of targeted patching: *balanced patching* and *clustered patching* (unbalanced patching).

17.5.2 Balanced Graph Partitioning

Intuitively, the significance level of each partition should be similar so that the worm damage to each partition can be balanced. As mentioned before, vertex weight and edge weight can be viewed as metrics for significance level. The vertex degree denotes how many victims an infected mobile is able to reach while the edge weight represents the probability that worms can propagate through this link successfully. Due to different ways of balancing these two metrics, we define balanced graph partitioning as follows.

Definition 17.2 (Balanced Graph Partitioning). Given an undirected weighted graph $G = (V,E)$, with weight $f(i)$ for each vertex $i \in V$ and $g(u,v)$ for each edge $(u,v) \in E$, a partition P cuts the vertices set V into $k(k > 1)$ subsets $V_1, V_2, \ldots, V_k$ such that $V_i \cap V_j = \phi$ for $i \neq j$, and $\cup_i V_i = V$, with the following two constraints satisfied:

- The total weights of vertices in each subset V_i are balanced.
- The total weights of all edges crossing any two subsets are minimized.

The first constraint in the definition requires the vertex weights for each partition to be balanced. Let LoadImbalance(P) denote the ratio of the highest partition weight over the average partition weight, i.e., $max_i(f(V_i))/(f(V)/k)$. The first constraint minimizes LoadImbalance(P). It tries to keep the significance level in each partition balanced, so that the damage to each partition is balanced and limited. The second constraint keeps the edge weights between partitions minimized so that partitions are less related to each other. Let Edge-Cut(P) denote the total weights of all edges crossing any two partitions. Then, the second constraint minimizes Edge-Cut(P).

Next, we try to find an appropriate theory to solve the above problem. Existing graph partitioning solutions [15, 27] are developed for high-performance parallel computing, circuit placement and other disciplines. All these solutions partition the vertices of the graph into equally weighted sets so that the weight of the edges crossing between sets is minimized. A new class of partitioning algorithms based on the multilevel paradigm [16, 30] has been developed and is considered to be the state-of-the-art as they provide extremely high-quality partitions. These algorithms are very fast, and can scale to graphs containing millions of vertices. The basic idea behind the multilevel approach is to first coarse down the graph G to a few hundred vertices or less. Then, some standard partitioning algorithm is used to partition the graph. Since the size of the graph is quite small, simple algorithms such as *Kernighan-Lin (KL)* [18] performs well. The final step is to project this partition back towards the original finer graph G. Some of these algorithms have also been incorporated into well-known software packages such as *METIS* [17].

These existing graph partitioning algorithms were originally designed for parallel computing, whose goal is to evenly distribute the computations over k processors by partitioning the vertices into k equally weighted sets while minimizing inter-processor communication represented by edges crossing between partitions. These two objectives exactly match the two constraints in our definition. Therefore, balanced graph partitioning can be easily solved by existing graph partitioning algorithms, for example, the multilevel KL algorithm.

17.5.3 Clustered Graph Partitioning

Balanced graph partitioning tries to maintain the significance level in each partition balanced, so that the damage to each partition is balanced and limited. However, it does not give high priority to minimize the edge-cut, therefore does not guarantee that worms can always be successfully contained within individual partitions. For example, if the weights of the edges across two partitions are very large, the probability of worm propagation through this edge will be very high. Then, the worms may have already propagated across the two partitions before patches are distributed. Therefore, rather than partitioning the graph into balanced parts, we want to partition the graph according to the trusted social relations. This method is referred to as *clustered partitioning* where edges within each partition have higher weights compared to the edges between the two partitions.

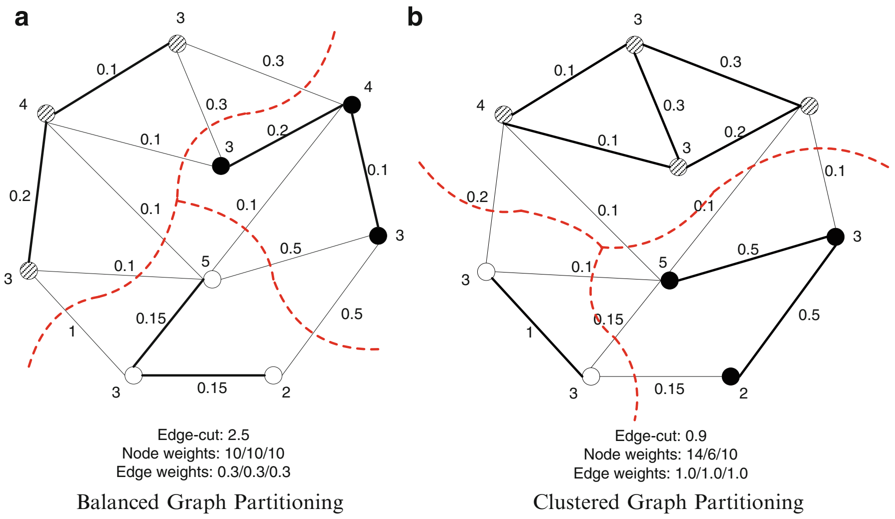

a
b
Edge-cut: 2.5
Node weights: 10/10/10
Edge weights: 0.3/0.3/0.3
Balanced Graph Partitioning
Edge-cut: 0.9
Node weights: 14/6/10
Edge weights: 1.0/1.0/1.0
Clustered Graph Partitioning

Fig. 17.3 Examples of two different graph partitioning schemes

With clustered partitioning, we keep the mobiles that are socially close to each other in the same partition, and divide nodes that are not close into different partitions. This is because closer nodes are more likely to infect each other quickly as soon as the worms breakout. We cannot do too much about it as the infection may have already happened before patching, so we prefer leaving them in the same partition. On the other hand, two nodes with a low weight link may have not communicated with each other and there is a low probability for a worm to spread across the link. Therefore, keeping them in two different partitions can effectively prevent the worm in one partition from infecting the other. Note that if there is no edge between the two nodes, they will be divided into two different partitions.

Definition 17.3 (Clustered Graph Partitioning). Given an undirected weighted graph $G = (V,E)$, with weight $f(i)$ for each vertex $i \in V$ and $g(u,v)$ for each edge $(u,v) \in E$, a partition P cuts the vertice set V into $k(k > 1)$ subsets $V_1, V_2, \ldots, V_k$ such that $V_i \cap V_j = \phi$ for $i \neq j$, and $\cup_i V_i = V$, with the following two constraints satisfied:

- The averaged edge weights (i.e., the total edge weights divided by the number of nodes) in each subset V_i are maximized: $max(\sum_{m \in V_i, n \in V_i} g(m,n))/|V_i|$.
- The total weights of all edges crossing subsets are minimized.

Figure 17.3 shows the node weights and edge weights for each partition by the two partitioning schemes on the social relationship graph shown in Fig. 17.2. We can see clearly from the example that balanced partitioning has an edge-cut of 2.5 while the clustered partitioning achieves an edge-cut of 0.9. As a result, it takes longer time for worms to propagate between partitions under clustered partitioning, which leaves itself more response time.

17.5.4 Measurement of Connectivity

Unfortunately, this problem is `NP-hard` and these two constraints cannot be achieved at the same time. Thus, we can only apply heuristics to generate approximate solutions. We develop a new measurement called `Connectivity` and propose a recursive clustered partitioning algorithm based on this concept.

Theorem 17.1. *The connectivity C in a social relationship graph is measured recursively as follows:*

Connectivity between two nodes. If i and j are two nodes and the edge between them has a weight of $w(i,j)$, then the connectivity between node i and j is $C(i,j) = w(i,j)$. If there is no edge between i and j, $C(i,j) = 0$.

Connectivity between a node and a set. S is a set with more than one node in it, and i is a node outside of S. Then the connectivity between node i and set S is $C(i,S) = \sum_{j \in S} C(i,j)$.

Connectivity between two sets. S_1 and S_2 are two sets in the graph, the connectivity between set S_1 and S_2 is $C(S_1,S_2) = \sum_{i \in S_1} C(i,S_2)$.

Connectivity of a set. The connectivity of set S is defined as the expected connectivity of any node i in the set S to the set S_{-i}, S_{-i} is the set S excluding node i. Then, $C(S) = \frac{\sum_{i \in S} C(i,S_{-i})}{n}$, where n is the number of nodes in S.

The connectivity C denotes the connectivity level or closeness between two objects as in Fig. 17.4. For example, consider the closeness between a node i and a set S. Node i has one or more edges connected to set S, with weight $p_1, p_2, \cdots p_k$ respectively. As each edge weight p_i denotes the probability that a message is successfully delivered from i to S through that particular edge i, the probability that a message is successfully delivered from i to S can be computed by $1-(1-p_1)(1-p_2)\cdots(1-p_k)$. After ignoring the product items, it can be simplified as $p_1 + p_2 + \cdots + p_k$, which is the connectivity $C(i,S)$ between i and S.

Consider the connectivity of a set S. According to the definition of $C(S)$, each edge weight in S would be counted twice. Therefore, the connectivity of S can also be presented as $C(S) = \frac{\sum_{i \in S, j \in S} 2w(i,j)}{n}$. Without losing generality, we can rewrite it as $C(S) = \frac{\sum_{i \in S, j \in S} w(i,j)}{n}$, which is exactly the same as our fist constraint. Therefore, to satisfy the first constraint of clustered partitioning, we just need to maximize the connectivity C for each partition. Based on the definition of connectivity, we propose a heuristic algorithm to separate a graph into no more than k clustered partitions. k is a pre-defined threshold for the number of partitions.

