Summer 2016

Modeling influence diffusion in networks for community detection, resilience analysis and viral marketing

Wenjun Wang
University of Iowa

MODELING INFLUENCE DIFFUSION IN NETWORKS FOR COMMUNITY DETECTION, RESILIENCE ANALYSIS AND VIRAL MARKETING

by

Wenjun Wang

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy degree
in Business Administration (Management Sciences)
in the Graduate College of
The University of Iowa

August 2016

Thesis Supervisor: Professor Nick Street

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Wenjun Wang

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Business Administration (Management Sciences) at the August 2016 graduation.

Thesis Committee: _____
Nick Street, Thesis Supervisor

_____
Jeffrey Ohlmann

_____
Gautam Pant

_____
Kang Zhao

_____
Xun Zhou

ACKNOWLEDGEMENTS

Completing my PhD degree is one of the most exciting and challenging tasks in my life. I could not have made it without the help of many people. First and foremost, I would like to express my sincere gratitude to my advisor, Professor Nick Street, for his continued support and great help. He has given me so much guidance, encouragement, and patience during the five-year journey of my PhD study. I deeply appreciate the opportunity and training he kindly offered to me.

I would like to give my sincere thanks to my thesis committee: Professors Jeffrey Ohlmann, Gautam Pant, Kang Zhao, and Xun Zhou for their time, great help and insightful comments.

I am grateful to Professor Renato de Matta for his co-authorship, recommendation, and encouragement. I would also like to thank Professors Ann Campbell, Samuel Burer, Barrett Thomas, Padmini Srinivasan, Hantao Zhang, Teodor Rus, Jia Lu, Ray Han, and Zhaochang Zheng for their help and support.

Besides, I want to thank Scott Hansen, Barb Murphy, Renea Jay, and my friends in the PhD program: Lian Duan, Michael Rechenthin, Huan Jin, Senay Yasar Saglam, Fahrettin Cakir, Xi Chen, Shu Zhang, Qiong Zhang, Stacy Voccia, Guanglin Xu, and Amin Vahedian Khezerlou, for their kind help whenever I asked.

I would like to give special thanks to my colleagues in the Department of Otolaryngology at the University of Iowa: Professors Camille Dunn-Johnson, Richard Tyler and Bruce Gantz, and Grant Worthington as well as many clinicians for their help, support and patience.

Last but not the least, I am deeply thankful for my beloved parents (Shibin Wang and Huizhen Hu), dear wife (Xiaofang Wang), lovely daughters (Jenna Wang and Cindy Wang), and younger brother (Wenyi Wang).

ABSTRACT

The past decades have seen a fast-growing and dynamic trend of network science and its applications. From the Internet to Facebook, from telecommunications to power grids, from protein interactions to paper citations, networks are everywhere and the network paradigm is pervasive. Network analysis and mining has become an important tool for scientific research and industrial applications to diverse domains. For example, finding communities within social networks enables us to identify groups of densely connected customers who may share similar interests and behaviors and thus generate more effective recommender systems; investigating the supply-network topological structure and growth model improves the resilience of supply networks against disruptions; and modeling influence diffusion in social networks provides insights into viral marketing strategies. However, none of these tasks is trivial. In fact, community detection, resilience analysis, and influence-diffusion modeling are all important challenges in complex networks. My PhD research contributes to these endeavors by exploring the implicit knowledge of connectivity and proximity encoded in the network graph topology.

Our research originated from an attempt to find communities in networks. After carefully examining real-life communities and the features and limitations of a set of widely-used centrality measures, we develop a simple but powerful *reachability-based* influence-diffusion model. Based upon this model, we propose a new *influence centrality* and a novel *shared-influence-neighbor* (SIN) similarity. The former differentiates the comprehensive influence significance more precisely, and the latter gives rise to a refined vertex-pair closeness metric. Then we develop an *influence-guided spherical K-means* (IGSK) algorithm for community detection. Further, we propose two novel *influence-guided label propagation* (IGLP) algorithms for finding hierarchical communities in complex networks. Experiments on both real-life networks and synthetic benchmarks demonstrate superior performance of our algorithms in both

undirected/directed and unweighted/weighted networks.

Another research topic we investigated is resilience analysis of supply networks. Supply networks play an important role in product distribution, and survivability is a critical concern in supply-network design and analysis. We exploit the resilience embedded in supply-network topology by exploring the *multiple-path reachability* of each demand node to other nodes, and propose a novel resilience metric. We also develop new supply-network growth mechanisms that reflect the heterogeneous roles of different types of units in supply networks. We incorporate them into two fundamental network topologies (random-graph topology and scale-free topology), and evaluate the resilience under random disruptions and targeted attacks using the new resilience metric. The experimental results verify the validity of our resilience metric and the effectiveness of our growth model. This research provides a generic framework and important insights into the construction of robust supply networks.

Finally, we investigate *activation-based* influence-diffusion modeling for viral marketing. One of the fundamental problems in viral marketing is to find a small set of initial adopters who can trigger the largest further adoptions through word-of-mouth-based influence propagation in the network. We propose a novel *multiple-path asynchronous threshold* (MAT) model, in which we quantitatively measure influence and keep track of its diffusion and aggregation during the diffusion process. Our MAT model captures both direct and indirect influence, influence attenuation along diffusion paths, temporal influence decay, and individual diffusion dynamics. Our work is an important step toward a more realistic diffusion model. Further, we develop two effective and efficient heuristics (IV-Greedy and IV-Community) to tackle the influence-maximization problem. Our experiments on four real-life networks demonstrate their excellent performance in terms of both influence spread and efficiency. Our work provides preliminary but significant insights and implications for diffusion research and marketing practice.

PUBLIC ABSTRACT

People live in various social networks, in which they build up relationships with their family members, friends, work colleagues, casual acquaintances, and so on. You may find that there are many different groups of people who have higher internal connectivity than external connectivity. Each of such densely connected groups is called a community. Community is one of the most significant structural properties of networks. *Community detection* is of great importance and a variety of applications. For example, if we are able to identify such communities of customers in a consumer network, we can investigate their similar interests and purchase behaviors and then generate more effective recommender systems to increase sales.

We like to share information and ideas with friends. In particular, new technologies and various social media rapidly penetrate into every aspect of our daily life, and provide us new channels and great convenience to exchange messages and express opinions. During these processes, we not only pass information on to each other, but also spread influence to each other. An interesting topic is to find a small set of most influential individuals as initial adopters of a new product, hoping that they can spread their influence through the network and trigger the largest further adoptions of the product. This is so-called influence-maximization problem of *viral marketing* in business.

In this dissertation, we develop a novel *reachability-based* diffusion model and three algorithms for community detection in networks. We then adapt this model to resilience analysis of supply networks under random disruptions and targeted attacks. We present a novel resilience metric and new supply-network growth mechanisms to build resiliency into the construction of supply networks. Finally, we propose a new, more realistic *activation-based* diffusion model and two effective algorithms for addressing the influence-maximization problem of viral marketing.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

The past decades have seen an unprecedented growth of network science and its applications. From the Internet to Facebook, from mobile phones to power grids, from protein interactions to paper citations, networks are everywhere and the network paradigm is pervasive. Network analysis and mining has become an important tool for scientific research and industrial applications to diverse domains.

A rich body of such studies focuses on the analysis of information and influence diffusion in complex networks. Influence is defined as the *capacity or power of persons or things to be a compelling force on or produce effects on the actions, behavior, opinions, etc., of others*[1]. Influence itself is usually imperceptible. In most cases, it is implicitly embedded in various types of information propagated from one to another. The information can be a message, an idea, an opinion, an advertisement, a behavior, a rumor, or a virus, and so on. For example, in word-of-mouth marketing, when Sam told his friends that he bought an iPhone and he liked it, he more or less influenced his friends on their purchase decision-making. When Jenna posts a tweet on Twitter and some of her followers retweet it, this is actually two steps of influence propagation originated from Jenna. In a citation network, a citing paper receives influence from the cited paper. In general cases of network analysis, influence and information are interchangeable. For convenience and simplicity, we use the term *influence diffusion* to indicate the diffusion of information and influence in networks.

Influence diffusion is a significant research topic in network analysis, which finds a wide range of applications to viral marketing, epidemic spread, outbreak detection, immunization strategies, etc. One well-known example is the commercial success of MSN Hotmail. It simply appended the text *"Join the world's largest e-mail service*

[1] dictionary.reference.com

with MSN Hotmail. http://www.hotmail.com" to the end of each email message. It quickly attracted 12 million users in only 18 months [65]. Another famous example is Christakis and Fowler's study on the spread of obesity in a large social network over 32 years [24]. They examined social contagion from various domains, such as obesity, happiness, wealth, and political beliefs [25,26], and proposed an important *3-degrees-of-influence* phenomenon that *"everything we do or say tends to ripple through our network, having an impact on our friends (one degree), our friends' friends (two degrees), and even our friends' friends' friends (three degrees). Our influence gradually dissipates and ceases to have a noticeable effect on people beyond the social frontier that lies at three degrees of separation"*. Online social networks have proved to be even more powerful in influence diffusion, such as Twitter during the 2008 US presidential elections [60] and Facebook during the 2010 Arab spring [58].

However, modeling influence diffusion is of great challenges. Although influence analysis has attracted a great deal of attention and extensive studies from many fields, the majority of the studies are still primarily rooted in a few classic influence-diffusion models, such as the *voter* model [28], the two *epidemic* models (SIR and SIS), the *Markov random field* (MRF) model [32,110], and the well-known *independent cascade* (IC) and *linear threshold* (LT) models proposed by Kempe et al. [69], etc. In spite of their popularity, those models have some singificant limitations on resembling real-world situations. For example, the SIR and SIS models completely ignore the underlying contact network, and assume every individual has the same probability to have direct contact with every other individual. In other words, they regard the contact network as a complete graph. The IC and LT models rely on the network connectivity, but they only consider the *direct influence* from the activated neighbors (*influencers*). In reality, a node in the network could be influenced and activated by its inactive neighbors who pass the influence from influencers on to the node of interest. In other words, the IC and LT models fail to capture the *indirect influence*

passed along by *messengers*. In addition, none of these models takes into account influence attenuation along diffusion paths or temporal influence decay.

There is another subtle issue. The influence spread in these models is measured by the total number of nodes that are *activated* in the end of the influence-diffusion process. The activation can be any type of status change, such as opinion change in the voter model, infection in the epidemic models, or new product adoption in the MRF and IC/LT models, etc. We call the influence in these models *activation-based* influence, and these models are thus called activation-based diffusion models. However, influence does not have to be confined to activation-based scenarios. For example, various centralities, such as *closeness* centrality [114] and *betweenness* centrality [42], have been widely used to identify the most important/influential vertices within a network graph, but none of them is activation-based. In general cases, a node's influence can be measured by the total amount of information it is able to spread through the network, i.e., in terms of the information's visibility or reachability regardless of the information's effect. We call it *reachability-based* influence. When one receives enough such information, he or she could be activated. In other words, reachability-based influence can be converted into activation-based influence. Reachability-based influence diffusion and how to incorporate it in the activation-based diffusion model are rarely studied in the literature.

Therefore, it is desirable to develop a more realistic and comprehensive influence-diffusion model that enables us to capture the dynamics of influence diffusion in more detailed manner. In this thesis, we use the concepts and techniques from the fields of network modeling, artificial intelligence, data mining, and social science to exploit the implicit knowledge of influence-based connectivity, centrality, and vertex-pair similarity encoded in network graph topology. We arrive at a set of powerful influence-diffusion models for community detection, resilience analysis, and viral marketing in complex networks.

This research originates from an attempt to find communities in social networks. Community is one of the most significant structural properties in networks. Without a quantitatively precise definition, a community is generally described as a group of nodes that have higher internal than external connectivity. Community detection is of great importance and a wide variety of applications. However, finding meaningful communities is a difficult task. Despite the active attention from many disciplines, there are still many open issues. Our work [127, 129] is motivated by the observation of the important roles of influence in real-world communities. Not only does influence differentiate individual roles in a community, but also acts as the force holding the individuals together to form and maintain the community. Influence is a natural fit for community detection. After carefully examining the features and limitations of a set of widely used centrality measures, we arrive at a novel reachability-based influence-diffusion model, which builds egocentric diffusion trees and generates an influence vector for each node. The influence vector captures not only the total influence but also the distribution of influence that each node spreads over its neighborhood. Based upon this model, we propose a new *influence centrality* in terms of the total amount of influence a node spreads out, and a novel *shared-influence-neighbor* (SIN) similarity that measures the closeness of a pair of nodes by their mutual influence and the common set of nodes they both influence. The former differentiates the nodes' comprehensive influence significance in a more detailed manner, and the latter gives rise to a refined vertex-pair closeness metric.

Moreover, we first develop an *influence-guided spherical K-means* (IGSK) algorithm for community detection, and then propose two novel *influence-guided label propagation* (IGLP) algorithms [128] for uncovering the community hierarchy in complex network systems, where small communities successively group together to form larger ones. One is called *IGLP-Weighted-Ensemble* (IGLP-WE), in which each node adopts the community label, carried by the majority of its neighbors and weighted

by the corresponding SIN similarity to the node of interest. This simple weighting scheme not only effectively resolves the significant stability issue in conventional label propagation algorithms, but also greatly improves the accuracy. The other is called *IGLP-Direct-Passing* (IGLP-DP), in which the community label is propagated directly from one node to its closest neighbor iteratively. This new label propagation method produces a deterministic partition and requires no convergent iterations. For IGLP-WE and IGLP-DP, the resultant partitioning is regarded as the initial configuration of the community structure. We define a new cluster-proximity measure and perform agglomerative hierarchical clustering to find communities at different scales. Further, we adapt our algorithms to identification of overlapping communities and individual roles in each community. Our approach naturally incorporates influence ranking, community detection and role identification into one unified framework, and is applicable to both undirected/directed and unweighted/weighted networks. We conduct extensive tests on on both real-life networks and synthetic benchmarks, and demonstrate superior performance of our algorithms among a set of state-of-the-art algorithms. IGLP-WE and IGLP-DP also manifest promising scalability for large-scale networks.

Another research topic we investigate is the resilience analysis of supply networks. Along with the globalization and rapid advancement of information technology, the traditional hierarchical supply chain has been quickly evolving into a variety of supply networks. Supply networks play an important role in product distribution, but they are often subject to various disruptions. Resiliency or survivability is a critical concern in supply-network design and analysis. As indicated in previous research [119, 138], supply network shares many characteristics that most real-life networks commonly have, and its graph topology has great impact on its resilience against disruptions. A challenging problem is how to measure the resilience from a topological perspective. Previous literature [95, 121, 143] mainly focus on some macro-level or generic metrics,

such as the size of the largest connected component, the average/maximum shortest path length, the supply availability rate, etc. While these metrics capture the intuitive concepts of network resilience from some specific angles, it is hard to piece them together to provide a comprehensive and reliable judgement on the network resilience. There are many other issues on these metrics, which we detail in Chapter 3.

The strategy we take is a novel bottom-up approach [126], in which we adapt our reachability-based influence-diffusion model for community detection to topological resilience analysis of supply networks. More precisely, we exploit the resilience embedded in the network topology by investigating in depth the multiple-path reachability of each demand node to other nodes, and quantify the network resilience by the aggregated resilience of all demand nodes in the supply network. Our approach gives rise to a new, more comprehensive network resilience metric. It differentiates the heterogeneous roles of different types of nodes, aggregates the resilience of all demand nodes, and incorporates the reachability to both supply nodes and demand nodes. Moreover, it considers not only the shortest paths but also other paths, and takes into account depth-associated penalty. Further, we adopt the multiagent modeling framework, and develop new supply-network growth mechanisms that enable us to leverage the network robustness against random disruptions and targeted attacks and improve the overall resilience. We incorporate them into two fundamental network topologies (random-graph network and scale-free network), and perform simulations to evaluate their resilience using our resilience metric. Experimental results verify the validity of our resilience metric and the effectiveness of our growth model. This research provides a generic framework and important insights into the construction and resilience analysis of supply networks.

Finally, we investigate *activation-based* influence-diffusion modeling for viral marketing, which is an important marketing technique that induces users in a social network to pass on a marketing message to other users so as to achieve a poten-

tially exponential growth in brand awareness or product sales. Marketing researchers study viral marketing using various descriptive/statistical models and field experiments [15, 56, 57, 62, 74]. It was first posed as an algorithmic *influence-maximization* problem by Domingos and Richardson [32, 110], in which they model the influence on each other as a Markov random field [72]. The objective of this problem is to find a seed set of most influential individuals in a social network such that activating them initially would trigger the largest influence spread in the network. In other words, it is to maximize the expected number of activated nodes at the end of the influence-diffusion process. In their seminal paper [69], Kempe et al. formulate the influence-maximization problem as a discrete optimization problem in three widely-studied stochastic diffusion models. As discussed above and detailed in Chapter 4, there are many important features missing in these models, such as indirect influence, influence attenuation, etc.

We propose a novel, more realistic influence-diffusion model, the *multiple-path asynchronous threshold* (MAT) model, in which we adapt our reachability-based influence diffusion to the activation-based scenario of viral marketing. Our MAT model takes into account both direct and indirect influence, influence attenuation along diffusion paths, influence decay with time, and individual diffusion dynamics. Further, we tackle the influence-maximization problem under the MAT model. We carry out theoretical analysis on its complexity and properties, and develop two effective and efficient approximation algorithms (IV-Greedy and IV-Community) along with a set of baseline heuristics. We run experiments on four real-life networks to evaluate our MAT model and algorithms against the baseline heuristics in terms of influence spread and time complexity. Our work provides preliminary but significant insights and implications for diffusion research and marketing practice.