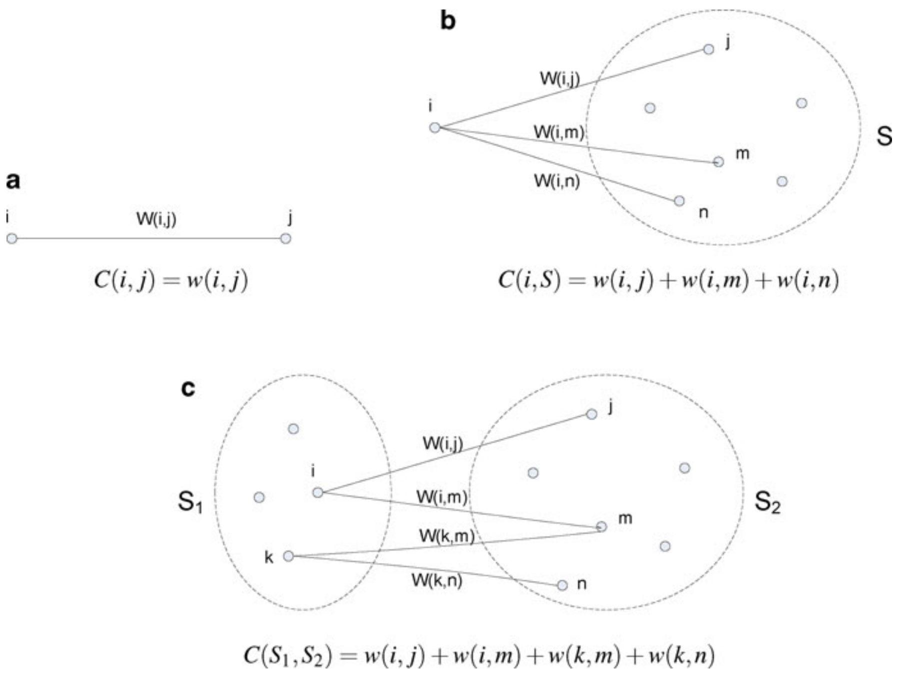


Fig. 17.4 Examples of how to measure the connectivity (**a**): between two nodes; (**b**): between a node and set; (**c**): between two sets

17.5.5 Clustered Graph Partitioning Algorithm

The basic idea behind clustered graph partitioning algorithm is to enlarge each partition from individual nodes based on the metric of connectivity; i.e., a new node which has the largest connectivity with the current partition is chosen and added to the partition. This process stops until any node's joining could not increase the connectivity for the partition. Then another partition expanding process is started from a remaining node. When there is no more partition growing, the graph has been partitioned to clusters, which is called a round. If the number of partitions is still larger than k, a new round is started, where each partition is contracted to a node and the partition expanding process is performed on the updated graph. The detailed algorithm includes following three recursive stages:

- *Expanding Stage*

  – Sort all edges in graph G by their weights w. Pick the edge with the largest weight and put its two end nodes into one partition P.

- Partition P grows as follows: for all neighboring nodes of this partition, choose node i which has the largest connectivity with partition P and add node i to form a new partition P'. Update $C(P')$. Repeat the above step on the new partition P' until there is no neighboring node that can achieve $C(P'') \geq C(P')$.
- Pick the edge with the largest weight from the rest of the edges and perform the above expanding process. The expanding stage stops when every node has been added to a partition.

• *Contracting*

- Based on the resulting partitions from the expanding stage, contract G to a condensed graph G' such that each partition P_i in G becomes a node i in G' and all the interconnection edges between two partitions P_i and P_j become an edge e(i,j) between the two corresponding nodes i and j in G'. $w(i,j) = C(P_i,P_j)$.
- Recursively apply the Expanding stage and the Contracting stage on graph G'. It stops when the number of partitions falls below the specified value k.

• *Restoring*

- Restore the original graph G by replacing each condensed node in each partition with its original nodes in the corresponding partition created in the contracting stage. Then, graph G is cut into less than k partitions.

Figure 17.5 illustrates how this algorithm works on a clustered graph. The distance between any two nodes denotes the closeness relationship between these two nodes. Thus, two nodes that are closer to each other in the plane would have higher connectivity and should be partitioned together.

The time complexity of this algorithm can be easily analyzed. At the beginning, there are n nodes in the graph which can be viewed as n individual partitions. In the end, the number of partitions is lower than k. As each partition expanding adds at least one node or one subset to a partition, there are at most $n-k$ times of partition expanding. For each partition expanding, a node with the largest connectivity to the partition is searched. This takes time $O(n_p * C)$, where n_p is the number of nodes in the current partition and C is the average degree of each node. In the worst case, the partition is as large as the entire graph and n_p becomes n. Therefore the total time for the algorithm is $O(n^2)$.

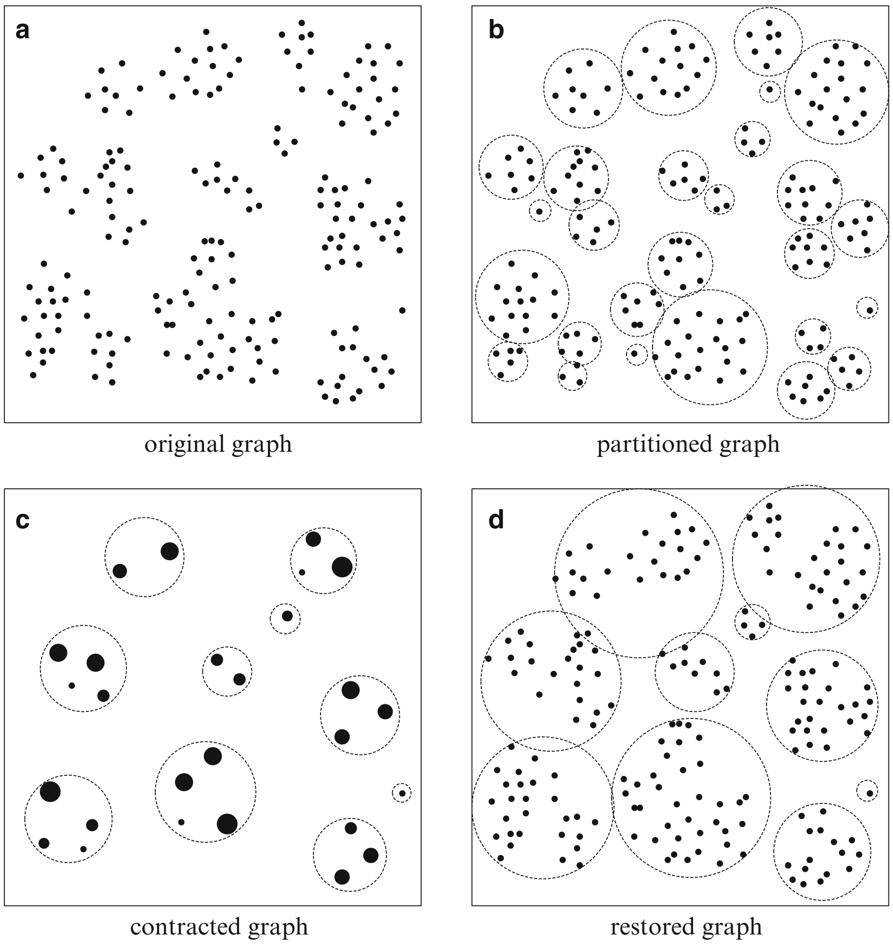


Fig. 17.5 An example of the clustered partitioning algorithm. (**a**) to (**b**) shows the partition expanding process. (**b**) to (**c**) shows contracted and partitioned graph. (**d**) restores to the original graph with 10 partitions

17.5.6 Worm Containment and Patching

In this part, we propose a systematic method to contain worms within different partitions. There are four steps to achieve it:

1. Build an undirected weighted graph G representing the mobiles' social relationship in the cellular network from a real trace.
2. Apply either balanced partitioning or clustered partitioning algorithm to graph G to obtain a partitioning and the corresponding cut edges.
3. Use the Minimum Vertex Separator Algorithm shown in Algorithm 1 to compute a minimum set of vertex separators from the set of cut edges.

Algorithm 1 Minimum Vertex Separator Algorithm

```
Input: $E_C$: the set of cut edges;
1: $V_S \leftarrow \phi$
2: while $E_C \neq \phi$ do
3:     Select $v \in V'$ which is shared by the most number of cut edges in $E_C$
4:     Add $v$ to $V_S$
5:     Remove from $E_C$ any cut edge whose end point is $v$
6: end while
Output: $V_S$: the set of vertex separators;
```

4. Send the security patches to separator nodes to block the worm propagation between partitions. These separator nodes could be responsible for forwarding the patches to other nodes in the same partition.

To obtain a set of separator nodes from the set of cut edges has been shown to be `NP-Complete` [12]. We propose Algorithm 1 to approximately solve this problem by the Greedy paradigm, in which the next vertex selected for the Minimum Separator Set is the vertex that covers the most uncovered elements.