The remainder of this thesis is organized as follows. In Chapter 2, we study community detection in complex networks. We start with an introduction and discussions

about related work, and describe in detail our reachability-based influence-diffusion model. We then propose three community-detection algorithms, and adapt them for overlapping-community identification and role detection. Last, we show our experimental results and performance comparison, followed by the conclusion. In Chapter 3, we present our work on topological resilience analysis of supply networks. After reviewing the background and related work, we elaborate our methodology as well as the new resilience metric and attachment strategies. We then perform simulation to evaluate our resilience metric and network growth model. In Chapter 4, we investigate activation-based influence-diffusion modeling for viral marketing. Similarly, we begin with an introduction and literature review, and then elaborate our MAT model. We then propose two approximation algorithms to tackle the influence-maximization problem under the MAT model, and run experiments on four real-life networks.

CHAPTER 2
COMMUNITY DETECTION

2.1 Introduction

Network analysis and mining has emerged in diverse disciplines as a means of analyzing complex systems. Most complex systems from various domains can be naturally mapped to different types of graph structures, such as the Internet and power grids in technological networks, protein-protein interactions and food webs in biological networks, and various information networks and social networks, etc. In the study of networks, a number of common characteristics have been discovered, including power-law degree distribution, small-world phenomenon, homophily and self-organization, among others. Another significant topological characteristic is *community structure*, which implies that nodes on the network are naturally grouped into different communities with dense connections internally and sparser connections between communities. From a karate club of tens of members [141] to large-scale online social networks with millions of participants, from functional protein modules in biology to arXiv citation networks of scientific papers, community structures occur in various network systems.

We illustrate in Figure 2.1 some real-life networks and schematic examples. Figure 2.1(a) is a simple network of three communities. Nodes in each community are more densely connected to each other than to the rest of the network. Figure 2.1(b) displays a well-known real-life social network of 62 bottlenose dolphins analyzed by Lusseau et al. [89]. Each node represents a dolphin, and the edge between a pair of nodes indicates they stay together more often than expected by chance. This network consists of two evident communities as indicated by the colors, and has been widely used as a benchmark to test community-detection algorithms. The edges in these two examples are undirected, which means the relationships between vertex pairs of the

networks are reciprocal or symmetric, such as the friendship between two users on Facebook and the co-authorship between authors on collaboration networks. However, in many network systems, the relationships or interactions among individuals are not reciprocal. The edge directionality leads to numerous directed networks in diverse domains by nature, such as hyperlinks in the World Wide Web, follower-followee relationships on Twitter networks, citation relationships among scientific papers on citation networks, etc. Figure 2.1(c) is a sample graph of web pages with directed hyperlinks pointing from one page to another. The colors denote the communities detected by Newman and Girvan [101], in which they ignore the edge direction and treat the network as undirected. Moreover, many complex networks exhibit overlapping community structures in the sense that some nodes are characterized with multiple community memberships. Figure 2.1(d) shows an exemplar network of overlapping communities. The red nodes are overlapping nodes that usually play an important role of *inter-community liaisons* in real-life network systems. Further, most complex networks often show a hierarchical organization, with small communities nested within large communities at different scales. Figure 2.1(e) illustrates a schematic network with hierarchical community structure. For instance, it can be depicted as a university of four colleges at the macroscopic level. When zooming into each college, one could find different departments as sub-communities at the mesoscopic level. Within each department, one may make a subdivision of faculty members into various research groups.

Community detection is of great importance in a wide variety of applications. For example, finding communities within social networks enables us to identify groups of densely connected individuals as well as their similar interests and behaviors so as to build more effective recommender systems. Identifying communities in metabolic networks provides insight into how functional modules form and work. Community detection in co-authorship networks helps reveal the collaboration patterns among

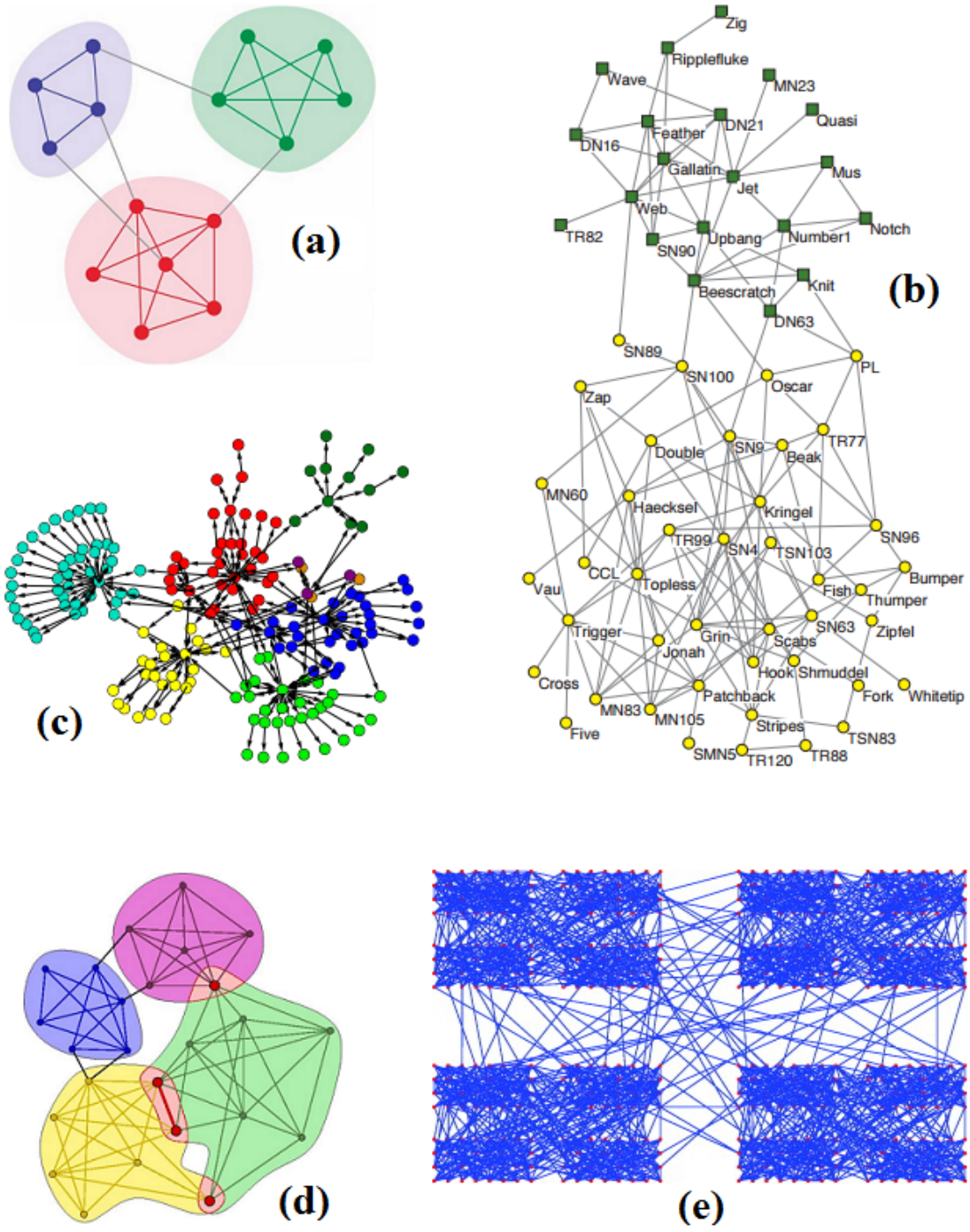

Figure 2.1: Communities in network systems. (a) a simple graph with three communities; (b) Lusseau's dolphin social network [6, 89]; (c) a network graph of web pages with directed links [101]; (d) an example of overlapping communities. Red nodes are overlapping nodes closely related to two communities; (e) a schematic network with hierarchical community structure [78]

scientists. One can also do community detection to find the most influential individuals and take advantage of the community structure to facilitate influence spread through a social network for viral marketing or to improve data forwarding in P2P networks. However, finding communities in complex networks is a difficult task. Research on community detection has a long history in sociology, biology, physics, computer science, economics, etc. It has attracted huge efforts and extensive studies from many disciplines, and a strikingly large number of algorithms have been presented [39, 91, 135]. Nevertheless, this problem is not yet satisfactorily solved.

A critical obstacle is the lack of a quantitatively precise definition of community. Some researchers refer to a community as a densely connected subgraph in the network graph topology [100, 131]. It is also depicted as a functional module, a set of members who share similar properties, a chain of adjacent cliques, or a k clan/club [104]. The widely-used definition follows the basic intuition that describes a community as a group of nodes with higher internal than external connectivity. However, this notion of connectivity is ambiguous, and results in many different objective functions and performance metrics. From hierarchical clustering [47], graph partitioning [116], and spectral methods [33] to density-based clustering [118], modularity maximization [9, 100], statistical mechanics [106], and label propagation [108], most existing algorithms are rooted in degree or betweenness centrality, edge density, or random-walk-based closeness. While these notions capture the intuition of network connectivity to some extent, the existing literature [39, 82, 139] suggests there are still significant areas for improvement. Further, as mentioned above, numerous networks in diverse domains are directed and/or have weights on edges to differentiate the strength of the connections. Nevertheless, most existing algorithms ignore the link direction and/or the weights, which may cause considerable loss of information and lead to unreliable or even misleading results. Moreover, complex network systems often exhibit hierarchical community structure in which small communities succes-

sively group together to form larger ones. However, the number of algorithms that can identify hierarchical communities in complex networks is limited, and those that do are usually not scalable to large-scale networks. To this end, we propose the following general but essential design principles for developing new community-detection algorithms, and suggest using them as evaluation criteria for a more comprehensive performance comparison across various algorithms.

- **Applicability:** Many algorithms have been proposed for undirected binary networks. However, directionality and weights are usually essential features of diverse networks. Ignoring the edge direction and weights fails to capture the asymmetric relationship or the strength of the relationship between vertex pairs. An *applicable* algorithm should be able to incorporate the directionality and weights such that it can be applied to not only undirected binary networks but also directed/weighted networks.

- **Quality:** An algorithm should be evaluated using a set of well-known network benchmarks with or without known communities, i.e., ground truth. For a network with known communities, the widely-used *normalized mutual information* (NMI) [30] is suitable for measuring the accuracy of a community partition against the ground truth. For network benchmarks without ground truth, *modularity* [97] is a well-defined measure for the quality of a community partition. A *reliable* algorithm should demonstrate high accuracy in terms of NMI scores on benchmarks with ground truth, and also find community partitions with high modularity scores on benchmarks without ground truth.

- **Hierarchy:** Real-life network systems often exhibit some form of hierarchical organization; large communities are generally found to be made up of smaller, tighter ones. We argue that even the "ground truth" of the network benchmark is actually not unique but is subject to the desired granularity level in the hi-

erarchy of the community structure. Moreover, the ground-truth community partition does not have to be the one that maximizes the modularity score. Instead, it could be the one that most closely matches the hierarchical community structure at some specific scale, which partially explains why exhaustive modularity maximization algorithms often fail to arrive at the ground truth of real-life network benchmarks. Therefore, a *robust* algorithm should be able to find hierarchical communities and enable us to zoom into the community hierarchy at different scales.

- **Scalability:** In the era of big data, a network may grow unprecedentedly large in size. However, most existing algorithms are too computationally expensive to be scalable for large-scale networks. A *feasible* algorithm should be not only effective but also efficient enough to be of promising scalability, especially if we require the discovery of hierarchical community structure.

We attempt to find a more precise measure to decode the connectivity and proximity embedded in the network graph topology, and use this measure to extract community structure. Our work is motivated by observations from real-world communities. Community members have individual social roles: leaders, core members, liaisons between communities, etc. Some may be even simultaneously associated with multiple communities. Some are more influential than others, and some are more susceptible to influence. We argue that individual roles, influence, and susceptibility are implicitly embedded in the network topology. In fact, it is influence that not only differentiates individual roles but also acts as the force holding the individuals together to form and maintain the community. On the other hand, we notice a simple but meaningful *shared-nearest-neighbor* (SNN) similarity [63] used in traditional clustering, which indicates that two nodes both being close to a set of neighboring nodes suggests they are close to each other. This is naturally extended to our influence-based scenario, and can be rephrased as: *that two nodes both influencing a set of*

(direct and indirect) neighbors suggests they are close to each other. We refer to it as *shared-influence-neighbor* (SIN) similarity, which measures the closeness of any pair of nodes in terms of their mutual influence and the common set of nodes they both influence. Our SIN similarity captures the intuitive notion of community: *that two nodes influencing each other and a common set of neighbors confirms their closeness and implies that they are more likely in the same community.*

All of this makes influence a natural context for community detection, but the question is how to define a quantitatively precise measure of influence. This leads to another widely-studied topic, namely, influence analysis. One fundamental step in influence analysis is to differentiate the relative influence significance among the nodes, where influence is often characterized using various centralities. The four most widely-used measures of centrality in network analysis are *degree* centrality, *closeness* centrality, *betweenness* centrality, and *eigenvector* centrality. Each indicates some specific strength of a node's structural role in the network. However, these measures are not fine enough to quantify a node's comprehensive strength in terms of the total amount of information or influence it can spread through the network as a seed node. Further, none of them is able to measure the influence-based proximity between vertex pairs, which as indicated could be a desirable metric for community detection.

We use concepts and techniques from the fields of network modeling, artificial intelligence, data mining, and social science to exploit the *influence-based* network connectivity and vertex-pair similarity embedded in the network graph topology. We arrive at a novel *reachability-based* influence-diffusion model [127, 129], which builds egocentric diffusion trees and generates an influence vector for each node. Using this model, we define a new *influence centrality* that differentiates the nodes' relative significance in terms of the total amount of influence a node spreads out or the multiple-path reachability of a node to other nodes. More importantly, we naturally quantify the SIN similarity of a pair of nodes using their respective influence vectors,

which gives rise to a new, refined vertex-pair closeness metric. We apply the SIN similarity to spherical K-means clustering [31], and develop an *influence-guided spherical K-means* (IGSK) algorithm for community detection. Extensive tests demonstrate the effectiveness of IGSK and verify the validity of our SIN similarity and influence-diffusion model. Moreover, our approach can be easily adapted to identification of overlapping communities and individual roles in each community.

To find hierarchical communities and improve the efficiency, we develop two novel *influence-guided label propagation* (IGLP) algorithms, *IGLP-weighted-ensemble* (IGLP-WE) and *IGLP-direct-passing* (IGLP-DP) [128]. Our work on IGLP-WE is motivated by the well-known *label propagation algorithm* (LPA) [108] and our SIN similarity. LPA is basically an ensemble-based label association method, in which each node adopts the label that the majority of its neighbors have in an iterative process until a global consensus is reached. Thanks to its nearly linear time complexity, this algorithm has received much attention. Unfortunately, it comes with a significant stability issue due to random tie breaking. The problem exists in its majority voting rule, which simply counts the number of neighbors that carry the same label. This is a coarse technique that implicitly assumes the votes of all neighbors have the same weight, resulting in many ties. An intuitive solution is to weight the vote of each neighbor by its closeness to the node of interest. We effectively resolve this issue using our SIN similarity. Furthermore, we develop IGLP-DP inspired by the following intuition: *given only the network graph topology, any node should belong to the same community as its closest (most structurally-similar) neighbor*. We apply this idea to a new label-propagation framework: once a node identifies its closest neighbor using SIN similarity, it passes its community label to that neighbor, which in turn passes the label to its closest neighbor and so on. In the end of the label propagation, the nodes of the same labels are grouped into respective sub-communities. Each sub-community is composed of a group of closest neighbors chained together. These sub-communities

constitute the initial, most compact configuration of the community structure.

For both IGLP-WE and IGLP-DP, we regard the resultant community partitions directly derived from the label-propagation process as the building blocks, and perform agglomerative hierarchical clustering using a novel boundary-node-based cluster-proximity measure to reveal a complete community hierarchy. We conduct extensive tests on a set of real-life networks and synthetic benchmarks, and demonstrate their superior performance in terms of high quality and efficiency in both undirected/directed and unweighted/weighted networks.

2.2 Related Work

Community detection and influence analysis are essential tasks in network analysis. They have received extensive attention and great efforts from many disciplines. Especially, a plethora of algorithms have been presented for community detection over the years. An in-depth survey can be found in [39, 91, 135]. We focus here on papers that are most relevant to our concerns and considerations.

2.2.1 Centralities

Centrality concepts were originally developed and well studied in social network analysis. Centrality refers to the identification of the most important or the most influential nodes in a social network. A set of noteworthy centrality measures were presented and discussed in details in [131]. There are four widely-used centralities that measure the significance of a node from different perspectives. They are *degree* centrality, *closeness* centrality, *betweenness* centrality, and *eigenvector* centrality.

Degree centrality is simply quantified by the number of links/neighbors of individual nodes. Obviously it is a very coarse measure. There are many ties and it fails to take into account the structural information or the significance of neighbors. *Closeness* centrality [114] is defined as the inverse of the sum of lengths of the shortest paths of a node to all other nodes. It measures how fast a node can spread infor-

mation in the network. *Betweenness* centrality [42] quantifies the number of times a node appears in the shortest path of two other nodes. It can be regarded as a measure of how often a node acts as a broker or gatekeeper of information flow. The closeness and betweenness centralities are both based on the shortest paths. However, the spread of information does not always go along the shortest paths in reality. To address this issue, researchers present a number of variants. Brandes and Fleischer [14] assume information spreads like an electrical current, and propose *current-flow closeness/betweenness* centralities. However, information spread is significantly different from current flow. There exist two inherent defects in this approach. First, information spread in networks does not split like current in electrical networks. More precisely, information spreads through replication rather than transfer in the sense that one does not lose the information after she passes it on to others. But the current in an electrical network follows the Kirchhoff's current law, which ensures there should be as much current entering each node as leaving it. Second, current can only flow from a high-potential node to low-potential nodes (Kirchhoff's potential law). However, the information spread does not have this directionality. There are some other variants, such as *information* centrality [117], *random-walk closeness* [102], *random-walk betweenness* [99], *maximum-flow betweenness* [43], etc.

Eigenvector centrality [11] is the component of the principal eigenvector of the adjacency matrix, which captures an intuitive but important concept: *connecting to a more influential node contributes more influence weight to the node of interest than connecting to a less influential node.* Unfortunately, it fails to capture the fact that influence is attenuated when passing through the network. The well-known PageRank [103] is a variant of eigenvector centrality. For undirected graphs, PageRank degenerates into degree centrality. *Katz* centrality [68] can be regarded as a generalization of degree centrality and eigenvector centrality. It measures the influence significance of a node by counting the number of walks of length one to infinity, while

the contribution of long walks are penalized. Let A denote the adjacency matrix of a network of N nodes, k denote the length of a walk from nodes i to j, and α denote an attenuation factor in (0,1). Then *Katz* centrality of node i is written as

$$C(i) = \sum_{k=0}^{\infty} \sum_{j=1}^{N} \alpha^k (A^k)_{ij}.$$

Note that the (i, j)-entry of the k^{th} power of the adjacency matrix, $(A^k)_{ij}$, gives the total number of walks of length k from nodes i to j. Unlike the *closeness* centrality that considers only the shortest paths, *Katz* centrality takes into account all the walks. However, there is one significant defect from the perspective of information/influence diffusion. For any undirected graphs/links or directed graphs with cycles, it allows the same information to be transmitted around in a loop many times up to infinity. This is not meaningful or realistic. In addition, its attenuation factor α is a user-specified parameter, which makes its influence ranking nondeterministic and unreliable.

2.2.2 Connectivity Metrics

As discussed above, there is no unanimous agreement on the formal definition of community, and the notion of internal/external connectivity is too general and ambiguous, which leads to many different objective functions, performance metrics, and algorithmic recipes. We list in Table 2.1 a set of commonly-used connectivity and performance metrics in the literature, where $G(V, E)$ denotes an undirected graph with a total of n nodes and m edges, S is a subgraph with a set of nodes, n_S is the number of nodes in S, m_S is the number of internal edges in S, c_S is the number of boundary edges of S (also called the cut size of S), $d(u)$ is the degree of node u, $m_{G \setminus S}$ is the total number of edges of the rest of the graph $G \setminus S$, $u_i^{int}(S)$ and $u_i^{ext}(S)$ are the number of internal and external edges for node u_i in S, respectively.

From the simplest *node degree* [107] to the most popular *modularity* [100, 101], many connectivity metrics are commonly used in the literature, such as *internal den-*

Table 2.1: Some representative connectivity/performance metrics

Metrics	**Formulas**	**Comments**
Node Degree	$u_i^{int}(S) > u_i^{ext}(S), \forall u_i \in S$	*Strong community*: each node has more internal connections than external connections
	$\sum u_i^{int}(S) > \sum u_i^{ext}(S), \forall u_i \in S$	*Weak community*: the sum of all degrees within S is greater than the sum of all degrees toward the rest of G
Internal Density	$\frac{2m_S}{n_S(n_S-1)}$	The ratio of the number of internal edges to the number of all possible internal edges
Conductance	$\frac{c_S}{min(m_S, m_{G\setminus S})}$	The fraction of edge volume of the cut
Ratio Cut	$\frac{c_S}{n_S(n-n_S)}$	The ratio of the boundary edges to all possible edges across the boundary
Normalized Cut	$\frac{c_S}{2m_S+c_S} + \frac{c_S}{2(m-m_S)+c_S}$	The cut cost as a fraction of the total edge connections to all the nodes in the graph
Average ODF	$\frac{1}{n_S}\sum \frac{u_i^{ext}(S)}{d(u_i)}, u_i \in S$	ODF: Out-Degree-Fraction. The average fraction of edges pointing outside of S over the node degree
Modularity	$\frac{1}{4m}[m_S - E(m_S)]$	$E(m_S)$ is the expected number of edges in S in a random graph with the same node-degree distribution

sity [82], *conductance* [66], *ratio cut* [133], *normalized cut* [116], *average out-degree fraction* [38] and so on. Leskovec et al. [82] present an empirical comparison of a range of community-detection algorithms that are based on the above and some other similar metrics. They test a total of eight different classes of algorithms with 12 common objective functions on a set of more than 40 networks. They point out that these intuitive notions of cluster quality tend to fail as one aggressively optimizes the community score, and conclude that approximate optimization of the community score introduces a systematic bias into the extracted clusters. Another evaluation of various objective functions based on those connectivity metrics is proposed by Yang et al. [139]. Their experimental results also cast doubt on the quality of those commonly-used connec-

tivity metrics and corresponding objective functions. For instance, RankClus [120], on one hand, achieves the best performance on the Cities-Services dataset [139] in terms of *internal density*, and however, its scores of *conductance*, *ratio cut*, and *modularity* suggest the worst quality of the detected communities. On the other hand, it correctly clusters all nodes of Zachary's karate-club dataset [141], but its *internal density* gives much lower score than some other algorithms. WalkTrap [106] totally fails to cluster the Mexican Political-Power dataset [46] and the Cities-Services dataset despite perfect *conductance* and *ratio-cut* scores. Overall, none of those connectivity or performance metrics is consistently reliable.

Among all the connectivity/performance metrics discussed above, modularity is the most widely used one. It was originally proposed by Newman and Girvan [100,101] to measure the quality of a given partition of the network in terms of how much the resultant community structure deviates from random graphs that have the same expected degree distribution as the network under consideration. More precisely, let A_{ij} denote the (i,j)-entry of the adjacency matrix of the network with $k_i = \sum_j A_{ij}$, $k_j = \sum_i A_{ij}$, and $m = \frac{1}{2}\sum_{ij} A_{ij}$. The modularity is given by

$$Q = \frac{1}{2m}\sum_{ij}\Big[A_{ij} - \frac{k_i k_j}{2m}\Big]\delta(c_i, c_j),$$

where c_i and c_j are the community labels of nodes i and j respectively, and the Kronecker delta function $\delta(c_i, c_j) = 1$ if $c_i = c_j$ (i.e., nodes i and j reside in the same community) or 0 otherwise. It is worth pointing out that the modularity defined above was initially introduced for undirected binary networks, but the same formula is actually applicable to directed and weighted networks as well [5,97], provided that the adjacency matrix A is interpreted accordingly. For an undirected binary network, $A_{ij} = 1$ if there is an edge connecting nodes i and j or 0 otherwise; A is symmetric, k_i and k_j are degrees of nodes i and j respectively, and m is the total number of edges in the network. For a directed binary network, $A_{ij} = 1$ if there is a link pointing

from node i to node j or 0 otherwise; A is not symmetric, k_i is the out-degree of node i, k_j is the in-degree of node j, and $2m$ is the total number of links in the network. For a directed weighted network, A is the weighted adjacency matrix of the network, and $A_{ij} = w_{ij}$ if there is a link pointing from node i to node j with a weight of w_{ij} on the link (0 if no link exists); A is not symmetric, k_i is the total output weight/strength of node i, k_j is the total input weight/strength of node j, and $2m$ is the total weight/strength of the whole network.

Modularity is the best-known objective function and quality metric for community detection. It makes intuitive sense that higher modularity scores indicate better partitions and clearer community structure. Motivated by this assumption, researchers present many algorithms that use various optimization techniques to maximize the modularity [9, 27, 52, 98, 100, 101]. However, modularity-based methods also have crucial limits in spite of their popularity. Guimera et al. [54] demonstrate random graphs may have partitions with large modularity values due to fluctuations in the edge distribution. Fortunato and Barthelemy [40] suggest a more fundamental issue, showing that modularity optimization has a resolution limit that may prevent it from identifying well-defined communities below a certain size. There is another serious issue. Good et al. [48] reveal that there typically exist an exponential number of distinct partitions with suboptimal high modularity scores and there typically lacks of a clear global maximum. We argue that the ground-truth community partition does not have to be exactly the one that maximizes the modularity. These two factors explain why exhaustive modularity-maximization algorithms often fail to find the ground-truth communities. In fact, self-organization is a common feature of most networks. The formation of community structure in networks, especially in large-scale complex networks, is not necessarily the result of a top-down centralized/global optimization procedure. Instead, it is the outcome of a bottom-up decentralized/local aggregation process. Hence, we suggest using the modularity as a quality metric instead of a

single objective function to maximize aggressively in the first place.

2.2.3 Similarity Measures

Another important class of community-detection algorithms address the problem using various similarity measures. With the success of the application of similarity/distance measures in conventional data-mining clustering analysis, it is natural to bring in those notions and techniques for community detection in networks. Especially, it makes perfect sense that members in the same community are more similar or closer to each other than to the rest of the network. However, defining a meaningful and quantitatively precise similarity measure in connectivity-based graph topology is not straightforward.

Superficially, the shortest path between a pair of nodes seems like a direct measure of their distance. Unfortunately, it is not fine enough to differentiate the closeness of nodes within the context of community-structural characteristics in networks since a single edge can easily link a node deeply located in one community to a node densely connected in another community. Alternatively, one could take into account all paths running between two nodes. To capture that information can in fact spread along paths other than the shortest path, Estrada and Hatano [35] define the *communicability* of the vertex pair (i, j) as the weighted-sum of all the walks connecting them, that is,

$$G_{ij} = \sum_{k=0}^{\infty} \frac{(A^k)_{ij}}{k!}.$$

Recall that the (i, j)-entry of the k^{th} power of the adjacency matrix, $(A^k)_{ij}$, is the total number of walks of length k from nodes i to j. Since the nodes/edges can be revisited many times along the way, the total number of walks between any vertex pair is infinite, and the walk length could be infinitely large. Consequently, they use the inverse factorial of the walk length as the weight to reflect shorter walks make larger

contribution to the communicability than longer ones. They end up developing a community-detection algorithm using the concept of the communicability graph [36]. However, this *communicability* has inherent defects like *Katz* centrality. It does not make sense to allow the same information to be repeatedly transmitted around in a loop. In addition, although it seems straightforward to extend it to directed networks, it is not applicable to weighted networks.

Many sophisticated measures of vertex-pair similarity are proposed based on various stochastic models. Zhou and Lipowsky [145] introduce the *proximity index* between neighboring vertices using a biased Brownian motion. More precisely, for each nearest-neighboring pair of vertices i and j, the proximity index is defined as

$$\Lambda(i,j) = \frac{\sqrt{\sum_{k\neq i,j}(d_{ik} - d_{jk})^2}}{N-2},$$

where N is the total number of vertices in the network, and d_{ik} and d_{jk} are the mean-first-passage-time, i.e., the average number of steps the Brownian particle takes before it reaches vertex k for the first time starting from vertices i and j, respectively. An important feature of the proximity index is that it measures the closeness of vertex pair (i,j) in terms of the difference of the mean-first-passage-times of vertices i and j to any other vertex k instead of directly using the mean-first-passage-time between i and j. This technique captures more information on the community structure. They further define the proximity index between two communities and propose an agglomerative hierarchical algorithm (called *Netwalk*) for community detection. However, this algorithm suffers from a high time complexity of $O(N^3)$ since it needs to do matrix inversion to find the proximity indices for all nearest-neighboring vertex pairs. It also needs to tune a bias coefficient to get the best performance, and it does not work for directed networks.

Random walk is another commonly-used stochastic model used for defining sim-

ilarity measures for community detection. The underlying intuition is that random walks on a community-structured graph have a much higher chance to get trapped in a community than to travel between communities. Fouss et al. [41] exploit the *average-commute-time distance*, $n(i,j)$, which is quantified by the average number of steps for a random walker to take to go from node i to node j for the first time and then get back to node i. They find out that the square root of $n(i,j)$ is also a vertex-pair distance measure (called *Euclidean-commute-time distance*), which is closely related to the pseudo-inverse of the Laplacian matrix of the graph (L^+). By applying a sigmoid transformation on L^+, Yen et al. [140] propose a *sigmoid commute-time kernel* and perform the kernel clustering for community detection. Like the *Netwalk* algorithm discussed above, this kernel-based method does not scale well on large networks due to high computational cost. It is not applicable to directed networks either, and some parameters need to be tuned as well.

Pons and Latapy [106] propose another random-walk-based distance measure that quantify the structural similarity of vertex pairs. Let A be the adjacency matrix of an undirected network of n vertices and m edges, and $d(i) = \sum_j A_{ij}$ be the degree of vertex i. Then $P_{ij} = \frac{A_{ij}}{d(i)}$ is the transition probability from vertex i to vertex j, which defines the transition matrix P of the random walk processes. $(P^t)_{ij}$ gives the probability of going from i to j through a random walk of length t. They define the distance between vertices i and j as the degree-weighted Euclidean distance between their respective probability distribution, i.e.

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}}.$$

They generalize this vertex-pair distance to a distance between two communities and propose an agglomerative hierarchical clustering algorithm (called *Walktrap*) for community detection, which achieves a time-complexity of $O(n^2 \log n)$ in most cases and

$O(mn^2)$ in the worst case. Unfortunately, it is still intractable for large-scale networks. Especially, it requires a large amount of memory due to a space complexity of $O(n^2)$. While they claim it is directly usable for weighted networks, this algorithm does not work for directed networks. Moreover, the random-walk length t is a nondeterministic parameter that needs to be tuned for each network individually. It must be long enough to capture the community structure, but can not be too long to avoid reaching the stationary distribution, in which the transition probabilities degenerates into the degree of the vertices.

Another approach is to directly project the graph topology into a Euclidean space so that we can use well-defined spatial measures (like Euclidean distance or cosine similarity) and a variety of well-studied clustering methods. The spectral-based method proposed by Donetti and Muñoz [33] is such an example. Each node in the network is mapped to a point in a D-dimensional space in which the coordinates are given by its projections on the first D nontrivial eigenvectors of the Laplacian matrix. They apply the Euclidean or *angular* distance defined in this eigenvector space to a standard hierarchical clustering for community detection. They find that the angular distance always generates better results than the Euclidean one. The time complexity is determined by the computation of all eigenvectors, leading to $O(n^3)$ for most cases. Furthermore, the optimal value of D is not known a *priori*. They suggest repeating the hierarchical clustering using all possible values of D to find the one that gives the largest modularity score. Obviously, this algorithm is too computationally expensive to be feasible for large-scale networks. In addition, it is not applicable to directed and/or weighted networks.

Our *shared-influence-neighbor* (SIN) similarity [127, 129] also is a method that projects the graph topology into a Euclidean space. We map each node to a point in an n-dimensional influence space, in which the coordinates are given by its influence vector derived from our influence-diffusion model. The SIN similarity between any

pair of nodes is quantified using their respective influence vectors, which captures much structural information in terms of their mutual influence and the common set of nodes they both influence. We will elaborate our methodology in Section 2.3.

2.2.4 Label Propagation

The *label propagation algorithm* (LPA) for community detection is originally proposed by Raghavan et al. [108]. The main idea is to iteratively run a label-updating process until each node adopts the community label that the majority of its neighbors carry. We remark here that this idea implicitly uses the *strong-community* definition (see Table 2.1) as its objective function and stop criterion. Consequently, this LPA always delivers strong communities. When the community structure is relatively weak or fuzzy, LPA tends to arrive at one or several *monster* communities. Suppose that a node x has j neighbors and let $C_x(t)$ denote the community label of node x at the t^{th} iteration. The main steps of LPA can be described as follows:

1. Initialize the label of each node with its node index at $t = 0$, i.e., $C_x(0) = x$. Then set $t = 1$;

2. Randomly choose an order in which the nodes update their labels;

3. For each node x selected in the specified order, update its label using the majority voting rule, i.e., $C_x(t) = f(C_{x_1}(t), ..., C_{x_i}(t), C_{x_{i+1}}(t-1), ..., C_{x_j}(t-1))$, where $x_1, ..., x_i$ are the neighbors of x that have already updated their labels in the t^{th} iteration, $x_{i+1}, ..., x_j$ are the neighbors of x that have not updated their labels yet, and f returns the label that the maximum number of neighbors carry (ties are broken randomly);

4. If every node has the same label as the maximum number of its neighbors have, group the nodes sharing the same labels into respective communities and stop the algorithm. Otherwise, set $t = t + 1$ and go back to Step 2.

The label-updating process in Step 3 is asynchronous. In synchronous updating, node x at the t^{th} iteration updates its label based on the labels of its neighbors at $(t-1)^{th}$ iteration, i.e., $C_x(t) = f(C_{x_1}(t-1), ..., C_{x_j}(t-1))$. Raghavan et al. propose the use of asynchronous updating to avoid oscillations of labels that may occur in synchronous updating. This algorithm achieves a time complexity of $O(km)$, where k is the number of iterations and m is the total number of edges in the network. Although k varies from one network to another, the authors claim that k is independent of the network size and that 95% or more of the nodes acquire the labels that the maximum number of their neighbors have by the end of five iterations. It has gained much attention due to its simplicity and nearly-linear time complexity. However, it has a significant stability issue. It produces different community partitions in different runs due to the random tie breaking and the random ordering of nodes in its label-updating process, making the algorithm nondeterministic and unreliable.