To better illustrate how does our system work, a flow chart of the entire social network based worm containment system is shown in Fig. 17.6. The left part includes worm detection and patch dissemination while the right part involves trace analysis and separator set generation. All these components correlate with each other to work as a worm containment system.

17.6 Performance Evaluations

In this section, we evaluate and compare three different patching strategies, random patching, balanced patching and clustered patching based on the worm infection rate and the number of separator (patched) nodes.

17.6.1 Simulation Setup

Our experiments are based on the social relationship graph generated from a real network traffic trace from one of the largest cellular networks in the US. Although this trace data does not include all the mobile users nation-wide, it preserves similar characteristics as the national cellular networks. It is evaluated to have enough duration and scalability for our in-depth analysis of social interaction. Compared to related works that are based on cellular network traces, our trace analysis includes not only MMS and SMS messaging service, but also other popular services such as SIP based voice services as all of these services and interactions are equally likely to be exploited by worms.

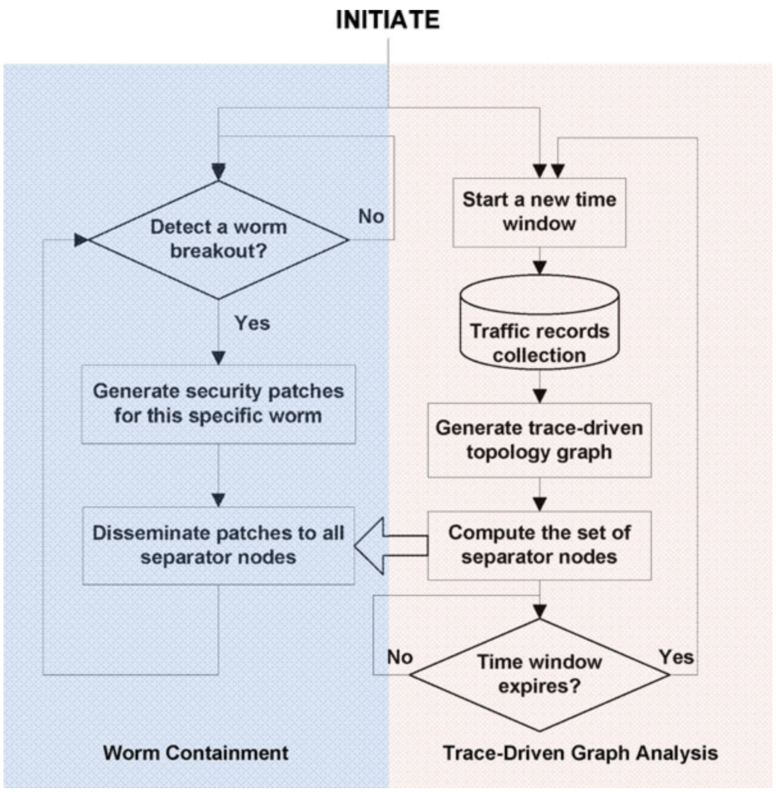

INITIATE
Detect a worm breakout?
No
Yes
Generate security patches for this specific worm
Disseminate patches to all separator nodes
Start a new time window
Traffic records collection
Generate trace-driven topology graph
Compute the set of separator nodes
Time window expires?
No
Yes
Worm Containment
Trace-Driven Graph Analysis

Fig. 17.6 The flow chart of trace-driven worm containment system

As far as we know, there does not exist any realistic model for worm propagation using SMS/MMS services in cellular networks. Although the work by Fleizach et al [11] models the mobile worm propagation, it is only based on US census data and estimated address book degree distribution. We construct a MMS worm propagation model as follows. We assume worms are able to exploit the social relationship information for propagating. We model the probability that a mobile will activate a worm received from another mobile as directly proportional to the connectivity level between them and model the time taken for the worm code to propagate from one mobile to another as inversely proportional to the connectivity level. This time includes the latency for worm transmission as well as the delay between the time of receiving the worm and the time of activating it. Once the worm has infected a new mobile, it starts to propagate to its neighbors after t time units.

We use a parameter of `Patching Threshold` α to control when the patching procedure starts. It is measured as the percentage of infected users in the network. This parameter represents the time delay since the worm starts propagating till it is detected by the network and a patch is generated. Once the percentage of infected users reaches this threshold α, the network would start to distribute patches to the chosen separator nodes.

Another parameter β is defined as the percentage of `Worm Sources` in the network. Worm Sources are a number of nodes which are randomly chosen from the network to initiate the infection process at the very beginning. This would provide the most pessimistic scenario as under a uniform distribution worm sources are more likely to be distributed across different clusters. One difference between α and β is that the infected users (determined by α) are chosen to be within the same cluster, while worm sources (determined by β) are uniformly distributed in the network. Each run of the simulation lasts for 2,000 time units.

17.6.2 Effects of the Patching Threshold

We assume that some kind of systematic detection system is deployed across the network to observe the abnormal traffic and detect any worm outbreak. The time when to start the patching procedure depends on the strength of the detection system. Obviously, early detection can achieve better effects on worm containment but consumes more resources on monitoring and computation, while later detection, though less resource intensive in terms of monitoring, could significantly delay worm containment. We use the parameter of patching threshold, α to simulate the time delay for worm detection and patch generation. Once the infection rate reaches this predefined threshold α, the network would start to distribute patches. Figure 17.7 compares the performance of three patching schemes: random patching, balanced patching, clustered patching under various α. We choose a number of ($\beta = 0.02\%$) nodes in the network by uniform distribution as the seed set of worm sources to initiate the infection process at the very beginning. As expected, the longer we wait to begin patching (higher patching threshold), the more number of nodes need to be patched for balanced or clustered patching to achieve the same infection rate. Interestingly, for random patching, the infection rate does not change irrespective of when to start the patching. Moreover, balanced patching has similar infection rate as random patching when patched nodes are under 2% in Fig. 17.7b and 2.6% in Fig. 17.7c due to the lack of enough separator nodes for effective partitioning.

As shown in Fig. 17.7, clustered patching requires much less patched nodes than balanced patching to achieve a certain infection rate in most cases. This is because clustered partitioning always cuts the graph from the least connected part, which results in less separator nodes, whereas balanced partitioning sometimes has to separate a strongly connected cluster apart and thus results in more separator nodes. Even in Fig. 17.7a, to achieve a low infection rate such as lower than 0.2, clustered patching requires much less patched nodes than balanced patching.

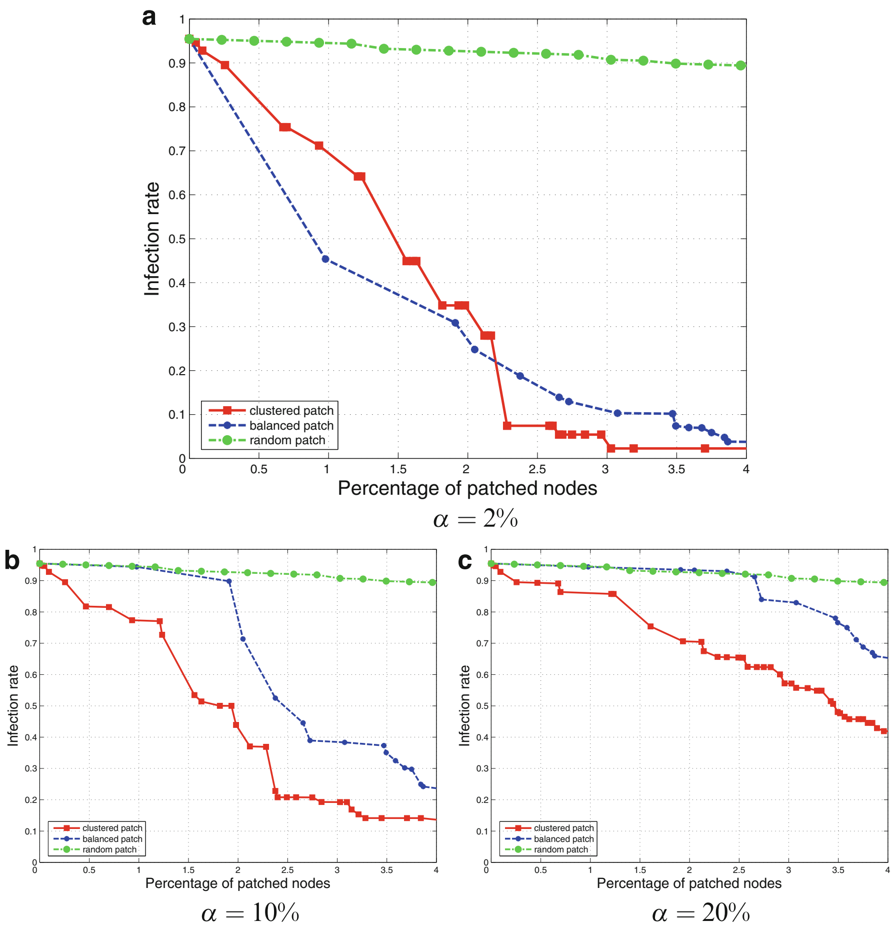


Fig. 17.7 Effect of patching threshold α ($\beta = 0.02\%$)

17.6.3 *Effect of β*

Worm Sources (β) are uniformly chosen from the network to initiate the infection process at the very beginning. This would provide the most pessimistic scenario as under a uniform distribution worm sources are more likely to be distributed across different clusters. In contrast, other distributions are more likely to decrease the number of clusters which have worm sources in them and thus would achieve a better performance for our containment strategy. A large set of worm sources helps to greatly speed up the worm propagation which makes the containment process more difficult. Figure 17.8 compares the performance of three patching schemes when β is equal to 0.02%, 0.1%, 0.2%, respectively. Note that both balanced patching and clustered patching perform better than random patching on both

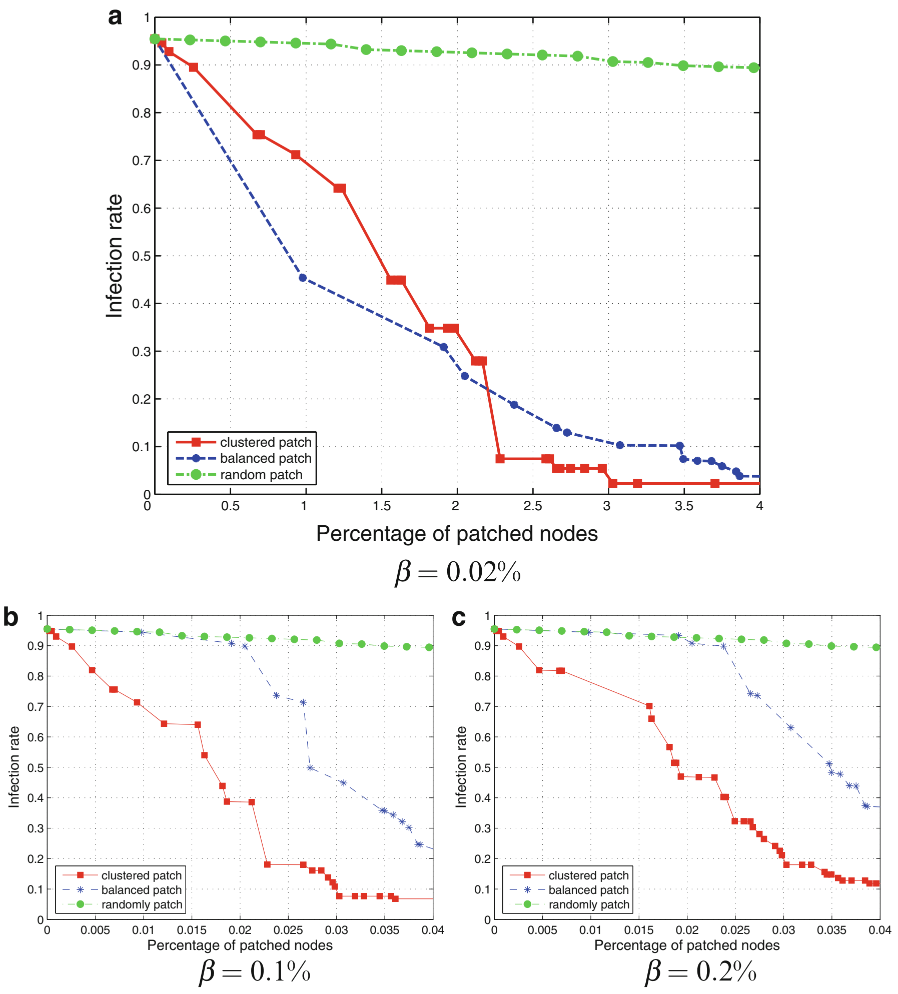
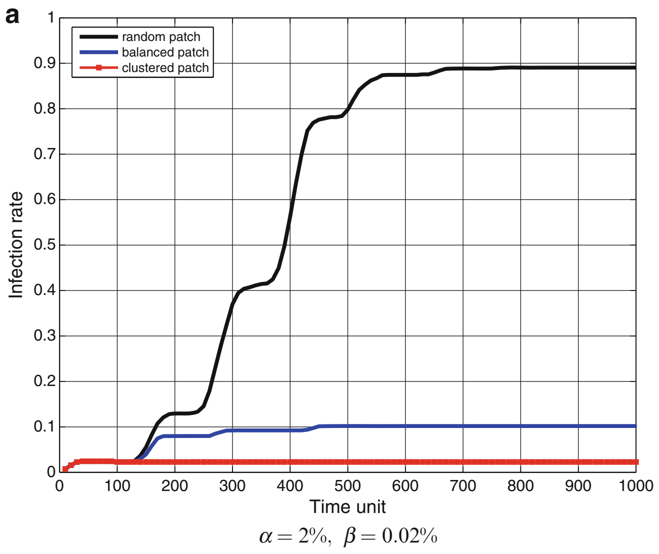


Fig. 17.8 Effect of the percentage of worm sources β ($\alpha = 2\%$)

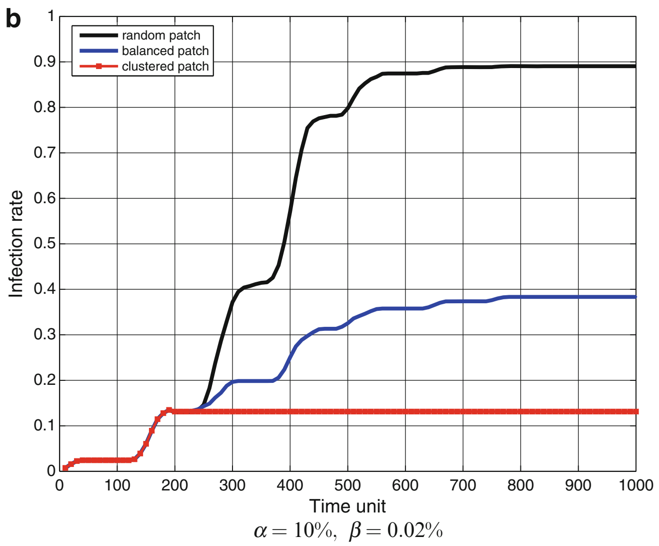
infection rate and number of patched nodes. Similar to Fig. 17.7, balanced patching has similar performance as random patching when patched nodes are under 2% in Fig. 17.8b and 2.4% in Fig. 17.8c.

17.6.4 Infection Rate Versus Time

Figure 17.9 shows how infection rate changes over time under different patching strategies. Clustered patching achieves the best performance as it limits the infection



$\alpha = 2\%,\ \beta = 0.02\%$



$\alpha = 10\%,\ \beta = 0.02\%$

Fig. 17.9 Infection rate vs. time (percentage of patched nodes = 4%)

rate within a certain bound much faster than the other two. Also, the infection rate can be bounded to a much lower value if the patching threshold is lower, i.e. patching is started earlier. We can observe from the figure if the operator were to begin patching the network after 2% of mobile devices had already been infected, then clustered partitioning bounds the infection rate to 0.025 within

30 time units. Balanced partitioning is only able to bound the infection rate to 0.1 with a longer 450 time units. However, both schemes perform significantly better than random patching, which leads to 90% of nodes getting infected after 900 time units.

17.6.5 Effect of Dynamic Graph Topology

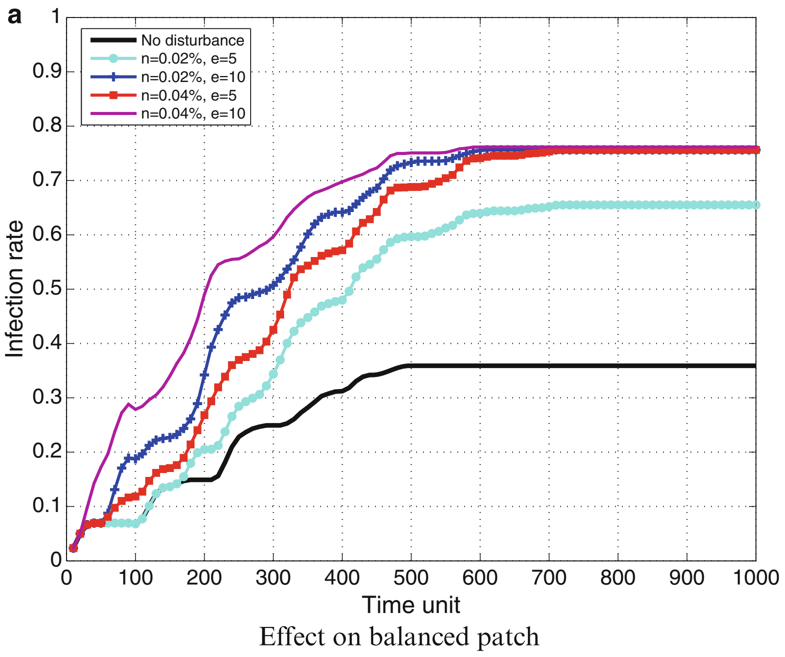
The trace we collected for social interaction analysis may not always be up to date unless it is frequently updated. To avoid the updating overhead, there will be a gap between the time when the social relationship graph is generated and the time of worm breakout. For example, a few new users may register and join the network, and start to build their social relationships. This may result in inaccuracies in our patching schemes. Figure 17.10 shows the effect of dynamic topology changes on the two patching schemes under various disturbance levels, where n denotes the percentage of new users joining the network and e denotes the number of edges for each new user connecting to other users. Notice that we cannot differentiate the curve of no disturbance and curve of $n = 0.02\%, e = 5$ for clustered patching because they behave the same. From the figure we can see that clustered patching is always behaving robuster to network disturbance than balanced patching. This is because new users usually join a certain cluster and only communicate with users in this cluster. Therefore, clustered patching can tolerate this disturbance more effectively.