Leung et al. [84] contrast asynchronous with synchronous updating, and show that synchronous updating is more stable on average but converges much more slowly than asynchronous updating. They also introduce a hop-attenuation parameter δ $(0 < \delta < 1)$ that governs how far a particular label can spread as a function of the geodesic distance from its origin. Although this technique may help avoid the formation of monster communities and thus improve the overall performance, this additional parameter adds extra uncertainties to the algorithm.

To improve the stability, Xing et al. [137] propose a *node influence based label propagation algorithm* (NIBLPA), in which they attempt to avoid the complete randomness using *node influence*. They define node influence based on the k-shell values of a node and its neighbors. A k-shell is a maximal connected subgraph in which every node's degree is at least k. The k-shell value of a node is k, which indicates that node belongs to a k-shell but not to any $(k+1)$-shell. For a node i, let $K_s(i)$ denote its k-shell value and $N(i)$ denote the set of neighbors of node i. Then the

node influence of i is defined as

$$NI(i) = K_s(i) + \alpha \times \sum_{j \in N(i)} \frac{K_s(j)}{d(j)},$$

where $d(j)$ is the degree of node j and α is a parameter in range (0,1). They fix the node ordering of label updating in the descending order of the node influence. They also define a *label influence* based on the node influence to break the tie when multiple labels are carried by the same maximum number of neighbors of the node under consideration. They improve the stability to some extent, but the quality of the resultant community partition is not consistently satisfactory. In fact, the node influence is not fine enough to rank all the nodes in the network, which implies there still exist random selections in the node ordering of the label-updating process. Moreover, they have to tune the parameter α for each individual network to get the best result.

Based on the idea of simulating the propagation of labels in the network, Xie and Szymanski [136] propose a stabilized label propagation algorithm called *LabelRank*. They introduce four operators to control the label propagation: *propagation*, *inflation* Γ_{in} (*in* is a parameter taking on real values), *cutoff* Φ_r (r is a threshold in [0,1]), and *conditional update* Θ_q (q is a parameter chosen from [0,1]). The label propagation stops when the number of nodes that potentially change their communities exceeds a predefined frequency *numChange*. LabelRank resolves the randomness issue of LPA and improves the performance. However, it requires adjusting the four parameters, which is cumbersome and impractical. In fact, it partially shifts the randomness issue from tie breaking and node ordering to the selection of those parameters. In this regard, the stability issue of LPA still remains unsolved. In addition, most LPA algorithms, including all the aforementioned ones, mainly focus on undirected and unweighted networks. Few LPA algorithms in the literature indicate justified

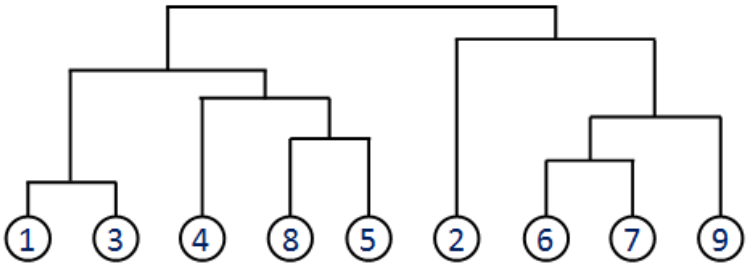

Figure 2.2: A schematic dendrogram of a small network

applicability for directed and/or weighted networks.

2.2.5 Hierarchical Methods

Many network systems are characterized with a hierarchical organization, in which small communities are nested within larger ones successively at different scales. Finding hierarchical communities in complex networks is highly desirable but even more challenging. It is tempting to use the well-studied hierarchical-clustering techniques in data mining, which build a hierarchy of clusters. Those techniques can be generally classified into two categories. One is a top-down *divisive* approach, which starts with all the nodes in one cluster and iteratively splits clusters into smaller ones; the other is a bottom-up *agglomerative* approach, which starts with each node in its own singleton cluster and iteratively merges the closest clusters to form larger ones. For both approaches, the resultant community hierarchy can be illustrated as a *dendrogram*, a hierarchical tree structure. Figure 2.2 is a schematic example. The top level represents a single community of the network as a whole, the leaves at the bottom represent individual nodes with each of them in a singleton community of its own. A horizontal cut through the tree at any level gives a set of subgraphs or communities at a particular scale.

The divisive approach focuses on identifying and removing the *inter-community* edges that connect vertices of different communities. The most representative algo-

rithm is the one proposed by Girvan and Newman [47]. They generalize Freeman's betweenness centrality [42] of vertices to edges, and define the *edge betweenness* of an edge as the number of shortest paths between all vertex pairs that run along it. It is intuitive that inter-community edges have high edge betweenness. At each iteration step of their algorithm, they recalculate the edge betweenness for all edges and remove the edge with the highest betweenness. The algorithm iterates until no edges remain. The main drawback of this method is its high computational cost. For a network of n nodes and m edges, calculating the betweenness of all edges runs in time $O(mn)$. Since it needs to iterate m times to remove the edges one at a time, the complete algorithm runs in time $O(m^2n)$, which is intractable in large-scale networks. There are some variants [39] that use different betweenness centralities or attempt to reduce the time complexity. Unfortunately, the improvement is either limited or a tradeoff between accuracy and efficiency to some extent.

The agglomerative approach draws more attention than the divisive one. In general, once a vertex-pair similarity (or closeness) measure is defined, it can be generalized to define the cluster proximity, and thus the agglomerative hierarchical clustering can be applied. However, defining a meaningful and quantitatively precise similarity measure is nontrivial. Moreover, the agglomerative approach often starts with assigning each node to a singleton community, and merges a pair of closest communities at each iteration step. It has to take $(n-1)$ iterations to build a complete hierarchy of communities. Consequently, the conventional agglomerative approach has a time complexity of $O(n^3)$ in the general case, which is too costly to scale well. The *Netwalk* [145] and *Walktrap* [106] algorithms discussed in Section 2.2.3 are two of such examples. The well-known modularity-maximization algorithm proposed by Newman [98] falls in the category of the agglomerative approach as well. Instead of using a vertex-pair similarity measure or any cluster proximity metric, they repeatedly merge communities in pairs such that the merger at each iteration step results

in the greatest increase in the modularity. This algorithm starts with each node in a separate community, and ends at some level of the dendrogram when no mergers of any pairs of communities can further increase the modularity. It achieves a worst-case time complexity of $O((m+n)n)$, or $O(n^2)$ on a sparse graph.

Blondel et al. [9] propose a heuristic algorithm (known as the *Louvain method*), which consists of two phases. The first phase is a local-modularity-optimization process, in which they initially assign each node to a singleton community, and then for each node i, they calculate the gain of modularity in case i is removed from its community and reassigned to the community of its neighbor j. Node i is finally placed in the community for which this gain is maximum. If there is no positive gain, i stays in its original community. This is a sequential sweep over all nodes, and it is repeated until no further gain of modularity can be achieved. The second phase is to rebuild a new weighted network, in which each node is a community found in the first phase and each edge connecting a pair of nodes has a weight that is the sum of the weights of all the inter-community edges of the two communities corresponding to the two nodes. These two phases alternate repeatedly until the maximum modularity is attained. This algorithm is well known for its high efficiency (roughly $O(n \log n)$) and high modularity. This method has an issue that the order of visiting each node may affect not only the the computation time but also the final community partition. In addition, the resultant hierarchy of communities is not complete, as many intermediate levels are skipped.

Huang et al. [59] propose another two-phase hierarchical clustering algorithm (called *SHRINK*). They define a structural similarity between two adjacent nodes in terms of the cosine similarity of their respective adjacent-node sets, and use it in a similarity-based modularity Q_s. In the first phase, they sequentially sweep over all nodes to find local micro-communities based on the structural similarity. In the second phase, they evaluate each local micro-community and shrink it into a super-

node if the gain of Q_s is positive. These two phases are executed in turns until the maximum Q_s is obtained. This algorithm achieves a time complexity of $O(m \log n)$. Like the *Louvain method*, *SHRINK* needs to rebuild a new weighted network in the second phase of each iteration. A subtle issue is that it is not clear whether the intermediate partitions in the dendrogram correctly reflect the community hierarchy of the original network. In addition, neither of them is applicable to directed networks.

Lancichinetti et al. [79] present an alternative approach, which is based on the local optimization of a simple *fitness* function

$$f_C = \frac{k_{in}^C}{(k_{in}^C + k_{out}^C)^\alpha},$$

where k_{in}^C and k_{out}^C are the total internal and external degrees of the nodes in community C, and α is a positive real-valued parameter, controlling the size of the communities. The community structure is revealed by peaks in the fitness histogram, and different hierarchical levels can be investigated by tuning α. Large α yields small communities, and small α delivers large communities. However, it is unknown *a priori* how large the communities are. Even at a fixed scale, the communities may vary in size significantly. There are infinite values of α for selection, but there is no indication which values produce the most reliable community partitions. Moreover, for different values of α, the respective community partitions are usually mixed with each other. It is hard to map a specific partition onto a corresponding level in the dendrogram to clearly show how small communities are nested and merged into larger ones. In addition, this algorithm has a stability issue due to the random selection of seed nodes. The fitness histogram resulting from a specific choice of the seeds is not reliable, thus requiring multiple runs with different seeds to find the most relevant commmunity structures.

2.3 Influence-Diffusion Model

We draw inspiration from the PageRank algorithm in the sense that we cannot solely rely on the node degree. We have to find an intelligent way to embed influence into a node and pass it around in the network. This leads to a novel influence-diffusion model. It is noted that the influence defined in our diffusion model is different from the influence defined in many *activation-based* diffusion models such as the *epidemic* model and the *linear threshold* model [69], in which the influence of a node is quantified by the number of inactive nodes it can activate. Our diffusion model is *reachability-based*. We measure a node's significance by the total amount of influence it diffuses in its neighborhood in terms of *multiple-path reachability*.

Our approach differs from prior work in many ways. From the point of view of centralities, our model extends *degree* centrality from immediate neighbors to multi-step neighborhood, includes not only the shortest paths (that the *closeness and betweenness* centralities rely on) but also other paths, and takes into account neighbors' influence significance (like *eigenvector* centrality) and depth-associated influence attenuation (like *Katz* centrality but without cycling). Our *influence centrality* gives rise to a new, more precise measure of a node's comprehensive strength on diffusing influence in the network as a seed node. Further, we not only find the total influence a node spreads out, but also keep track of where and how much its influence is distributed in its neighborhood so as to construct its influence vector for community detection.

The influence in our diffusion model can be interpreted in terms of a message, an idea, an advertisement, or a rumor. Similar to the word-of-mouth communication or storytelling, the message spreads in the network through parallel replication (like a radio broadcast) rather than transfer since one does not lose the message after he forwards it to another person. Those replicas might not be exactly the same as the original message. They may slightly vary from each other, and thus each of them

can be regarded as a new message. One important and distinguishing feature in our diffusion model is that *no loops or cycles are allowed*. The best way of thinking about this mechanism may be to suppose that anyone who spreads a message needs to endorse it and his endorsement is kept on all replicas of this message along its diffusion paths. Then in case the message circulates back to him, he knows he previously endorsed that message and will not spread it repeatedly. For example, if person A spreads a rumor to person B, B passes it to person C, and if C passes it back to B and A, both B and A will ignore it after they find they have already endorsed it. Cycles are thus avoided. On the other hand, suppose that A spreads a rumor to both B and D. Later when D passes the rumor to B, B will take and keep broadcasting it since B did not endorse that piece of rumor before. From a graph-theoretic point of view, the rumor traverses the network via walks, i.e., both nodes and links can be revisited multiple times. It captures the *multiple-path reachability* from one node to another but without cycling. Another distinctive feature of our diffusion model is that we take into consideration that the message may lose its effectiveness and fidelity while it is transmitted in the network, and its *influence gradually fades away along the diffusion path*. We summarize these features in three important rules as follows:

1. *Cycling is prohibited*. No one should repeatedly exert influence in cycles in the same round of an influence diffusion process. This distinguishes our model from *Katz* centrality and most random-walk-based algorithms.

2. *Revisits along different routes are allowed and independent*. This is a realistic imitation in the sense that the influence originating from an influencer may be delivered to a person via many different routes independently. This distinguishes our centrality from closeness and betweenness centralities which only focus on the shortest path.

3. *Influence gradually dissipates along diffusion paths*. This is a reasonable as-

sumption that captures the influence locality such as the well-known *3-degree-of-influence* phenomenon [24]. Specifically, we set a *depth limit* to 3 by default for all nodes in the network.

Our influence-diffusion model is built upon directed weighted networks, and thus naturally integrate undirected/directed and unweighted/weighted networks into one unified framework. For any undirected edge connecting two nodes, we simply replace it with a pair of directed links pointing to each other of the two nodes. Influence diffusion follows link directions. Each node spreads out influence through out-links, and acquires influence via in-links. Consequently, for a specific real-life network, we need to know for what the link direction represents in its application. For example, in a citation network, if paper i cites paper j, then the network contains a directed link from node i to node j. However, this directed link does not reflect the direction of the influence propagation since it is actually the cited paper j that influences the citing paper i. Thus we need to reverse the citation network to fit into our influence diffusion model. For weighted networks, the weight on a link describes the strength of the relationship between a pair of nodes of interest. Generally, the stronger relationship implies the stronger influence. However, the influence that a node i exerts on its neighboring node j is not solely measured by the *absolute strength* of the relationship from node i to j. Instead, it is determined by the *relative strength* when compared to the strength of influence that node j receives from other neighbors. In other words, it depends on node j's *relative susceptibility* to the influence of node i. Therefore, we should consider all in-link neighbors of node j since nodes acquire influence via in-links. We propose a simple normalization scheme as follows to fit the weight on links in our reachability-based influence scenario. Given a directed link pointing from node i to node j with a raw weight w_{ij}, and L_j denoting the set of node j's in-link neighbors, we define the *normalized susceptibility weight* $\hat{w}_{ij}$ as the ratio of w_{ij} to the maximum raw weight of j's in-links, i.e.,

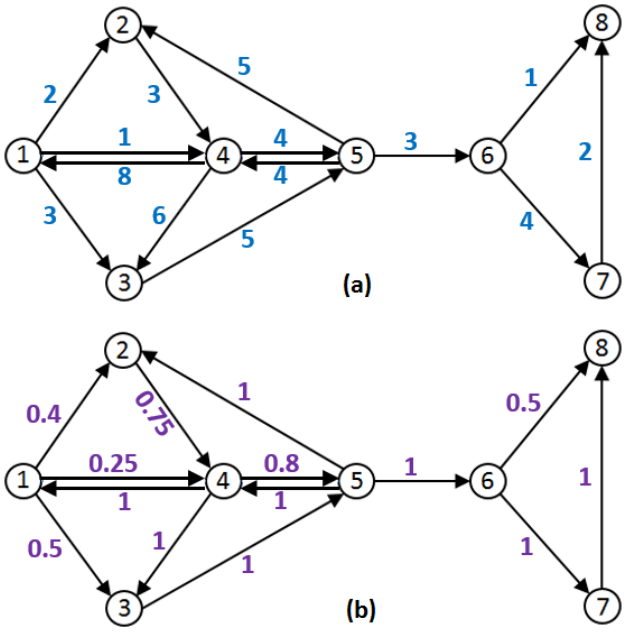

Figure 2.3: Example of a directed weighted network with (a) raw weights, (b) normalized susceptibility weights

$$\hat{w}_{ij} = \frac{w_{ij}}{\max_{k \in L_j} w_{kj}}.$$

As an example, we illustrate in Figure 2.3 (a) a simple directed weighted network. It is noted that the original undirected edge between nodes 4 and 5 is replaced with a pair of directed links with the same raw weight in each direction. The corresponding normalized susceptibility weights are shown in Figure 2.3 (b). We use the normalized susceptibility weight on each link to estimate the weight-associated attenuation of influence when it is transmitted from one node to another following the link direction. When all weights are set to one, this normalization scheme reduces to the case of an unweighted network.

Our influence-diffusion model can be regarded as a constraint branching process on a network graph, in which influence originating from a root node propagates to its offsprings following out-links with two constraints: 1) no node can be an offspring

of its own; and 2) the process dies out within a pre-specified number of generations. We refer to the resultant generation-branching tree as the diffusion tree of the root node. While the influence propagates along a path, it is attenuated in two independent ways. One is the *weight-associated attenuation* in terms of the normalized susceptibility weight on each link. The weight-associated attenuation of influence from a source node to a destination node is the *product* of the normalized susceptibility weights of the corresponding links that constitute the path. The other is the *depth-associated attenuation*. We draw inspiration from the small-world phenomenon and the concentric scales of resolution around a particular node depicted in [34]. It is claimed that the probability of a center node linking to a node at a fixed distance d of the ring is proportional to d^{-2}, which fits well in our influence scenario. We therefore define the depth-associated attenuation as the *inverse square of the depth* from the root node, which can be interpreted as the probability of the root node's influence reaching a node at that depth (number of hops). From a probabilistic perspective, letting a random variable X_i denote the total amount of influence a root node i spreads, and letting a random variable Y denote any diffusion path from i, then we can quantify node i's total influence as the conditional expectation

$$E[X_i] = \sum_y E[X_i|Y = y]P\{Y = y\},$$

where y is a specific path from node i to a destination node j. $E[X_i|Y = y]$ is the expected influence j acquires along the path y, which is estimated by the weight-associated attenuation from i to j. $P\{Y = y\}$ is the probability that the influence diffusion reaches j along the path y, which is exactly the corresponding depth-associated attenuation at the depth from i to j along the path y, as defined above.