17.7 Discussions

Our worm containment strategy assumes the presence of a detection system to detect a newly propagating worm. There are several works for detecting mobile worms at the network-level such as SmartSiren [3] and at the host-level using behavior anomaly detection [1] or energy anomaly detection [20]. These mobile worm detection systems can detect a worm within a reasonable latency and hence could serve as the initial trigger for our worm containment via patch distribution. Moreover, as a game between the worm designer and the patch system designer, patch designer should choose carefully between balanced patching and clustered patching to cope with worm designer's various tactics. Next, we discuss several other related issues.

17.7.1 Patch Generation and Distribution

Service providers usually take a multi-pronged strategy towards containing a zero-day worm – they start rate-limiting or filtering outbound traffic from hosts that



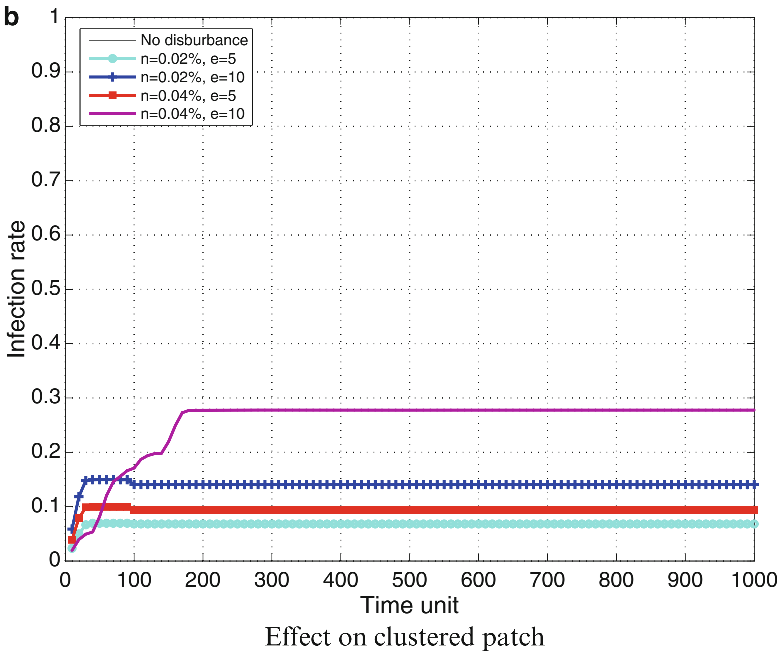


Fig. 17.10 Effect of dynamic topology under various disturbance levels (percentage of patched nodes $= 4\%, \alpha = 2\%$)

are infected and also start extracting the signature of the worm so that uninfected hosts can be protected. Developing a patch typically takes a substantial amount of time and manual efforts. The time scale required to generate security patches or signatures is up to 2 h [6]. Regardless of whether the service provider is able to develop the patch within hours of worm outbreak, our proposed mechanism can also be used for rate-limiting in the following way. Once the service provider has identified the set of key nodes via our partitioning algorithms, he can generate *stricter* filtering or rate-limiting rules for outbound traffic from these nodes so that the damage due to the worm can be contained more effectively.

It is up to the network operator to decide which approach to use to distribute the patches: push, pull or traffic controlling. In the case of pull, some mobile users may refuse to install the patch since they may not trust the source of the patch. An efficient way is needed for the network operator to have the patch messages authenticated. Fortunately, many mobile operators have some "Wake Up" mechanism to directly distribute software or patches to devices without any intervention from the users to make patches activated.

17.7.2 *Efficiently Utilize the Wireless Bandwidth*

An effective patch distribution strategy needs to make sure that the patches do not compete for the scarce bandwidth resources. The uniform patching in this regards becomes *white worm* [32], which usually deploys an arbitrary payload and cannot be practically bounded. Our patch distribution mechanisms take a hierarchical approach and instead of flooding out the patch to *all* nodes, we determine an optimal set of nodes to which the patch must be sent to obtain a bounded infection rate. These patches involve nothing about broadcasting and only bring limited traffic into the network, e.g. only 0.25% of devices need to be patched via clustered patching to bound the infection rate at 0.90 (Fig. 17.8b) compared to random patching for which a much larger 4% of devices would have to be patched to achieve an equivalent infection rate. Even considering the case in which patches are propagated from the separators to other mobiles within a partition, it would not cause any problem as the propagation is only bounded in those infected partitions which often have a limited number.

Cellular network bandwidth usually places constraints on worm propagation and patch dissemination. However, we can skip this influence in our simulation since once our patching strategy is deployed, the worms should have been contained and stopped at the very early stage before saturating the network capacity. For patch distribution, as only a limited number of mobiles are patched while no broadcasting is introduced, the limited patching traffic is far away from saturating the cellular network bandwidth.

17.7.3 Social Factors Affecting the MMS Worm Propagation

MMS worms can send a copy of itself to all mobile phones whose numbers are found in the infected phones' address books, or numbers found from message boxes, and cause tremendous damage to mobile phone users and even the entire network. Other than the social relationship considered in our worm propagation model, the following social factors may also effect the MMS worm propagation:

- *User's Confirmation*: While the MMS worms can autonomously copy themselves from one device to another, users have to actively install them for the worm to be propagated. Statistically, 25% of MMS messages have never been opened by the recipients. This may be because they do not trust the senders, or they are not in a good time for checking messages.
- *Message Waiting Time*: Previous work assumes fixed time interval between the time when a worm message is sent out and the time for the victim to be infected. For example, [31] chooses 2 min as the time required for a MMS virus to be received by another handset and to install itself. However, there is always a message waiting time before the MMS message is actually retrieved. This message waiting time could vary from less than 1 minute to more than 1 day, depending on who's the sender and the receiving time during the day.
- *Time Zone*: Time zones play an important and unexplored role in malware epidemics. Dagon et al. [4] studied diurnal properties in botnet activity to understand how time and location affect computer malware spread dynamics. Clearly, computers that are turned off at night are not infectious. It is similar to the case of MMS worms in that mobile phone users are not infectious at night when they are in sleep and cannot activate the malicious message. However, unlike computer worms where malicious codes directly reach victim computers, malicious MMS messages are saved in the MMS server for a time period before users retrieve them onto their mobile phones.

Therefore, future research on MMS worm propagation modeling should consider these social factors, which may speed up or slow down the worm propagation from spatial or temporal perspectives.

17.8 Conclusion

This chapter proposed a methodology for effectively limiting the spread of MMS and SMS based worms via a graph partitioning approach. In our solution, mobile devices are divided into multiple partitions based on the social relationships among them. Two patching schemes, namely balanced and clustered patching are designed and their performance is evaluated using simulations based on data collected from real cellular networks. Through extensive evaluations, we demonstrate that our partitioning strategy can effectively contain worms.

This is one of the first work to use a real network traffic trace to study the social interactions and relations between any two mobiles and the vulnerability information exploited by worm for spreading. Further research in this area includes dealing with hybrid worms which can make use of both cellular network interface and Bluetooth interface to propagate, and looking into worms and users roaming between cellular networks operated by different service providers.

References

1. A. Bose, X. Hu, K.G. Shin, and T. Park. Behavioral detection of malware on mobile handsets. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 225–238. ACM, 2008.
2. A. Bose and K.G. Shin. Proactive security for mobile messaging networks. In *Proceedings of the 5th ACM workshop on Wireless security*, page 104. ACM, 2006.
3. J. Cheng, S.H.Y. Wong, H. Yang, and S. Lu. SmartSiren: virus detection and alert for smartphones. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, page 271. ACM, 2007.
4. D. David, C. Zou, and W. Lee. Model Botnet Propagation Using Time Zones. In *Proceeding of the Network and Distributed System Security (NDSS) Symposium 2006*.
5. W. Enck, P. Traynor, P. McDaniel, and T. La Porta. Exploiting open functionality in SMS-capable cellular networks. In *Proceedings of the 12th ACM conference on Computer and communications security*, page 404. ACM, 2005.
6. F-SECURE. Close the zero-hour gap: Protection from emerging virus threats, http://www.f-secure.com/f-secure/marketing/white_papers.
7. F-SECURE. F-secure deepguard – a proactive response to the evolving threat scenario, http://www.f-secure.com/f-secure/marketing/white_papers.
8. F-SECURE. F-secure malware information pages: Sms-worm:symbos/feak, http://www.f-secure.com/v-descs/sms-worm_symbos_feak.shtml.
9. F-SECURE. F-secure virus information pages: Cabir, http://www.f-secure.com/v-descs/cabir.shtml.
10. F-SECURE. F-secure virus information pages: Commwarrior, http://www.f-secure.com/v-descs/commwarrior.shtml.
11. C. Fleizach, M. Liljenstam, P. Johansson, G.M. Voelker, and A. Mehes. Can you infect me now?: malware propagation in mobile phone networks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*, page 68. ACM, 2007.
12. M.R. Garey and D.S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences*. WH Freeman and Company, San Francisco, Calif, 1979.
13. M. Ghaderi and S. Keshav. Multimedia messaging service: system description and performance analysis. In *First International Conference on Wireless Internet, 2005. Proceedings*, pages 198–205, 2005.
14. J.C. Haartsen, E.R.S. BV, and N. Emmen. The Bluetooth radio system. *IEEE Personal Communications*, 7(1):28–36, 2000.
15. B. Hendrickson and T.G. Kolda. Graph partitioning models for parallel computing* 1. *Parallel Computing*, 26(12):1519–1534, 2000.
16. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359, 1999.
17. G. Karypis, K. Schloegel, and V. Kumar. ParMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library Version 3.1. *University of Minnesota, Minneapolis*, 2003.

18. B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.
19. M. Khouzani, S. Sarkar, and E. Altman. Maximum Damage Malware Attack in Mobile Wireless Networks. In *IEEE Societies INFOCOM 2010. Twenty-Nine Annual Joint Conference of the IEEE Computer and Communications*.
20. H. Kim, J. Smith, and K.G. Shin. Detecting energy-greedy anomalies and mobile malware variants. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 239–252. ACM, 2008.
21. F. Li, Y. Yang, and J. Wu. CPMC: An Efficient Proximity Malware Coping Scheme in Smartphone-based Mobile Networks. In *IEEE Societies INFOCOM 2010. Twenty-Nine Annual Joint Conference of the IEEE Computer and Communications*.
22. X. Meng, P. Zerfos, V. Samanta, S.H.Y. Wong, and S. Lu. Analysis of the reliability of a nationwide short message service. In *IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pages 1811–1819, 2007.
23. J.W. Mickens and B.D. Noble. Modeling epidemic spreading in mobile environments. In *Proceedings of the 4th ACM workshop on Wireless security*, page 86. ACM, 2005.
24. A.G. Miklas, K.K. Gollu, K.K.W. Chan, S. Saroiu, K.P. Gummadi, and E. De Lara. Exploiting social interactions in mobile systems. In *Proceedings of the 9th international conference on Ubiquitous computing*, pages 409–428. Springer-Verlag, 2007.
25. D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The spread of the sapphire/slammer worm, http://www.caida.org/publications/papers/2003/sapphire/ sapphire.html, 2003.
26. D. Moore, C. Shannon, G.M. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In *IEEE Societies INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*, pages 1901–1910.
27. K. Schloegel, G. Karypis, and V. Kumar. Graph partitioning for high-performance scientific simulations, Sourcebook of parallel computing, 2003.
28. E. Van Ruitenbeek, T. Courtney, W.H. Sanders, and F. Stevens. Quantifying the effectiveness of mobile phone virus response mechanisms. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007. DSN'07*, pages 790–800, 2007.
29. M. VojnoviĆ and A. Ganesh. On the effectiveness of automatic patching. In *Proceedings of the 2005 ACM workshop on Rapid malcode*, page 50. ACM, 2005.
30. C. Walshaw and M. Cross. Parallel optimisation algorithms for multilevel mesh partitioning. *Parallel Computing*, 26(12):1635–1660, 2000.
31. P. Wang, M.C. Gonzalez, C.A. Hidalgo, and A.L. Barabasi. Understanding the spreading patterns of mobile phone viruses. *Science*, 324(5930):1071, 2009.
32. N. Weaver and D. Ellis. White worms don't work. *Login*, 31:33–38, 2006.
33. C. Wong, S. Bielski, A. Studer, and C. Wang. Empirical analysis of rate limiting mechanisms. In *Recent Advances in Intrusion Detection*, pages 22–42. Springer, 2006.
34. Y. Yang, S. Zhu, and G. Cao. Improving sensor network immunity under worm attacks: a software diversity approach. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 149–158. ACM, 2008.
35. B. Zhao, C. Chi, W. Gao, S. Zhu, and G. Cao. A chain reaction DoS attack on 3G networks: analysis and defenses. In *IEEE Societies INFOCOM 2009. Twenty-Eight Annual Joint Conference of the IEEE Computer and Communications*.
36. Z. Zhu and G. Cao. Worms in Cellular Networks. Book Chapter in *Encyclopedia of cryptography and security (2nd Ed.)*. Springer Verlag, 2010.
37. Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci. A Social Network Based Patching Scheme for Worm Containment in Cellular Networks. In *IEEE Societies INFOCOM 2009. Twenty-Eight Annual Joint Conference of the IEEE Computer and Communications*, pages 1476–1484.

38. C.C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and early warning for internet worms. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 190–199. ACM, 2003.
39. G. Zyba, G.M. Voelker, M. Liljenstam, A. Méhes, and P. Johansson. Defending mobile phones from proximity malware. In *IEEE Societies INFOCOM 2009. Twenty-Eight Annual Joint Conference of the IEEE Computer and Communications*, 2009.

Index

A
Abilene network, 112, 134, 135
Accessibility, 4
Ad-hoc wireless network, 29, 37–38, 198
AI planning, 273, 275
Algebraic connectivity, 5, 23–26, 30, 101
All pairs shortest distance (APSD), 483–489
Animal flocking, 62
Anomaly detection, 165, 168, 523
arXiv e-print citation network, 307, 321, 324–325

B
Backbone network, 100
Backpressure-based learning, 427–453
Backtracking line search, 120, 121
Balanced graph partitioning, 510–512
Bandit-problem, 453
Barabási–Albert (BA), 28, 30, 31, 67, 69, 72, 73, 77, 78, 80–82, 86, 282
Behavioral characteristic, 165
Behavioral signature, 505
Betweenness, 6, 10, 11, 17, 18, 25–27, 63, 74–79, 83–88, 98, 101–110, 203–208, 243, 245
  centrality, 17–19, 25, 26, 45, 83, 98–104, 108, 109, 200, 202–206, 216, 280, 370, 407
Bilevel stochastic integer programming, 49
Bi-mesh structure, 180–185, 191, 192
Biochemical network, 3
Block structure, 166, 167, 172–173, 193
Blondel method, 321, 323–325
Bluetooth, 328, 350, 353, 365, 392, 393, 456, 472, 502, 505, 527
Branch-and-bound (B&B), 112, 155, 156, 221, 223, 226–232, 238
Branch & cut, 156
Bridging centrality (BC), 197–217
BUBBLE, 350–352, 363–365, 370–372, 376–380, 382, 383
BUBBLE Rap, 456, 457
Business integration, 301

C
Cache replacement, 458
Call network, 157, 158
Cambridge Haggle Trace, 471
Canonical order, 488
Carraghan–Pardalos algorithm, 153, 156
Cascade, 37–55, 86–88
  control, 63, 86–88
Cascading failure, 63, 84, 86–88
Causally-constrained queuing network, 63
Cellular network, 13, 331, 501–527
Cellular-social relationship graph, 507–509
Cellular tree hierarchy, 44
Centrality, 6, 10, 11, 23, 25, 70, 79, 80, 98, 101, 102, 197–217, 279, 287, 288, 291, 351, 352, 369–370, 372–375, 377, 380, 382, 398, 407–411, 457
Centrality-based forwarding, 407, 418
CFinder, 340–342, 345
Chaotic oscillator, 63
Classified traffic routing (CTR), 242, 252–260
Clique, 47, 50, 52–55, 144–149, 155–156, 158, 159, 318, 336, 337, 345, 419, 494–498
  relaxation models, 143–159
Cliquer, 154
Closeness centrality, 200

Cluster, 67, 158, 172, 174, 178, 179, 185, 200, 204, 206, 215, 217, 274, 283, 284, 302, 308, 314, 331–334, 344, 480, 503, 505, 514, 519, 520, 523
Clustered graph partitioning, 511–512, 514–516
Co-clustering, 166, 173–175, 179, 193
Cohesiveness, 144–146, 189
Cohesive subgroup, 144, 145, 147, 148, 158
Co-*k*-plex coloring problem, 156
CollabNet, 43
Collision avoidance, 220
Communication network, 13, 20, 41, 48, 76, 77, 97–141, 309, 327, 335
Communication pattern, 165, 166, 172, 173, 176, 193, 504
Community, 24, 70, 100, 166, 308, 351, 388, 456, 490
Community structure, 70, 186, 192, 193, 307–345, 365, 370–372, 379
Commute time, 76, 92, 131
Computational indistinguishability, 483
Conductance, 70–73, 82, 138, 139
Connectivity, 4–6, 10–13, 24–27, 29, 30, 39, 40, 42, 47, 50–52, 99, 101, 136, 148, 158, 170, 173, 180, 181, 185, 198–203, 210, 263, 264, 276, 286, 301, 337, 350–352, 355–358, 363, 381, 382, 391, 393, 394, 396, 398, 418, 493, 496–498, 513–515, 518
  attenuation, 42
  graph, 158, 203
Contact aware, 399, 401
Contact graph, 359, 366, 388, 390, 391, 393, 399–401, 403, 405, 422, 423
ContentPlace, 458, 459
Cooperative control, 62
Coordination phenomena, 62
COPRA, 340–343, 345
Correlated pool structure, 185
Coupled oscillator, 63
Coupling matrix, 65, 90
Coupon distribution, 433–436, 438–446, 448, 452
Coupon forwarding reward rate, 438, 441
Coupon rate selection, 445–446
Coupon routing at target rate, 439
CPMC, 505
Credential authority, 481, 498
Crime ring, 38
Cross-layer optimization, 220, 221, 235, 237, 238
Current flow, 62, 79–83
  betweenness, 63, 79–82