We take the simple network shown in Figure 2.3 (b) as an example, and illustrate the diffusion tree of node 1 in Figure 2.4. The first two rules described above are implemented in the construction of the diffusion tree. For example, when the influence

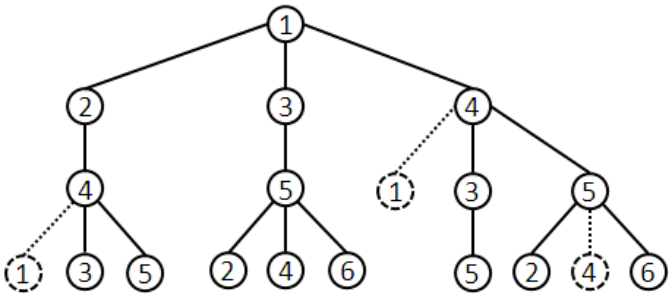

Figure 2.4: Diffusion tree of node 1

goes along the node path $1 \rightarrow 4$ and gets back to node 1, this branch flow stops there since no cycling is allowed. In fact, the loop is not even closed, as indicated by the dashed lines in the figure. Similarly, when the influence goes along the node path $1 \rightarrow 2 \rightarrow 4$, the branch flow going back to node 1 stops propagating before getting back to node 1. On the other hand, node 5 is visited four times along paths $1 \rightarrow 3 \rightarrow 5$, $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$, and so on. Following the diffusion path $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$, the influence that nodes 2, 4, and 5 acquire from node 1 is $0.4 \times \frac{1}{1^2}$, $0.4 \times 0.75 \times \frac{1}{2^2}$, and $0.4 \times 0.75 \times 0.8 \times \frac{1}{3^2}$, respectively.

2.3.1 Influence Matrix

We employ a modified depth-limited search algorithm to explore the diffusion tree of each node and generate an influence vector for each node, which records where and how much influence each node spreads out in its neighborhood within a fixed *depth limit*. Let NSW denote the normalized susceptibility weight, and IV denote the influence vector. The pseudocode in Algorithm 1 shows how we sweep over all the nodes to build the influence matrix that consists of the influence vectors of all the nodes in the network.

Without loss of generality, our algorithm takes a directed weighted network and a depth limit as input. It maintains an open list of to-be-explored nodes and a close list of already-explored nodes, both implemented as a stack (Last-In-First-Out). Each

Algorithm 1 InfluenceMatrix-Builder

procedure IM-BUILDER($G(V, E), n = |V|, depthLimit$)
  Calculate the NSW of each edge
  Set the influenceWeight of each node to 1
  for node $i = 1$ to n **do**
    Empty open/close list
    Set all nodes to be unexplored
    OpenList-PushStack ($Node(i)$)
    $Node(i).depth \leftarrow 0$
    while OpenList is not empty **do**
      $cNode \leftarrow$ OpenList-PopStack ( )
      $Node(i).IV(cNode.nIndex)$ += $(cNode.depth)^{-2} \times cNode.weight$
      Pop all nodes in CloseList with $depth \geq cNode.depth$
      Set those nodes to be unexplored
      if $cNode.depth < depthLimit$ **then**
        for each out-link neighbor j of $Node(cNode.nIndex)$ **do**
          if $Node(j)$ is unexplored **then**
            Create a new OpenList node $nNode$
            $nNode.nIndex \leftarrow j$
            $nNode.weight \leftarrow cNode.weight \times Node(cNode.nIndex).NSW(j)$
            $nNode.depth \leftarrow cNode.depth + 1$
            OpenList-PushStack($nNode$)
          end if
        end for
        Set $Node(cNode.nIndex)$ is explored
        CloseList-PushStack ($Node(cNode.nIndex)$)
      end if
    end while
    $IM(i) \leftarrow Node(i).IV$
  end for
  Return IM
end procedure

node in the open/close lists contains an integer variable *nIndex* that denotes its node index in the network, and another integer variable *depth* that indicates its depth in the diffusion tree of the root node. The node in the open list also contains a variable *weight* that stores the product of the normalized susceptibility weights along the diffusion path. Each node of the network contains a Boolean variable that indicates whether it has been explored so as to avoid cycling.

The algorithm starts with the calculation of the normalized susceptibility weight of each link using the normalization scheme described above, and assigns an initial influence weight of 1 to each node (Lines 2-3). For each iteration, after emptying the open/close lists and setting all nodes to be unexplored, a (root) node is pushed into the open list (Lines 5-8), and then the depth-limited search is explored until the open list is empty. Whenever a node (*cNode*) is popped from the open list (Line 10), we calculate the attenuated influence and accumulate it in the root node's influence vector accordingly (Line 11). Then we pop from the close list all the nodes whose depths are greater than or equal to the depth of node *cNode*, and set all of those nodes to be unexplored (Lines 12-13). This is the mechanism that allows revisits from different routes. Then we check whether the depth of node *cNode* is less than the depth limit. If it is, we continue the exploration by pushing to the open list all of the unexplored *out-link neighbors* of *Node(cNode.nIndex)*. For each of them, we create a new open-list node *nNode* that records the node index, current depth, and the chain product of the normalized susceptibility weights (Lines 15-23). Finally, we mark *Node(cNode.nIndex)* as explored, and push it into the close list (Lines 24-25). Each iteration of the *For* loop generates an influence vector for a specific node, which contains all the nodes it influences associated with the corresponding influence value. After sweeping over all the nodes, the algorithm creates an influence matrix for the network as a whole.

We also develop a closed form of the influence matrix for unweighted networks.

Let A denote the adjacency matrix of an unweighted network (without self-loops). The (i, j)-entry of the n^{th} power of the adjacency matrix, $(A^n)_{ij}$, gives the number of paths of length n from node i to node j. Let D_n denote the diagonal matrix of A^n; its entry d_{ii} is the number of paths of length n for node i to walk to itself. Then the influence matrix M up to a depth limit of 3 is given as

$$M_0 = I$$

$$M_1 = M_0 + A$$

$$M_2 = M_1 + \frac{1}{4}(A^2 - D_2)$$

$$M_3 = M_2 + \frac{1}{9}[(A^2 - D_2)A - D_3 - AD_2 + A \otimes A^T].$$

M_0 is the initial assignment of influence weight of 1 to each node at depth 0, where I is the identity matrix. M_1 is simply the first-step influence propagation. In M_2, we avoid two-step cycling by subtracting D_2 from A^2 and then multiply it by 2^{-2}, which is the two-step influence attenuation coefficient. In M_3, we first let all the nodes on the second depth propagate to depth 3, which is represented by $(A^2 - D_2)A$, then subtract the three-step cycling D_3 of the root node and the two-step cycling of all the first-step nodes AD_2. For all those first-step nodes that link to the root node with an undirected link, we remove them twice, one in D_2A and one in AD_2. And so we get one back by adding $A \otimes A^T$, which is the component-wise multiplication of matrix A and its transpose matrix A^T. Finally, we multiply by 3^{-2} to reflect the three-step influence attenuation. In practice, we do not need to create an $n \times n$ influence matrix. Instead, Each influence vector is represented by a compact dynamic array, in which we only store the node index and the corresponding influence value of those nodes reachable from the respective root node. This implementation not only significantly reduces the space complexity, but also improves the time complexity.

2.3.2 Influence Centrality

As described above, the influence significance of a node is quantified by the total influence it spreads out in the network. Once the influence matrix is built, it is straightforward to compute the influence significance of each node as the sum of all elements in the corresponding influence vector (a row vector in the influence matrix). Let $R(i)$ denote the influence significance of node i and N denote the total number of nodes in the network. We can write it as

$$R(i) = \sum_{j=1\ (j \neq i)}^{N} M_{ij}.$$

Since the root node never distributes its own influence to itself, each diagonal element of the influence matrix has a value of 1. Thus M_{ii} is not included in the calculation of the influence significance even though it does not change the influence ranking. We refer to this influence significance as *influence centrality*. The pre-specified depth limit is a nice gauge to measure the influence at different scales. When the depth limit is set to 1, our influence centrality reduces to degree centrality.

Further, an important characteristic is hidden in the influence matrix. Let $Row(i)$ and $Column(i)$ denote the influence matrix's i^{th} row vector and i^{th} column vector, respectively. Then $Row(i)$ is the influence vector of node i that describes where and how much influence node i distributes in the network. Interestingly, the column vector $Column(i)$ is exactly a representation of where and how much influence node i acquires from the network. In other words, $Row(i)$ consists of the set of nodes that are influenced by node i, and $Column(i)$ represents the set of nodes that influence node i. The summation of all the elements in $Column(i)$ is the total influence node i receives from other nodes, which could be a good indicator of susceptibility ranking among all the nodes in the network.

2.3.3 Shared-Influence-Neighbor Similarity

From a geometric perspective, for a network of N nodes, our influence-diffusion model projects the graph into an N-dimensional influence space, in which each node defines one dimension. The position of each node in this space is determined by its influence vector. These influence vectors incorporate rich structural connectivity information and enable us to differentiate the vertex-pair similarity in a more detailed and precise manner. For a pair of nodes i and j with their respective influence vectors denoted by V_i and V_j, we can measure their similarity S_{ij} with the cosine similarity of V_i and V_j. It can be done in two steps. First, we normalize V_i and V_j into respective unit vectors $\hat{V}_i$ and $\hat{V}_j$ using L_2-norm, i.e.,

$$\hat{V}_i(k) = \frac{V_i(k)}{\sqrt{\sum_{n=1}^{N} V_i^2(n)}}, \; and \;\; \hat{V}_j(k) = \frac{V_j(k)}{\sqrt{\sum_{n=1}^{N} V_j^2(n)}}.$$

Then we calculate S_{ij} using the dot product of $\hat{V}_i$ and $\hat{V}_j$, i.e.,

$$S_{ij} = \sum_{k=1}^{N} \hat{V}_i(k) \times \hat{V}_j(k).$$

This is a *loose* definition of our *shared-influence-neighbor* (SIN) similarity. As discussed in previous subsection, all diagonal elements of the influence matrix have a value of 1 since no node is supposed to influence itself. We should exclude $V_i(i)$ and $V_j(j)$ in the computation of SIN similarity. Consequently, we do the normalization as follows

$$\hat{V}_i(k) = \frac{V_i(k)}{\sqrt{\sum_{n=1\ (n\neq i)}^{N} V_i^2(n)}}, \; k \neq i,$$

$$\hat{V}_j(k) = \frac{V_j(k)}{\sqrt{\sum_{n=1\ (n\neq j)}^{N} V_j^2(n)}}, \; k \neq j.$$

Then our *strict* definition of SIN similarity is defined as

$$S_{ij} = \hat{V}_i(j) \times \hat{V}_j(i) \; + \sum_{k=1 \; (k \neq i, \, k \neq j)}^{N} \hat{V}_i(k) \times \hat{V}_j(k).$$

This definition has a clear interpretation: for a pair of nodes i and j, their SIN similarity is measured by their mutual influence (first term) and the similarity in terms of the set of neighbors they both influence, which is computed as the cosine similarity of their influence vectors (second term). This *strict* definition is more accurate than the *soft* one even though the discrepancy might be negligible in most cases.

2.4 Community-Detection Algorithms

Now that we have the closeness measure for any pair of nodes in the network, a variety of well-studied clustering algorithms can be applied to find communities. To get preliminary insights, we first apply our (*loose*) SIN similarity to spherical K-means clustering [31], and develop an *influence-guided spherical K-means* (IGSK) algorithm [129]. Then we take full advantage of (*strict*) SIN similarity in label-propagation framework and arrive at two novel *influence-guided label-propagation* (IGLP) algorithms, *IGLP-Weighted-Ensemble* (IGLP-WE) and *IGLP-Direct-Passing* (IGLP-DP) [128].

2.4.1 Influence-Guided Spherical K-means

K-means clustering is a simple but popular unsupervised learning algorithm in data mining. Given a set of n points $(x_1, x_2, ..., x_n)$ in a d-dimensional space and a number K ($K \leq n$), it aims to partition the n points into K clusters ($C = \{C_1, C_2, ..., C_K\}$) so as to minimize the sum of *distance* of all points to the centroids of their respective clusters. In other words, it is to find C^* such that

$$C^* = \arg\min_{C} \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2,$$

where μ_i is the centroid of cluster i, and (squared) Euclidean distance is used. It is noted that the distance can be measured by any distance metrics and/or distance functions besides the commonly-used Euclidean distance.

As discussed in Section 2.2.3, our influence-diffusion model maps each node in a network to a point in an influence space, whose position is determined by the node's influence vector. The closeness of a pair of nodes can be measured by their SIN similarity. Thus it is natural to apply our SIN similarity to K-means clustering, which leads to our *influence-guided spherical K-means* (IGSK) algorithm. The objective of IGSK is to find the *optimal* community partition C^* that maximizes the sum of SIN similarity of all nodes to the centroids of their respective communities, i.e.,

$$C^* = \arg\max_{C} \sum_{i=1}^{K} \sum_{x \in C_i} \text{SIN-Similarity}(x, \mu_i).$$

IGSK algorithm starts with the generation of the influence vector of each node, do an initial assignment of K community centroids, and proceeds by alternating between two steps. One is *assignment* step, in which each node is assigned to its closest community according to its SIN similarity to each community centroid. The other is *update* step, in which the centroid of each community is recalculated with $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$. The algorithm stops when the assignments no longer change. Since the centroid is generally a virtual point in each community, the *strict* SIN similarity is not applicable in this case. We thus use the *loose* SIN similarity in IGSK. It is known that K-means clustering is sensitive to the initial selection of K centroids. Poor initialization may lead to slow convergence and/or poor results. In IGSK, we take advantage of our influence centrality in a heuristic way for better initialization. Intuitively, the most influential member of a community may have a higher probability

to be located in the center area of its community. Consequently, we first choose the node of the highest influence ranking as the centroid of community 1. Then for the next $(K-1)$ centroids, we choose the remaining node of the highest influence ranking, and assign it as a centroid of a community if and only if its SIN similarity with each of the already-selected centroids is less than a pre-specified *closeness threshold* ϕ. This mechanism significantly improves both the accuracy and efficiency. In addition, as a spherical K-means algorithm, we normalize the influence vector of each node and the vector of each community centroid into unit vectors in IGSK.

It is worth pointing out that in our influence-diffusion model we set the depth limit to 3 (by default) to capture the three-degrees-of-influence phenomenon. For small networks/communities or when the community structure is fuzzy, a depth limit of 2 may lead to better performance than a depth limit of 3. In practice, we run IGSK twice by setting the depth limit to 2 and 3, respectively, and finalize the community partition with the one that achieves higher modularity [97].

2.4.2 IGLP-Weighted-Ensemble

As discussed above, the traditional LPA has a significant stability issue due to random tie breaking and random ordering of nodes in the label-updating process. The key problem exists in its majority voting rule, which simply counts the number of neighbors of the same labels. It implicitly assumes that all neighbors have the same weight of 1 applied to their votes, i.e.,

$$c_i = \arg\max_l \sum_{j \in N_i^l} 1,$$

where c_i is the new label of node i and N_i^l is the set of neighbors of node i with label l. However, the similarity of a node to its different neighbors is not exactly the same. Our SIN similarity enables us to differentiate the structural proximity of a node to its neighbors in detailed manner. Therefore, we propose a straightforward extension to

the traditional LPA by weighting the vote of each neighbor using its SIN similarity to the node of interest. This algorithm is termed as *influence-guided label-propagation weighted ensemble* (IGLP-WE). With S_{ij} denoting the SIN similarity of nodes i to j, our *weighted majority voting* rule can be written as

$$c_i = \arg\max_l \sum_{j \in N_i^l} S_{ij}.$$

This simple weighting scheme not only directly resolves the random tie-breaking issue but also implicitly addresses the random node-ordering issue. Our experiments demonstrate that IGLP-WE always produces the same community partition regardless of the order of nodes in the label-updating process. Hence, we ignore the step of randomly choosing a node-updating order. Our IGLP-WE algorithm can be described in the following steps:

1. Generate the influence vector of each node (call IM-Builder());
2. For each node, calculate its SIN similarity to each of its neighbors;
3. Initialize the label of each node with its node index;
4. For each node, implement the asynchronous label updating using the *weighted* majority voting rule;
5. If no node changes its label, group the nodes into respective communities indicated by their labels and stop. Otherwise, go to Step 4.

2.4.3 IGLP-Direct-Passing

Given only the network graph topology, it is sensible to assume that *any node belongs to the same community as its closest (most similar) neighbor*. Inspired by this intuition, we leverage SIN similarity further to arrive at a new label-propagation framework, in which the community label is directly passed from a node to its most

similar neighbor iteratively. We refer to this algorithm as *influence-guided label-propagation direct passing* (IGLP-DP). At the end of this process, we are left with a set of most compact sub-communities. Each sub-community is composed of a group of most similar neighbors chained up from one to another.