D
Damping factor, 389, 400, 403–406, 416, 417, 423
DC-model, 138
Degree centrality, 6, 101, 200–201, 214, 288
Degree distribution, 5, 12, 29, 31, 51, 55, 65, 66, 69, 72, 78, 91, 170, 172, 244, 279, 281, 340, 450, 492, 518
Degree-based forwarding, 407, 408, 418
Degree-degree correlation, 67, 89
Delay-based coupon forwarding, 447–448
Delay tolerant mobile ad hoc network (DT-MANETs), 456–458, 460, 477
Delay-tolerant network (DTN), 206, 213, 351, 380, 382, 383, 397, 457, 458
Delivery ratio, 328, 329, 350–356, 358, 372, 379, 381, 383
Demand spike, 41
Density function, 336, 338
Deterministic betweenness, 103
Diagonalization, 63, 64, 67, 68, 70, 80
Diameter, 8, 24–26, 29, 99, 146, 147, 154, 156, 158, 280, 281, 392, 415
Dijkstra's algorithm, 45, 280, 397, 483, 486
Directed graph, 130, 131, 141, 490
Distance, 8, 15, 16, 18, 21–23, 25, 28, 41, 42, 44, 45, 50, 53, 89, 98, 99, 101, 108–113, 120, 130, 138–140, 146, 147, 150, 154, 155, 223, 224, 229, 233, 237, 275, 279–282, 284, 339, 393, 398, 406, 419, 420, 481, 483–487, 493, 495, 515
Distance-attenuated reach, 41
DOCA, 309, 310, 335, 337, 339–343, 345
DROP algorithm, 155
DROP & CONSTELLATION algorithm, 155
DTN. *See* Delay-tolerant network (DTN)
Dual decomposition, 123
Duality gap, 119, 120, 228
Dunbar's number, 353
Dynamic social networks, 307–345

E
Edge connectivity, 4, 11, 50
Edge coverage, 176, 177, 181
Edwards–Wilkinson (EW) process, 64, 70, 71
Ego-centric centrality, 203–206, 208, 382, 457
Eigenratio, 63
Eigenvector centrality (EVC), 6, 199–202, 212, 213, 216
Email spam, 182, 183, 192, 480
Empirical data, 145, 360

ENRON email network, 309, 321–324
Equivalent fraction, 362, 363
Erdos–Renyi, 50, 52, 53
  network, 47, 81, 82, 281
Espionage, 38
Evolution algorithm, 495–498
Exponential decay distribution, 458

F
Facebook network, 182, 183, 189, 263, 308, 309, 321, 325–327, 330, 332, 388, 399, 482
FacetNet, 344
Fair load distribution, 415–417
Feasible scaling factor, 223, 225, 234, 235, 237
Financial market, 38
Finite buffer, 458, 464, 469–471, 474–476
Fitness, 42–55, 367, 369, 433
Flow, 20, 40, 50, 61–92, 98, 99, 101–103, 106, 109, 110, 121–135, 138, 140, 165, 166, 168–172, 179, 181, 182, 184, 186, 188, 190, 191, 200, 202, 204, 206, 210, 211, 221–224, 238, 264–268, 274–276, 296, 429–432, 434, 438, 483, 507, 517, 518
  assignment problem, 125, 131
  betweenness, 102
Flow-level traffic clustering, 165
Floyd-Warshall, 488
FTP network, 42–44

G
Game-theoretic, 47
Geodesic bottleneck, 21, 22
Giant connected components (GCCs), 87, 158, 166, 170–173, 179, 184
Gnutella, 29, 43, 171, 183
Gradient vector, 120
Graph
  clustering, 308
  decomposition, 166, 167, 173–179, 190, 193
GraphScope, 344
Graph theory, 4, 23, 24, 39, 99, 101, 102, 145, 159, 172, 200, 481, 483, 484, 509–517
Graphviz tool, 169
Greedy perturbation algorithm, 490–491
Greedy randomized adaptive search procedures (GRASP), 157, 158
Green's function, 71
Grid search, 50, 51
Guerrilla group, 38

H
Harmonic mean, 45
Heavy-hitters, 165
Height fluctuation, 67, 68
Hessian matrix, 120
Heterogeneous, 31, 63, 65, 69, 72–73, 82–83, 88, 89, 243, 273, 275, 352, 359, 381
Heuristic control, 273, 274
Hits algorithm, 389
Hitting time, 76, 92, 131
Homophily phenomenon, 456, 459
HOPE dataset, 392–394, 411, 415
Host-level traffic behavior profiling, 165
Hub, 29, 55, 72, 74, 79, 83, 84, 88, 243, 245, 252, 254, 255, 352, 369, 370, 372, 389
  avoidance, 77, 83, 84
Human contact network, 349–383
Human mobility, 366, 417, 423

I
InDegree algorithm, 389
Independent paths, 4, 8, 148
Infinite buffer, 391, 463–469, 471–475
Infocom06 dataset, 364–366, 371, 379, 380, 393, 407, 410
In-star, 179, 181–184, 190
Integer programming, 49, 149, 152, 273, 274, 278
Intelligent tag, 220
Interaction structure, 165, 166, 173, 176, 186
Inter-community links, 311, 313, 319
Interior-point method, 120
Internet search, 480
Intra-community links, 311, 313, 316, 319
Inverse geodesic distance, 45, 99
IS-IS, 134

J
Joint graph, 483–488
Joint optimization, 97–141, 235, 428

K
K-anonymity, 491–495, 497, 498
K-clique, 147, 149–150, 153–154, 344, 345, 365, 366, 372
K-clique (*cont.*)
  algorithm, 153–354, 365
  community detection, 365, 366, 368
  percolation technique, 344
K-club, 147, 149–151, 154–155
K-component, 8, 23

K-copy forwarding, 351, 362, 363
K-core, 147–149, 155–156
Kernighan-Lin (KL), 511
Kirchhoff's Current Law, 70, 105
Kolmogorov-Smirnov goodness-of-fit test, 177
K-plex, 148, 151–152, 155–157
K-robust, 99, 130, 136, 138–140
Kronecker delta function, 103, 123
Kruskal, 488
Kullback-Leibler (K-L) divergence method, 462–463
Kuramoto model, 63
K-vertex connected subgraph, 148, 149

L
LABEL, 328, 329, 372, 379, 382
Lagrangian, 117, 123, 128, 129
Langevin equation, 64
Laplacian, 18, 25, 62–65, 70, 79, 80, 89, 92, 101, 103, 108, 115, 117, 131
Large-scale, 49–50, 156, 157, 275, 278, 292, 301, 327, 344, 419, 423, 448, 481, 489, 502
Latent hypergraph, 167, 176, 179, 183–185, 190
LFR overlapping benchmark, 340
Linear logic theorem, 273
Link-based vulnerabilities, 17–19, 28, 30, 32
Link failure, 5, 26, 121, 129–130, 137, 140, 279
Link layer scheduling, 220, 222, 235, 238
Lipschitz-continuous, 48
Local community, 336–339
Local search, 229, 231–233, 314
Logarithmic barrier, 120
Low convergence rate, 178–179

M
11M, 42–45
Marketing, 301, 428
Master stability function, 63
Mathematical programming formulation, 145, 146, 149–153
Maximum clique problem, 149, 150, 153–156, 158
MaxProp, 456
Mean contagion size, 51
Mean-field (MF) approximation, 66–69, 71, 80, 90, 91
Menger's theorem, 148
Mesh-Mon, 198, 212
MIEN algorithm, 324, 328
Minimum separator set, 517
Minimum spanning tree, 483, 487–488
MIT reality mining project, 353, 364
MIT Reality Mining Trace, 471–474, 476
Mixed integer program (MIP), 149, 153, 157
MobiClique07, 391–392, 403, 404, 407, 408, 411–413, 416, 417
MobiClique08, 392
Mobile ad hoc networks (MANETs), 198, 206, 212, 309, 310, 327–329, 345, 351, 382, 419, 455–477
Mobile worm propagation, 502, 506, 518
Mobility, 209, 307, 327, 350, 359, 366, 369–373, 381, 390–393, 398, 399, 403, 407, 417, 418, 422, 423
Mobility-assisted data access, 456
Modularity, 280, 281, 309, 310, 312–314, 316, 317, 319–327, 344, 345, 367, 369
Module, 39, 158, 204, 275, 298, 299, 310, 314–316, 319, 321, 322, 335, 345
Multi-communities social forwarding, 419–422
Multi-criteria programming (MCP), 292–299
Multimedia messaging service (MMS), 502–507, 509, 517, 518, 526
Multi-objective optimization, 47, 48, 51, 54
Multiple-copy-multiple-hop (MCP), 328, 329, 372, 377, 379
Multi-scale node, 28, 30

N
Neoclassical planning, 273
Network
  congestion, 76–79
  criticality, 98, 108–113, 121, 122, 127–132, 135–137, 139, 140
  layer routing, 125, 220, 224, 235, 238
  planning problem, 120, 121, 124
  traffic, 77, 165, 173, 193, 504, 517, 527
Network-based data mining, 158, 159
Neural network, 303
Neurobiology, 61, 62
9/11, 42–45, 48
Node
  connectivity, 4, 6, 11, 30, 99, 492
  failure, 15, 16, 86, 87, 129, 139
  similarity, 11, 99, 521
Non-linear program (NLP), 221, 226, 228–230
Non-LOS LPI/D groundwave communication, 219
Normalized mutual information (NMI), 322–327, 340–342

NP-hard, 221, 226
  problem, 149, 153, 154, 156, 157, 226, 228, 236, 513

O
Ohm's law, 70
Online social networks, 182, 192, 308, 309, 321, 327, 330, 343, 345, 388, 399, 400, 427–453, 479–498
Open/close neighbors, 146
Optimal connectivity, 5, 11, 99
Optimal routing, 125, 140, 235–237
Optimization, 41, 62, 63, 65, 69, 71–73, 82, 89, 98, 145, 174, 215, 220, 242, 264, 428, 491
  decomposition, 429–434
  problem, 47, 49, 50, 65, 71, 89, 98, 101, 113–134, 136–141, 145, 146, 149, 155, 221, 223, 228, 235, 238, 291, 429, 438, 439
Optimized link state routing (OLSR), 198, 199, 206, 208, 209, 212–216
Orthogonal nonnegative matrix tri-factorization (tNMF), 166–168, 173–179, 185, 187, 189–191, 193
OSPF, 134, 136
Ostergard's algorithm, 156
Out-star, 180–185, 190
Overlapping communities, 335–343, 345, 365, 366
Overlapping ratio, 340, 341, 343

P
Packet-level malware signature, 165
PageRank algorithm, 388–390, 399
Parallel routing, 241–260
Pareto dominance, 272
Pareto frontier, 47, 51, 54, 55
Patch distribution procedure, 331, 332, 334
Patching threshold, 518–520, 522
Path delay distribution, 359–360
Peer-to-peer (P2P) application, 165–169, 171, 181, 183, 185, 191–193
PeopleRank, 387–424
Percolation phenomena, 40
Percolation transition, 44
Physical layer power control, 220, 221, 235, 238
Pocket switched network (PSN), 350–355, 358, 362, 365, 380–382
PodNet, 458, 459
Point-to-point resistance distance, 98, 99
Poisson distribution, 29, 281, 359, 360
Pool structure, 184–185, 190
PoP-level origin-destination (O-D) flow characterization, 165
Population dynamics, 61, 62
Power distribution system, 37, 39, 55
Power grid, 3, 32, 97–141
Power-law, 29, 77, 80, 359, 381, 450, 452, 453
  distribution, 29, 55, 244, 340, 458, 492
  tail, 70
Power signature, 505
Precision geolocation system, 219
Preferential-attachment., 67, 282, 450, 452
Prims, 488
Privacy-preserving graph, 483–489
Proportional flooding, 351, 358, 360–364

Q
QoS driven service selection, 270
QoS filtering algorithm, 275
Quality of service, 140, 247, 270–273, 275, 291, 292, 296, 298, 502
$\gamma-$Quasiclique, 148, 149, 152, 157
Quick community adaptation (QCA), 309, 321–331, 345

R
Randomly connected star structure, 50, 184, 191
Random walk, 62, 63, 66, 69, 73–79, 83, 102–108, 110, 131, 330, 381, 389, 390
  betweenness centrality, 98, 102, 108, 109
Rate feasibility problem (RFP), 225–228, 230–232
Rate limiting and quarantine, 505
Reachability, 101, 145–147, 352
Recommendation network, 136
Reformulation-linearization technique (RLT), 219–238
Resilience, 5, 37–55, 358, 382
  to path failures, 358–363
Resistor network, 62, 63, 70–75, 79, 92
Resource
  allocation, 123, 125, 274, 300, 429, 430
  description framework, 461
Robust capacity assignment, 136–138
Robustness, 3–32, 39, 61–92, 98–101, 108–110, 138, 139, 145, 146, 148, 198, 199, 204, 289, 310, 327, 335, 357
Rocketfuel topologies, 131, 136, 137

Route, 4, 97–141, 202, 211, 222, 235, 247, 250, 252, 309, 351, 363, 380, 430–433, 484
Routing, 62, 73, 74, 76, 77, 79, 83, 98, 158, 184, 198, 220, 242, 309, 351, 391, 432, 456

S
Scale-free network, 29, 31, 62, 72, 80, 83–87, 242–244, 281, 282, 450
SecondLife dataset, 392–393, 403, 408, 409, 414
Self-averaging, 64, 67
Semantic causal link matrices, 273, 274
Semantic matching algorithm, 268, 270, 273, 274
Semi-automatic method, 273
Semidefinite programming, 121–122
Semi-honest model, 483–489
Service network, 185, 263–302
Service oriented architecture protocol (SOAP), 275, 298, 299
Shortest path, 17, 42, 45, 83, 84, 86, 88, 102, 103, 122, 146, 154, 202, 242–245, 252, 280, 282, 286, 370, 407, 483, 484, 486–490
  betweenness, 63, 83–88, 102–104
  betweenness centrality, 83, 99, 102
Shortest path first (SPF), 134–136, 242–255, 257, 259
SimBet routing, 382, 456, 457
Single source shortest distance (SSSD), 483, 486, 489
Singular value decomposition (SVD), 174, 178–179, 193
SIR-Reach model, 52–54
Slater's condition, 124, 128
Small-world, 29, 63, 282, 353, 359, 457
Small-world network, 31, 32, 281, 282
SNA plugin (SNAP), 212–216
SOAP. *See* service oriented architecture protocol (SOAP)
Social-aware data diffusion, 455–477
Social-aware routing in MANETs, 327–329, 345
Social-based forwarding, 351, 373
Social-based routing, 370–380
SocialCast, 456
Social community network, 337
Social forwarding algorithm, 398, 418, 419, 421–423
Social network, 3, 6, 17, 29, 38, 39, 48, 52, 70, 143–149, 156, 159, 166, 182, 183, 192, 199–201, 203, 205, 208, 216, 217, 307–345, 365, 382, 388, 392, 393, 399, 400, 410, 419, 427–453, 456–458, 460, 479–498, 501–527
Social network analysis (SNA), 143–159, 199–207, 214, 216, 217, 481
Sociocentric betweenness centrality (C_{Soc}), 200, 202–204, 208, 209, 216
Socio-economic system, 38
SOLAR, 456
Spectral decomposition, 64, 70, 80
SSSD. *See* Single source shortest distance (SSSD)
StarClique, 491–498
Stationary distribution, 66
Steady-state width, 64, 69, 71
Storage site, 481, 482, 498
Storm worm traffic, 190–192
Structural cohesion, 149
Structural vulnerability, 3–32
Sub-optimality, 120
Susceptible-infected-recovered (S-I-R), 41, 51, 464
Sybil attack, 480
Synchronization, 25, 61–92

T
Tactical handheld & network LPI/D radio, 219
Targeted patching, 504, 510
Temporal stability, 188–190
Terrorist network, 38, 40, 42, 43, 143, 144
Throughput, 63, 77–79, 81–83, 86, 88, 100, 219–238, 242, 243, 245, 247, 430, 432, 434
Time-to-live (TTL), 328, 329, 372, 377, 379, 403, 414, 415
Topology-based propinquity, 344
TOTEM project, 112, 134
Trace-driven social relationship graph, 507–509
Traffic activity graphs (TAGs), 165–193
Traffic-aware betweenness, 98, 104, 110
Traffic-aware network criticality (TANC), 98, 99, 110, 112, 138
Traffic-aware node criticality (TANOC), 110
Traffic matrix, 100, 101, 104, 110, 112, 122, 123, 126, 131, 132, 134, 140
Traffic optimization, 241–260
Transition probability, 73–76, 103, 131
Transport, 3, 61, 62, 70–74, 242, 243, 358, 394, 395
Transportation network, 6, 32, 62

Trophic network, 13
Two-capacity model, 435–436

U
UCINET, 207
UCSD Wireless Topology Discovery, 353
UDDI, discovery and integration (UDDI), Universal description
Ultra-wideband (UWB), 219–238
Uncorrelated SF network, 66–68, 77, 79
Uniform patching, 510, 525
Universal description, discovery and integration (UDDI), 264, 268, 298

V
Vertex cover, 156, 331
Vertex separator, 335, 516, 517
Virtual network assignment, 131–134
Vitality, 101, 102
VoIP, 242, 247, 504, 506, 507
Voltage fluctuation, 41
Vulnerability, 4–28, 30–32, 79, 99, 100, 137, 502, 509, 527

W
WAIT, 328, 372
Web ranking, 388–390
Web service composition, 265, 266, 273–275, 279, 299, 301, 302
Web service ontology language (OWL), 276, 277
Web services description language (WSDL), 268, 276, 277, 285, 298, 299, 461
Weighted network analysis (WNA), 365–369
Wireless mesh network, 198–200, 203, 205–210, 216, 217
Worm containment, 309, 330–335, 343, 345, 501–527

Y
Yao's method, 488