The pseudocode of IGLP-DP is shown in Algorithm 2. Each node contains three integer variables: 1) *msn* denotes the node index of its most similar neighbor; 2) *label* denotes its community index; 3) *parent* denotes the node index of the node that passes the community label to the node of interest. The algorithm starts with the generation of the influence vector for each node based on the influence diffusion model (Line 2). Then it finds the most similar neighbor of each node using SIN similarity, and sets the community label of all nodes to 0 to indicate they are all unlabeled (Lines 3-6). In the following *For* loop, we perform label propagation (direct passing) for each node individually. For any node i (*root node*), we first check if it is unlabeled (Line 8). If yes, we assign its node index as its community label, and set its *parent* to be 0 to indicate its parent node is null (Line 9-10). We then assign its node index to *pIndex* and the node index of its most similar neighbor to *cIndex* (Lines 11-12). In the following *while* loop (Lines 13-18), we propagate *forward* the community label from *Node(pIndex)* to its most similar node *Node(cIndex)*, and denote *pIndex* as the node index of *Node(cIndex)*'s parent node. This forward propagation iterates until reaching a node that already has a community label, which is denoted as *newLabel* (Line 19). If *newLabel* is the same as the root node's label i (i.e., the label we are propagating forward), the label propagation of node i stops, and then we move on to the next node in the *For* loop. Otherwise, we propagate *backward newLabel* to *Node(pIndex)* and its parent node iteratively in a *while* loop until reaching the *root node* (Lines 20-25). After sweeping over all the nodes, IGLP-DP generates a set of sub-communities, in which each node acquires the same community label as its most similar neighbor.

Algorithm 2 IGLP-Direct-Passing

```
1: procedure IGLP-DP(G(V, E), n = |V|, depthLimit)
2:     Generate the influence vector for each node (call IM-Builder(G, n, depthLimit))
3:     for node i ← 1, n do
4:         Find Node(i).msn using the SIN similarity
5:         Node(i).label ← 0
6:     end for
7:     for node i ← 1, n do
8:         if Node(i).label = 0 then
9:             Node(i).label ← i
10:            Node(i).parent ← 0
11:            pIndex ← i
12:            cIndex ← Node(i).msn
13:            while Node(cIndex).label = 0 do
14:                Node(cIndex).label ← i
15:                Node(cIndex).parent ← pIndex
16:                pIndex ← cIndex
17:                cIndex ← Node(pIndex).msn
18:            end while
19:            newLabel ← Node(cIndex).label
20:            if newLabel ≠ i then
21:                while pIndex > 0 do
22:                    Node(pIndex).label ← newLabel
23:                    pIndex ← Node(pIndex).parent
24:                end while
25:            end if
26:        end if
27:    end for
28: end procedure
```

The backward propagation is essential to maintain the crucial property in which IGLP-DP is rooted, that is, *each* node should be in the same community as its most similar neighbor; meanwhile, it addresses the stability issue caused by node ordering in the label-updating process. IGLP-DP is easy to implement and highly efficient. It resolves the stability issue and always produces a deterministic partition. Another distinguishing feature is that it requires no convergent iteration.

2.4.4 Hierarchical Clustering

The resultant community partitioning from IGLP-WE and IGLP-DP provides an initial configuration of the community structure. Especially for IGLP-DP, the initial configuration contains many small, tight sub-communities. We use those communities in the initial configuration as building blocks for agglomerative hierarchical clustering to explore the community hierarchy at different scales.

The key issue here is how to define a cluster-proximity measure that is quantitatively accurate and computationally cheap. Since neighboring communities are connected by boundary nodes, it makes sense to measure cluster proximity using SIN similarity of the boundary nodes. More precisely, we focus on the out-link-based boundary nodes of each community. We refer to a node as an *out-link-based boundary node* if it has at least one out-link neighbor that resides in a different community. We measure the cluster proximity between a pair of neighboring communities using their repective out-link-based boundary nodes' SIN similarity. Considering that large communities tend to have more boundary nodes and more neighboring communities, we penalize the similarity by the number of community members and the number of neighboring communities to eliminate the bias on community size.

Let P_{ij} denote the cluster proximity between community i and community j, B_{ij} denote the set of boundary nodes in community i with out-link neighbors in community j, D_{kj} denote the set of node k's out-link neighbors in community j, S_{kl} denote the SIN similarity of nodes k to l, N_i denote the number of nodes in

community i, and C_i denote the number of neighboring communities that community i has. Similarly, let F_{ji} denote the set of boundary nodes in community j with out-link neighbors in community i, H_{mi} denote the set of node m's out-link neighbors in community i, S_{mn} denote the SIN similarity of nodes m to n, N_j denote the number of nodes in community j, and C_j denote the number of neighboring communities that community j has. We compute the cluster proximity P_{ij} as

$$P_{ij} = \frac{1}{N_i \times C_i} \sum_{k \in B_{ij}} \sum_{l \in D_{kj}} S_{kl} + \frac{1}{N_j \times C_j} \sum_{m \in F_{ji}} \sum_{n \in H_{mi}} S_{mn}.$$

A major weakness of traditional hierarchical clustering is its high computational cost. A straightforward implementation has time complexity $O(n^3)$, which makes it too slow to be scalable for large-scale datasets. We take advantage of our boundary-node-based cluster-proximity measure, and arrive at a novel and highly efficient hierarchical-clustering algorithm. A high-level description of this algorithm reads as follows:

1. For each cluster, sweep over each cluster member's out-link neighbors to find the boundary nodes and construct a neighboring-cluster list associated with the corresponding cluster proximity;

2. Sweep over each cluster's neighboring-cluster list to find the closest pair of clusters;

3. Merge the two closest clusters and relabel them;

4. For each cluster, sweep over each boundary node's out-link neighbors to reconstruct its neighboring-cluster list associated with the updated cluster proximity;

5. Repeat Steps 2 to 4 until only one cluster remains.

Our hierarchical clustering takes the initial community partition produced by the label propagation as input. This is a significant improvement in efficiency compared

to conventional agglomerative clustering that starts by assigning each node to a separate cluster, since the number of clusters in our initial community partition is much smaller than the number of nodes in the network. The efficiency is further improved by constructing a neighboring-cluster list for each cluster which greatly reduces the time complexity when searching for the closest pair of cluster in Step 2. Moreover, each node contains a *Boolean* variable that indicates whether it is a boundary node, which helps efficiently reconstruct the neighboring-cluster list and update the cluster proximity in Step 4. This is another considerable improvement in time complexity. It is noted that we incorporate our hierarchical-clustering algorithm as an integral part of the IGLP-WE and IGLP-DP algorithms. Hereafter whenever we refer to IGLP-WE or IGLP-DP, it means we perform the respective label propagation followed by the hierarchical clustering, and deliver a hierarchy of communities.

2.4.5 Overlapping Communities and Role Detection

Most complex networks exhibit overlapping community structures in which some nodes are characterized with multiple community memberships. In fact, the overlap is a significant feature of various social networks. However, finding overlapping communities or identifying overlapping nodes is another prominent challenge in community detection. Interestingly, our approach can be easily adapted to detection of overlapping communities/nodes. Using the influence vectors, we can naturally convert the non-overlapping community assignment produced by our IGSK, IGLP-WE or IGLP-DP, into a *fuzzy* overlapping assignment, in which each node is assigned to each community associated with a *belonging factor* that indicates the strength of the association of a node to a community.

Let N denote the total number of nodes in the network, K denote the total number of communities, V_i denote the influence vector of node i (without normalization), and C_j denote the set of nodes in community j (based on the non-overlapping assignment). The belonging factor of node i associated to community j is defined as

$$a_{ij} = \frac{\sum_{n \in C_j} V_i(n)}{\sum_{m=1\ (m \neq i)}^{N} V_i(m)}, \quad \forall i \in N,\ \forall j \in K.$$

This definition has a clear and natural interpretation: the belonging factor a_{ij} represents the ratio of the total influence node i spreads to community j to the total influence it spreads out in the network. Further, using a tunable *belonging threshold*, we can turn the fuzzy assignment into a *crisp* overlapping assignment (in which each node is associated to each community with a binary belonging factor) to identify overlapping nodes at different scales.

In addition, for each community (in the non-overlapping assignment), we can rank all of its community members by their *internal* influence, *external* influence, and *comprehensive* influence. For any node $i \in C_j$, the *internal* influence of node i is the total influence it spreads within its own community, i.e., $\sum_{n \in C_j} V_i(n)$. In contrast, the *external* influence of node i is the total influence it spreads to other communities, i.e., $\sum_{n \notin C_j} V_i(n)$. The *comprehensive* influence of node i is the sum of the internal and external influence. These three influence rankings enable us to identify the roles of individual members in each community.

2.5 Experiments

To get preliminary insights and verify the validity of our approach, we perform centrality analysis using two small real-life social networks and a large citation network, and then evaluate the performance of IGSK, IGLP-WE and IGLP-DP on community detection. We extensively test them on a large set of networks, which include six widely-used real-life networks and more than 500 LFR benchmarks [78]. For networks without ground truth, we adopt *modularity* [97] to evaluate the quality of a community partition. For networks with ground truth, we use *normalized mutual information* (NMI) [30] to evaluate the accuracy of a community partition. The definition of NMI is based on a confusion matrix $\mathbf{N}$, in which the rows correspond to

one partition A, the columns correspond to another partition B, and the (i, j)-entry $N_{i,j}$ is the number of nodes that appear in both community i of partition A and community j of partition B. NMI measures the similarity between the two partitions as follows

$$\mathrm{NMI}(A, B) = \frac{-2 \sum_{i=1}^{C_A} \sum_{j=1}^{C_B} N_{ij} \log(\frac{N_{ij} N}{N_{i.} N_{.j}})}{\sum_{i=1}^{C_A} N_{i.} \log(\frac{N_{i.}}{N}) + \sum_{j=1}^{C_B} N_{.j} \log(\frac{N_{.j}}{N})},$$

where C_A and C_B denote the number of communities in partitions A and B respectively, $N_{i.}$ denotes the sum over row i of matrix $\mathbf{N}$ (i.e., the number of nodes in community i of partition A), $N_{.j}$ denotes the sum over column j of matrix $\mathbf{N}$ (i.e., the number of nodes in community j of partition B), and N is the total number of nodes in the network. NMI is a well-defined and widely-used evaluation criterion for community detection. NMI gets its maximum value of 1 if two partitions are identical, whereas it equals 0 if the partitions are independent. For example, when one partition fails to find any communities but simply clusters the entire network into one community, we get an NMI score of 0.

2.5.1 Network Description

The three real-life networks used for centrality analysis are *Zachary's Karate Club* [141], *Sawmill communication* [92], and an *arXiv HEP-TH citation network* [45]. The six real-life networks used for community detection are *Zachary's Karate Club*, *Dolphin Social Network* [89], *Political Books* [101], *American College Football* [47], *Email* [53], and *PGP Network* [10]. All of them are widely-used benchmark networks. We list in Table 2.2 their basic information, including the number of ground-truth communities for networks with ground truth.

Since large real-life networks with reliable ground truth are rarely available (especially for directed and/or weighted networks), we further test our algorithms on LFR benchmarks. To compare with the algorithms examined in [78], we generate a set of

Table 2.2: Real-life networks

Networks	**Nodes**	**Edges**	**Communities**
Karate	34	78	2
Sawmill	36	62	3
Dolphins	62	159	2
PolBooks	105	441	3
Football	115	613	12
Email	1133	5451	—
PGP	10,680	24,316	—
HEP-TH	27,771	352,807	—

LFR benchmark graphs using the same parameter settings: *average degree* = 20, *maximum degree* = 50, *degree-distribution exponent* = -2, *community-size-distribution exponent* = -1. There are two different network sizes (1000 and 5000 nodes), and two different ranges for community size (S and B). "S" stands for "small", which means *min/max community size* = 10/50. In contrast, "B" stands for "big", which means *min/max community size* = 20/100. Another important parameter is the *topological mixing parameter* μ_t ($0 < \mu_t < 1$) [80]. Let k_i denote the degree of node i. Then node i's expected internal degree and external degree are $k_i^{(int)} = (1 - \mu_t)k_i$ and $k_i^{(ext)} = \mu_t k_i$, respectively. The smaller μ_t, the clearer community structure. Specifically, when μ_t is less than 0.5, each node has more internal connections than external connection, which implies all communities are strong communities. We generate eight sets of unweighted benchmark networks (four undirected and four directed), in which we vary μ_t from 0.1 to 0.8 at an increase of 0.1.

For weighted networks, there are two other parameters, *weight-strength-distribution exponent* β, and *weight mixing parameter* μ_w. The parameter β specifies the weight strength s_i with $s_i = k_i^{\beta}$, which is the sum of weights of all links of node i. The parameter μ_w is used to specify the expected internal weight strength $s_i^{(int)}$ and external weight strength $s_i^{(ext)}$, with $s_i^{(int)} = (1 - \mu_w)s_i$ and $s_i^{(ext)} = \mu_w s_i$. Generally speaking,

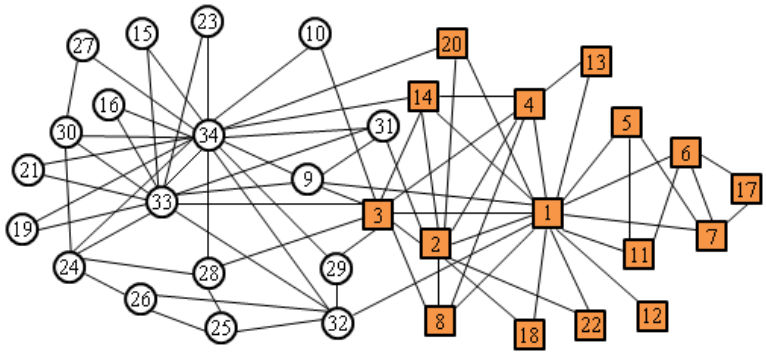

Figure 2.5: Zachary's karate club

the smaller μ_w, the more reinforcement of the weight distribution on the community structure. We generate four sets of LFR weighted networks with $\beta = 1.5$. As done in [78], we fix μ_t to 0.5 and 0.8 respectively, and vary μ_w from 0.1 to 0.8 at an increase of 0.1. We generate five realizations for each value of the topological/weight mixing parameters and average the results in each experiment.

2.5.2 Centrality Analysis

Figure 2.5 is a graphical illustration of the well-known Zachary's karate club network. The 34 club members split into two groups due to the disagreement between the club instructor (node 1) and the club president (node 34). The orange squares represent members associated with the instructor, and the white circles represent members in the president's group.

We show in Figure 2.6 its global influence ranking in terms of our *influence centrality* scores ($depthLimit = 2$). We use a rating scale of 0 to 10, with 10 meaning "most influential". As we can see, our *influence centrality* gives the two leaders (nodes 1 and 34) the highest scores, and finds a set of core members of the club (nodes 2, 3, 4, 9, 14, 32 and 33). One may argue it is expected that they get higher scores simply because of their higher degrees. In fact, it is not that straightforward. For example, although node 4 has a higher degree than node 14, its score is actually lower than

that of node 14. One may also notice that nodes 10 and 17 both have a degree of 2, but node 10 has a much higher score than node 17. It makes sense since node 10 connects to nodes 3 and 34, which are much more influential than nodes 6 and 7 to which node 17 connects. Moreover, even though node 12 only has a degree of 1, it also gets a score greater than node 17 because node 12 has a direct connection to the group leader (node 1). It follows our intuition that connecting to a more influential person contributes more influence to the person of interest than connecting to a less influential one. Our *influence centrality* unveils the connectivity-based influence significance in a more detailed manner.

We also perform centrality analysis on the sawmill communication network. Figure 2.7 illustrates its network graph and ground-truth communities indicated with different node colors/shapes. We list in Table 2.3 our *influence* centrality ($depthLimit = 3$; referred to as "Influence" in the table) and compare it against a set of conventional centralities, where "CFC" and "CFB" stand for *current-flow-closeness* and *current-flow-betweenness* centralities [14], respectively.

As we can see, *PageRank* simply degenerates into *degree* centrality as expected since this network is undirected. *Closeness* and *betweenness* centralities are not quantitatively fine or comprehensive enough to differentiate the overall influence ranking. The *closeness* centrality scores of 10 versus 4.12 do not show the expected large difference in influence significance of HM-1 versus HP-1 as compared to our *influence* centrality scores of 10 versus 0.62. *Betweenness* centrality fails to measure the influence significance of nine employees by giving them a score of zero. These include nodes 15 and 22, who are actually the immediate neighbors of the most influential employee, node 12. The drawbacks of *closeness/betweenness* centralities are inherent in their definitions since they only focus on the shortest paths and fail to incorporate the neighboring nodes' influence significance or the attenuation of influence along diffusion paths. *Current-flow closeness/betweenness* centralities do not show much

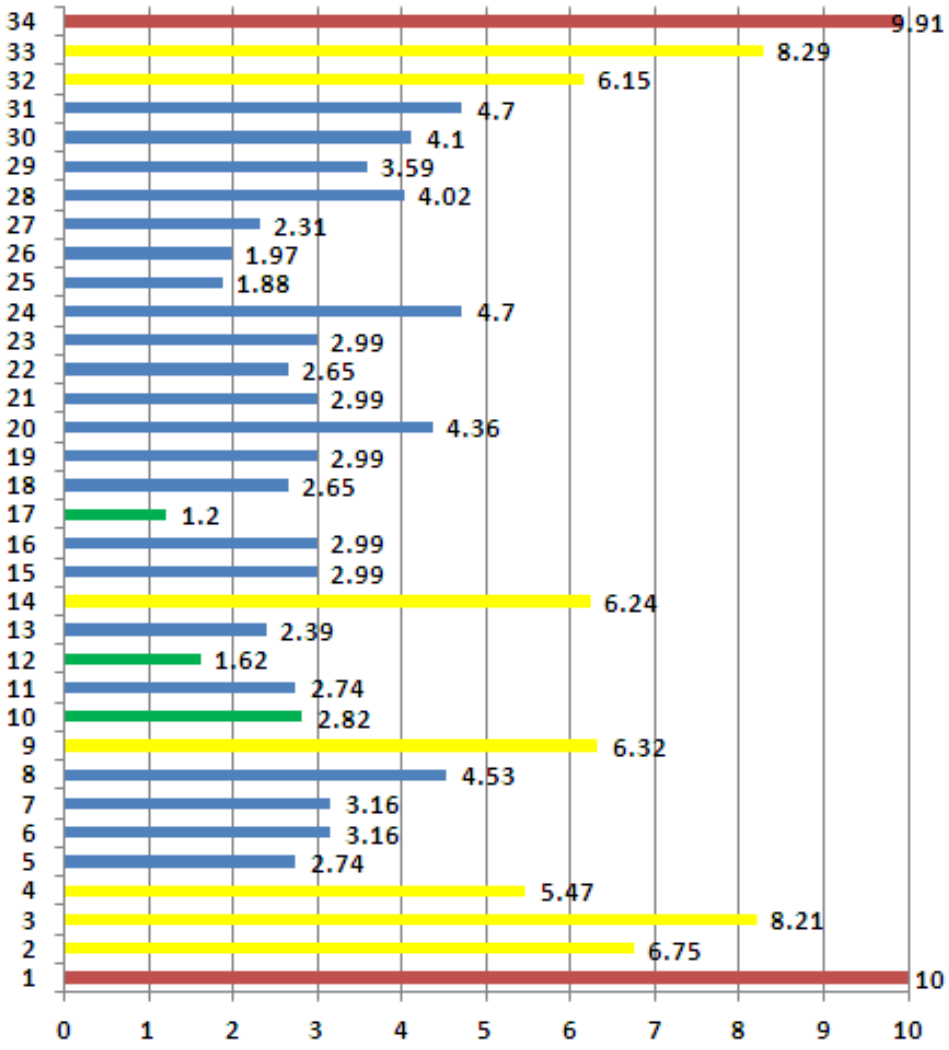

Figure 2.6: Influence ranking of Zachary's karate club

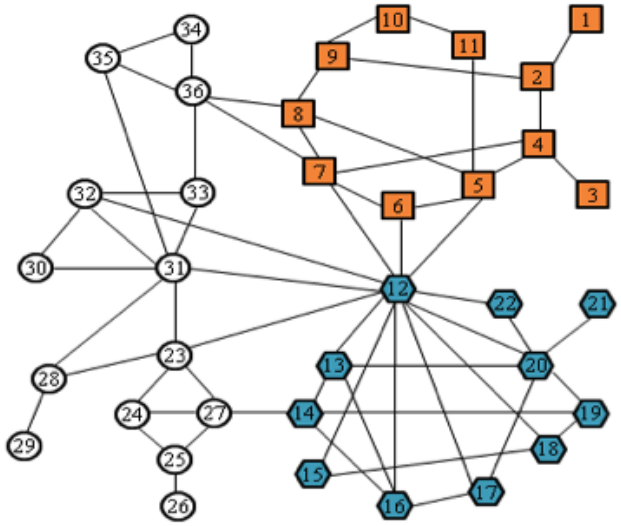

Figure 2.7: Sawmill communication network

improvement from the conventional ones. Only *eigenvector* centrality exhibits a similar ranking pattern to our *influence* centrality. We remark that this comparison is not to prove the failure of other centralities. As discussed in Section 2.2.1, each centrality provides a different perspective on some specific strength of a node's structural role in the network. What we want to show is that our *influence* centrality gives rise to a novel centrality measure that differentiates more precisely the nodes' *comprehensive* significance on reachability-based influence diffusion.

We examine the validity of our *influence* centrality in directed networks using the arXiv HEP-TH citation network, which consists of 27,771 papers and 352,807 citations among them. Those papers are in the field of high energy physics, and were added to the e-print arXiv between 1992-2003. The first four digits of each paper ID represent the year and the month when the paper was published online. For instance, paper *9510017* indicates it was published in October of 1995. Table 2.4 lists the top 10 papers identified by our *influence* centrality ($depthLimit = 3$), *in-degree* centrality, and *PageRank*, respectively. For each paper, we indicate in parenthesis the number of citations it receives, i.e., its in-degree. As discussed in Section 2.3, when computing

Table 2.3: Comparison of different centralities on sawmill communication network. CFC stands for *current-flow-closeness*, and CFB for *current-flow-betweenness*

Employee	Node	Influence	Eigenvector	Degree	PageRank	Closeness	CFC	Betweenness	CFB
HP-1	1	0.62	0.09	0.77	0.79	4.12	3.25	0.00	0.00
HP-2	2	1.69	0.47	2.31	2.37	5.19	5.27	1.05	1.80
HP-3	3	0.87	0.34	0.77	0.79	4.93	3.74	0.00	0.00
HP-4	4	3.35	1.68	3.08	3.13	6.54	6.70	2.47	2.72
HP-5	5	5.36	3.63	3.85	3.90	8.19	8.01	3.37	3.73
HP-6	6	4.95	3.52	2.31	2.33	7.56	7.32	0.01	1.65
HP-7	7	5.76	3.90	3.85	3.89	8.29	8.28	2.77	3.57
HP-8	8	3.92	2.05	3.08	3.13	6.67	7.47	1.22	2.71
HP-9	9	1.90	0.56	2.31	2.36	5.35	5.83	0.45	2.07
HP-10	10	1.14	0.27	1.54	1.58	5.00	4.74	0.06	0.89
HP-11	11	1.73	0.78	1.54	1.57	5.96	5.12	0.68	1.01
HM-1	12	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
HM-2	13	5.41	4.32	3.08	3.06	7.31	7.53	0.37	1.48
HM-3	14	3.73	2.45	3.08	3.06	6.13	7.33	0.53	2.54
HM-4	15	3.38	2.60	1.54	1.53	6.80	5.93	0.00	0.49
HM-5	16	5.29	4.14	3.08	3.06	7.23	7.47	0.34	1.41
HM-6	17	4.78	3.82	2.31	2.30	7.01	7.00	0.02	0.96
HM-7	18	3.93	2.95	2.31	2.30	6.94	6.69	0.34	1.18
HM-8	19	3.01	2.07	2.31	2.29	5.91	6.71	0.18	1.45
HM-9	20	5.99	4.86	4.62	4.60	7.23	7.90	1.38	2.73
HM-10	21	1.36	0.98	0.77	0.76	5.31	4.09	0.00	0.00
HM-11	22	3.78	2.99	1.54	1.53	6.87	6.16	0.00	0.59
EM-1	23	5.77	4.02	3.85	3.84	8.10	7.96	3.59	4.11
EM-2	24	2.63	1.27	2.31	2.30	6.13	5.88	0.74	1.44
EM-3	25	1.76	0.62	2.31	2.30	4.93	4.99	0.96	1.30
EM-4	26	0.62	0.12	0.77	0.77	3.95	3.14	0.00	0.00
EM-5	27	3.19	1.68	3.08	3.07	6.36	6.68	1.29	2.71
Y-1	28	3.18	1.91	2.31	2.30	6.24	6.13	0.96	1.80
Y-2	29	0.79	0.39	0.77	0.77	4.76	3.56	0.00	0.00
Forester	30	2.74	1.80	1.54	1.54	5.86	5.80	0.00	0.38
Manager	31	6.68	5.11	5.38	5.39	8.19	8.60	2.84	4.79
Owner	32	5.16	3.85	3.08	3.08	7.39	7.47	0.51	1.81
Operator	33	3.61	2.23	2.31	2.32	6.60	6.94	0.22	1.62
EP-1	34	1.70	0.75	1.54	1.55	5.48	5.37	0.00	0.57
EP-2	35	2.95	1.61	2.31	2.32	6.60	6.47	0.57	1.64
EP-3	36	3.91	2.12	3.85	3.88	6.80	7.68	1.15	3.51

our *influence* centrality, we reverse the link direction of the citation network to reflect the *real* influence diffusion from cited papers to citing papers. It is noted that the *in-degree* mentioned above refers to the original citation network.

Like most centrality measures, our *influence* centrality is correlated with degree centrality. All the top-10 *influence-centrality* papers have high in-degrees. It includes 7 of the top-10 *in-degree centrality* papers but ranks them in different order. It is hard to rigorously prove our *influence* centrality gives the exact significance ranking

Table 2.4: Centralities of the HEP-TH network

Rank	**Influence**	**In-degree**	**PageRank**
1	9510017 (1155)	9711200 (2414)	9402044 (257)
2	9503124 (1144)	9802150 (1775)	9205068 (167)
3	9711200 (2414)	9802109 (1641)	9205027 (191)
4	9410167 (748)	9407087 (1299)	9207053 (102)
5	9510135 (775)	9610043 (1199)	208020 (205)
6	9802150 (1775)	9510017 (1155)	9204102 (71)
7	9802109 (1641)	9908142 (1144)	9301042 (344)
8	9610043 (1199)	9503124 (1114)	9201019 (16)
9	9407087 (1299)	9906064 (1032)	9205081 (77)
10	9601029 (651)	9408099 (1006)	9209016 (76)

of those papers. We believe it differentiates the influence ranking more precisely than *in-degree* centrality. *Degree* centrality is actually a special case of our *influence* centrality, i.e., setting *depthLimit* to 1. Specifically, for this citation network, it simply counts the number of in-link neighbors of each node. When we set *depthLimit* to 3 as we do by default, we explore the whole 3-step neighborhood of each node, and incorporate the influence significance of its neighbors and its neighbors' neighbors. It is observed that *PageRank* fails to rank the influence significance in this case. All the top-10 *PageRank* papers have very low in-degrees (citations) compared to the top-10 *influence centrality* and *in-degree centrality* papers. They receive such high rankings simply because they are old papers that do not have any out-links (the papers they cited are too old to be included in the dataset).

2.5.3 Community-Detection Performance

We evaluate the performance of our IGSK, IGLP-WE and IGLP-DP algorithms on both real-life networks and synthetic benchmarks using modularity and NMI scores. In particular, for IGLP-WE and IGLP-DP, the modularity score we present is the maximum modularity we find in the resultant community hierarchy; for networks with

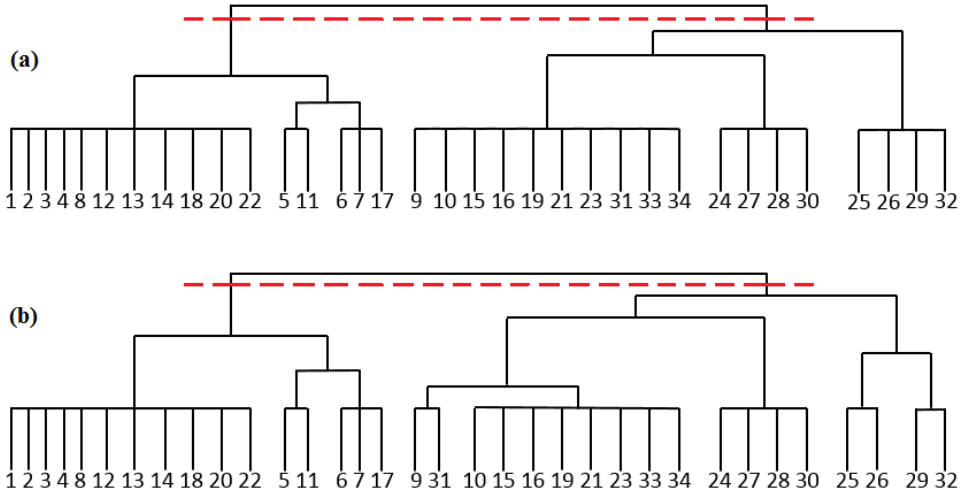

Figure 2.8: Dendrograms generated by: (a) IGLP-WE; (b) IGLP-DP

ground truth, we split the hierarchy dendrogram at the level at which the number of separated communities is the same number of communities in the ground truth, and then compute the NMI score.

Real-life networks. For the most commonly-used Zachary's karate club network as shown in Figure 2.5, our IGSK, IGLP-WE and IGLP-DP all exactly find the two ground-truth communities. We illustrate in Figure 2.8 its community dendrograms IGLP-WE and IGLP-DP generate. As we can see, IGLP-WE produces an initial configuration of six sub-communities of different sizes, and IGLP-DP delivers eight sub-communities by splitting two sub-communities detected by IGLP-WE into four smaller ones. IGLP-DP always produces more smaller and tighter sub-communities in the initial community-structure configuration than IGLP-WE. Both IGLP-WE and IGLP-DP find communities that match the ground truth perfectly when we split their respective hierarchy dendrogram at the level of two separate communities.

We compare the performance of IGSK, IGLP-WE and IGLP-DP against eight representative algorithms examined in the literature [79, 88, 137]. Their NMI and modularity scores are listed in Table 2.5. As we can see, IGSK, IGLP-WE and IGLP-

DP all demonstrate excellent performance. They clearly outperform *Fitness*, *CPM*, *Fastgreedy*, and *LPA* in terms of higher NMI scores on all networks with ground truth. All of them achieve perfect NMI score on *KarateClub* and top NMI scores on *Dolphins*. Moreover, when using modularity as the metric, IGLP-WE and IGLP-DP achieve superior performance. The only exception is the score of IGLP-WE on *Email*. It groups all the nodes in one community in its initial configuration. This is a common issue of most LPAs. As discussed before, what the traditional LPA finds is *strong* communities, in which each node has higher internal connections than external connections. When the community structure is weak, LPA arrives at one or several *monster* communities. IGLP-WE can find both strong and weak communities since it weights the votes with SIN similarity instead of simply counting them. But when the community structure is sufficiently fuzzy, IGLP-WE encounters the same problem. IGLP-DP exhibits great advantages in this regard. It enables us to zoom into smaller, tighter sub-communities and build a community hierarchy with finer granularity. IGLP-DP achieves higher modularity scores on both *Email* and *PGP* networks than *Fastgreedy*, which is a representative algorithm that aggressively maximizes modularity. IGLP-DP shows the best performance overall.

Table 2.5: Performance comparison on real-life networks

	NMI				**Modularity**					
Algorithm	Karate	Dolphins	PolBooks	Football	Karate	Dolphins	PolBooks	Football	Email	PGP
Fitness [79]	0.690	0.781	—	0.754	—	—	—	—	—	—
CPM [104]	0.170	0.254	—	0.697	—	—	—	—	—	—
DCBNT [88]	**1.000**	0.762	0.578	0.949	—	—	—	—	0.537	0.819
Infomap [113]	0.700	0.537	0.494	**0.972**	—	—	—	—	0.526	0.801
Fastgreedy [27]	0.693	0.557	0.531	0.753	—	—	—	—	0.507	0.853
Walktrap [106]	0.504	0.582	0.543	0.937	—	—	—	—	0.531	0.789
LPA [108]	0.583	0.516	0.572	0.863	0.296	0.465	0.489	0.582	0.38	0.806
NIBLPA [137]	**1.000**	0.622	**0.656**	0.872	0.423	0.521	0.497	0.582	0.427	0.783
IGSK	**1.000**	0.814	0.541	0.924	0.421	0.406	0.513	0.609	—	—
IGLP-WE	**1.000**	**0.889**	0.482	0.927	**0.450**	**0.546**	0.463	**0.613**	0.002	**0.868**
IGLP-DP	**1.000**	**0.889**	0.576	0.918	**0.450**	0.528	**0.521**	0.612	**0.547**	0.863

It is worth pointing out that: for some algorithms (like *Fitness* and *NIBLPA*), one has to tune a user-specified parameter to find the best result for *each* network individually. The *depth limit* is the only tunable parameter in our algorithms. For IGSK, we run it twice by setting the depth limit to 2 and 3, respectively. For IGLP-WE and IGLP-DP, the depth limit is set to 3 for *all* networks. In fact, it can be regarded as a fixed, built-in parameter in IGLP-WE and IGLP-DP, which matches the well-known *three-degrees-of-influence* phenomenon [24].

Undirected unweighted LFR benchmarks. We compare the performance of our algorithms against eight state-of-the-art algorithms examined in [78]. They are referred to as: *Blondel et al.* [9], *MCL* [124], *Infomod* [112], *Infomap* [113], *Cfinder* [104], *Clauset et al.* [27], *Radicchi et al.* [107], and *Sim. ann.* [52]. Note that *Blondel et al.* is exactly the *Louvain method* discussed in Section 2.2.5.

We illustrate in Figure 2.9 the results on the four sets of undirected unweighted LFR benchmarks, in which each curve shows the variation of the averaged NMI score with respect to the topological mixing parameter μ_t. It is shown that IGSK, IGLP-WE and IGLP-DP consistently exhibit excellent performance. Even when μ_t is set to 0.5 (the threshold of defining strong communities), IGSK gets NMI scores of 0.999, 0.992, 0.968 and 0.99 for 1000-S, 1000-B, 5000-S and 5000-B datasets, respectively. IGLP-WE and IGLP-DP achieve almost perfect NMI scores on all datasets when μ_t is less than or equal to 0.5. IGLP-WE maintains its superior performance on 5000-S datasets with μ_t up to 0.7. IGLP-DP shows even better performance than IGLP-WE overall. It is observed that they all perform better on larger networks and smaller communities. Compared against the eight algorithms, IGSK, IGLP-WE and IGLP-DP clearly outperform seven of them except *Infomap*, which achieves higher NMI scores when μ_t equals 0.6 or 0.7. In fact, *Infomap* is one of the best algorithms in the literature. As shown in Table 2.5, our algorithms demonstrate better performance than *Infomap* on real-life networks overall.

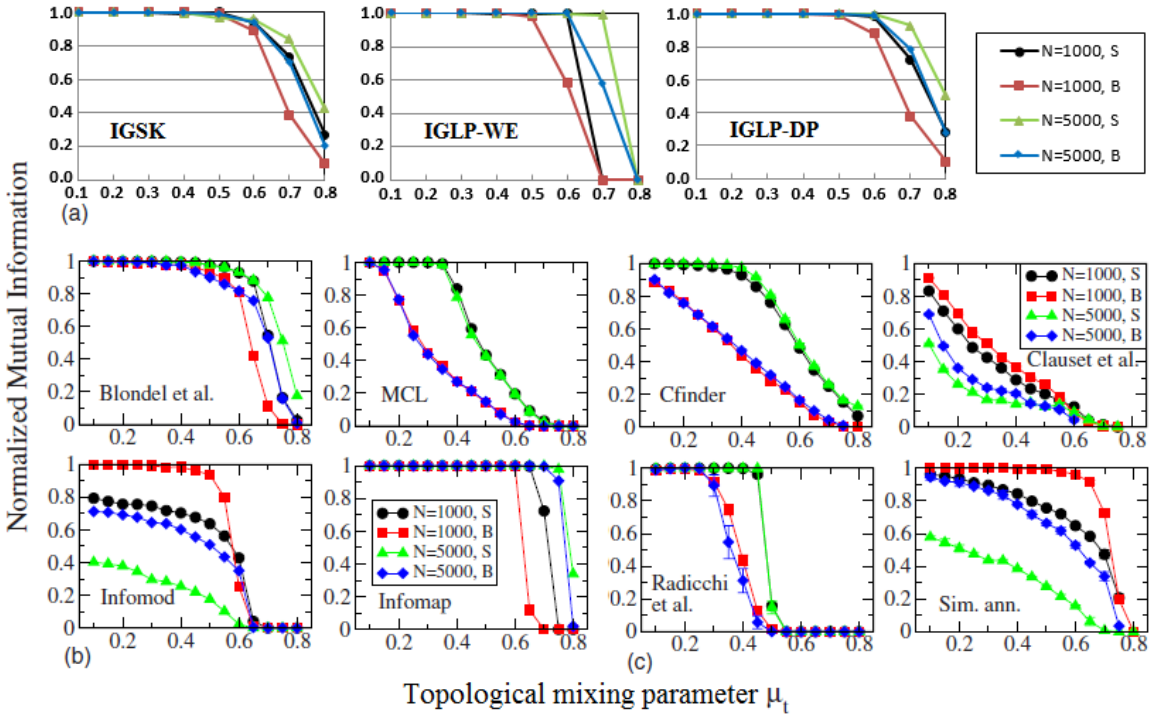

Figure 2.9: Performance comparison on undirected unweighted LFR benchmarks. Plots (b) and (c) are from [78], Copyright by The American Physics Society

Directed unweighted LFR benchmarks. Community detection in directed networks is more challenging. Most existing algorithms are not able to find communities in directed networks, and extension of an algorithm to directed networks is usually nontrivial. For performance evaluation, we generate four sets of directed unweighted LFR benchmark graphs using the parameters we discussed in Section 2.5.1. It is noted that both the *degree-distribution exponent* and the *topological mixing parameter* μ_t refer to the in-degree while the out-degree is kept constant for all nodes. This setting makes the resultant networks similar to citation networks in terms of their in-degree and out-degree distributions. Therefore, as we did with the arXiv HEP-TH citation network, we reverse the link direction of these directed LFR benchmark graphs to reflect the influence flow that fits in our influence-diffusion model.

We illustrate the results of IGSK, IGLP-WE and IGLP-DP in Figure 2.10 and

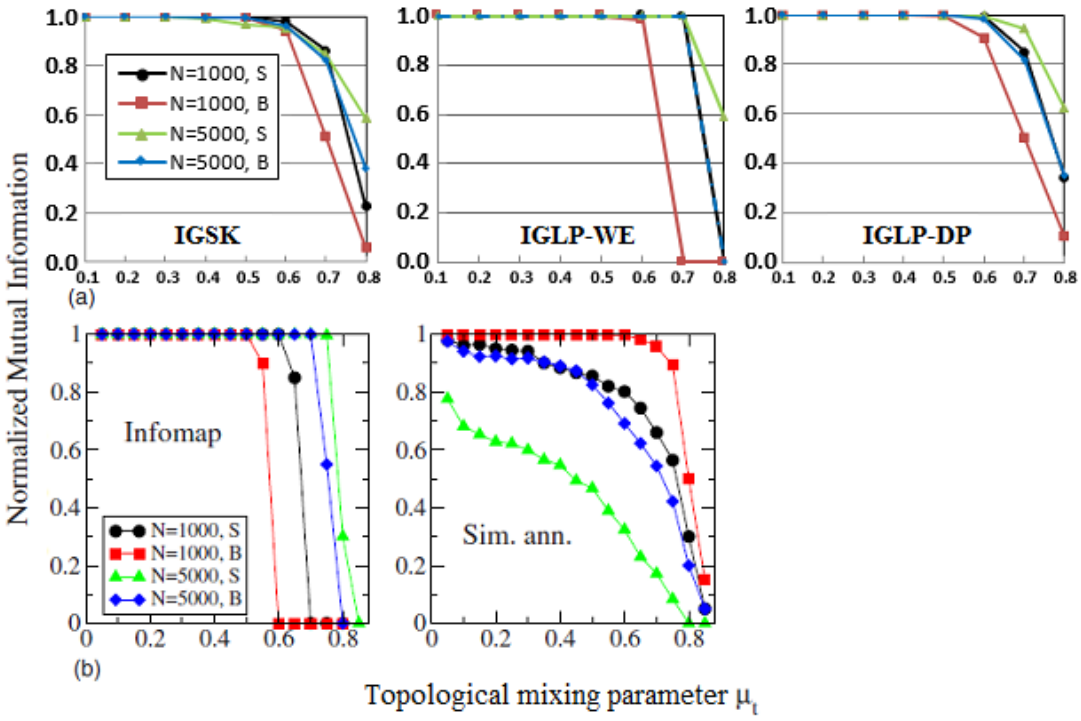

Figure 2.10: Performance comparison on directed unweighted LFR benchmarks. Plot (b) is from [78], Copyright by The American Physics Society

compare them against *Infomap* and *Sim. ann.* Once again, IGSK, IGLP-WE and IGLP-DP demonstrate outstanding performance in directed networks, which is even better than their performance in undirected networks. IGSK achieves very high NMI scores with μ_t up to 0.6, which are 0.978, 0.941, 0.954 and 0.958 for 1000-S, 1000-B, 5000-S and 5000-B datasets, respectively. IGLP-WE and IGLP-DP exhibit superior performance. IGLP-WE gets perfect NMI scores on 1000-S and 5000-S/B datasets with μ_t up to 0.7. Both IGLP-WE and IGLP-DP outperform *Infomap* on 1000-S/B datasets, and clearly beat *Sim. ann.* on 1000-S and 5000-S/B datasets. It is noted that IGLP-WE and IGLP-DP exhibit better performance on larger networks and smaller communities in directed networks as they do in undirected networks.

Undirected weighted LFR benchmarks. We fix the network size to 5000 nodes and set the *topological mixing parameter* μ_t to 0.5 and 0.8, respectively. Combining

with the two different ranges for community size (S and B), we generate four sets of undirected weighted LFR benchmark graphs, denoted as 5000-S-0.5, 5000-S-0.8, 5000-B-0.5 and 5000-B-0.8, respectively. For each dataset, we vary the *weight mixing parameter* μ_w from 0.1 to 0.8 at an increase of 0.1. Let us first take a look at the distribution of the weights as described in [77]. For a node of degree k_i, its expected *internal* weight and *external* weight can be expressed as

$$w_i^{(int)} = \frac{1-\mu_w}{1-\mu_t} k_i^{\beta-1}, \ and \ \ w_i^{(ext)} = \frac{\mu_w}{\mu_t} k_i^{\beta-1}.$$

Then the ratio of *internal* weight to *external* weight (referred to as *int/ext ratio*) is related to the two mixing parameters μ_t and μ_w in a simple way:

$$int/ext\ ratio = \frac{w_i^{(int)}}{w_i^{(ext)}} = \frac{\mu_t(1-\mu_w)}{\mu_w(1-\mu_t)}.$$

To verify our weight normalization scheme and better understand how the weight plays its role in shaping the community structure with respect to the network topology, we run IGSK on each weighted network twice. One ignores the weights, denoted as *IGSK-ignore*; the other considers the weights, denotes as *IGSK-consider*. The results are illustrated in the first two plots in Figure 2.11. As we can see, for 5000-S-0.8 and 5000-B-0.8 datasets, μ_t is set to 0.8, which indicates their community structure is fuzzy. While μ_w varies from 0.1 to 0.7, the *int/ext ratio* decreases from 36 to 1.7 but is always greater than 1. It implies that the weight distribution always reinforces the community structure in these cases, even though the reinforcement decays while μ_t increases. *IGSK-ignore* gives low NMI scores as expected since it completely ignores the useful weight information. In contrast, *IGSK-consider* takes advantage of the weight information and greatly improve the performance. Further, it reflects a sensible pattern: the higher *int/ext ratio* of the weight, the stronger reinforcement of the community structure, the greater performance improvement *IGSK-consider* achieves.

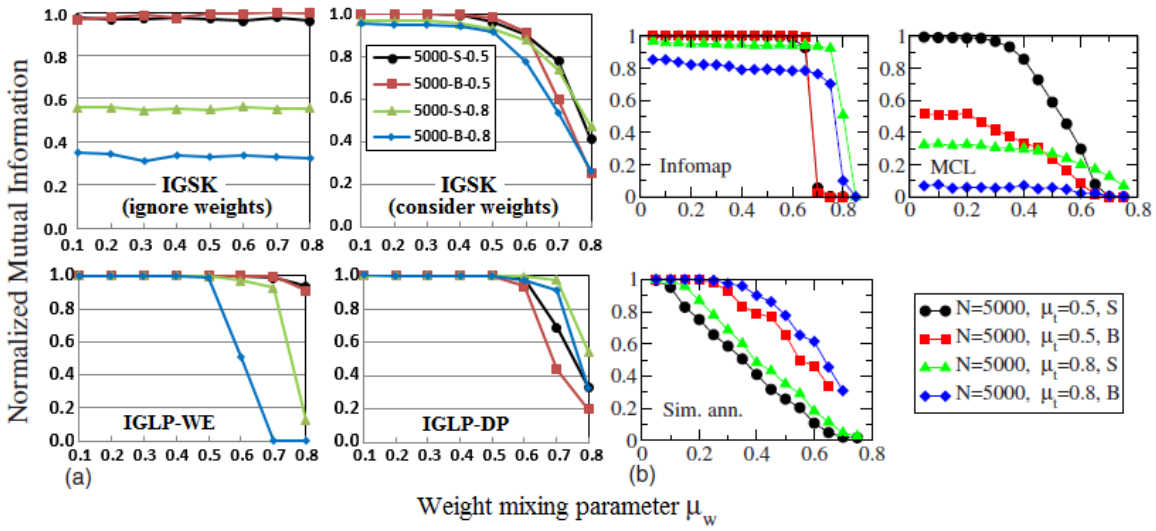

Figure 2.11: Performance comparison on undirected weighted LFR benchmarks. Plot (b) is from [78], Copyright by The American Physics Society

For 5000-S-0.5 and 5000-B-0.5 datasets, μ_t is 0.5, which indicates the community structure is relatively clear topologically. In this case, when μ_w falls in the range from 0.1 to 0.4, the weight distribution confirms the community structure since the corresponding *int/ext ratio* is greater than 1. However, when μ_w is greater than 0.5, the *int/ext ratio* becomes smaller than 1, which implies the weight distribution undermines the community structure. Our experimental results provide strong evidence of the above argument. On one hand, *IGSK-ignore* shows excellent performance consistently from $\mu_w = 0.1$ to 0.8, which is expected as IGSK does on unweighted networks when $\mu_t = 0.5$. On the other hand, *IGSK-consider* gives perfect NMI scores when $\mu_w < 0.5$, which outperforms *IGSK-ignore* by taking into account the weight information that reinforces the community structure. However, its performance worsens dramatically due to the misleading weight information when $\mu_w > 0.5$.

The tests on the four sets of weighted networks demonstrates that our IGSK algorithm effectively captures both network connectivity and weight information. However, for networks without ground truth, it is hard to tell whether the weight

information strengthens or undermines the community structure. In practice, we can run both *IGSK-consider* and *IGSK-ignore* and take the one with higher modularity. In addition, our experiment brings forth an important point that: *community structure is primarily determined by the network topology, and the weight information is a secondary factor that may reinforce or undermine the community structure.*

For IGLP-WE and IGLP-DP, the weights on edges are always considered. We illustrate their results in Figure 2.11 and perform comparison against IGSK and the three algorithms examined in [78]. Interestingly, while IGLP-WE and IGLP-DP consistently show excellent performance and maintain their preference on smaller community size, they show different performance on the topological mixing parameter. While IGLP-WE works better on 5000-S/B-0.5 datasets, IGLP-DP performs better on 5000-S/B-0.8 datasets. As discussed above, when μ_t is 0.8 (weak community), the weight distribution reinforces the community structure while μ_w is less than 0.8. However, if μ_t is set to 0.5 (threshold of strong community), the weight distribution undermines the community structure when $\mu_w > 0.5$. Therefore, we can infer that IGLP-DP is able to exploit more weight information than IGLP-WE, and so is more sensitive to the weight distribution. Except for IGLP-WE on 5000-B-0.8 dataset, IGLP-DP and IGLP-WE clearly outperform all other algorithms in the comparison, including IGSK and *infomap*.

Our extensive tests on both real-life networks and LFR benchmarks verify the validity of our approach. IGSK, IGLP-WE and IGLP-DP show superior performance consistently on undirected/directed and unweighted/weighted networks. They outperform a large set of state-of-the-art algorithms in the comparison, and IGLP-DP is the best performing algorithms overall. Moreover, unlike most existing algorithms in the literature that deliver a single community partition, IGLP-WE and IGLP-DP reveal a complete community hierarchy, which enables us to examine the communities at different levels of granularity.

Space complexity. As indicated in Section 2.3.1, each influence vector is stored in a compact dynamic array. The space complexity is $O(Ln)$ for IGSK, IGLP-WE and IGLP-DP, where n is the number of nodes in the network, and L is the average length of influence vectors. L is determined by the average node out-degree b, depth limit d_{max} and and the community structure. Generally, the more cohesive community structure, the shorter influence vectors. IGSK needs to keep all influence vectors until the end of the algorithm, and thus it is hard to improve its space complexity. For IGLP-WE and IGLP-DP, however, the space complexity can be greatly improved. Instead of simply following the node-index order to generate the influence vector for each node, we can implement a *breath-first* search algorithm, in which we generate the influence vector of a node followed by generating the influence vectors of its neighbors. Then we can calculate its SIN similarity to each of its neighbors. Once this is done, it is no need to keep the influence vector of that node. We can delete it and reclaim the space immediately.

Time complexity. It is hard to rigorously estimate the time complexity since it is closely related to the community structure and the average length of influence vectors. For IGLP-WE and IGLP-DP, it is also related to the number of sub-communities in the initial community configuration delivered by the label-propagation process.

Let n denote the total number of nodes in the network, K denote the number of communities to be found, L denote the average length of influence vectors, and I denote the number of iterations to converge. IGSK has a time complexity of $O(KILn)$. IGSK converges fast when the community structure is clear. For example, for almost all LFR benchmarks in our experiments, IGSK converges after two iterations when μ_t is 0.3 or less. When the community structure is fuzzy, however, it may take 10 iterations or more. We force it to stop if it does not converge after eight iterations. Although IGSK is fairly efficient, it does not scale well on large-scale networks, especially when the community structure is not clear.

IGLP-WE and IGLP-DP are much faster than IGSK. To examine their time complexity, we experiment on a set of undirected and unweighted LFR benchmarks with $\mu_t = 0.5$ and *min/max community size* = 20/100 (all other parameters are the same as described before). We vary the number of nodes n from 2,500 to 25,000 in an increment of 2,500, and generate five realizations for each value of n. All the experiments are carried out on a regular desktop PC with Intel(R) Core(TM) i5-4670 CPU @ 3.40 GHz and 8.0 GB memory under Windows 7 64-bit OS. Let m denote the number of edges in the network, K_{gt} denote the number of communities of ground truth, and K_{we} and K_{dp} denote the number of sub-communities in the initial community configuration given by IGLP-WE and IGLP-DP, respectively. We also include their respective NMI scores denoted by NMI_{we} and NMI_{dp}. The experimental results are shown in Table 2.6 and Figure 2.12, in which we present not only the total time but also the time spent on generating the influence vectors, label propagation and hierarchical clustering, respectively.

Table 2.6: A set of LFR benchmarks

n	m	K_{gt}	K_{we}	K_{dp}	$\mathbf{NMI}_{we}$	$\mathbf{NMI}_{dp}$
2,500	24,630	50	50	161	1	1
5,000	48,959	100	101	267	1	1
7,500	73,535	150	151	372	1	1
10,000	97,985	201	203	494	1	1
12,500	121,891	249	252	565	1	1
15,000	146,859	303	305	665	1	1
17,500	171,520	355	357	761	1	1
20,000	195,839	401	407	842	1	1
22,500	219,820	455	461	931	1	1
25,000	244,688	502	510	1,031	1	1

As we can see, both IGLP-WE and IGLP-DP not only consistently achieve perfect NMI scores for all datasets, but also run fast. For the 25,000-node ($\sim$245,000

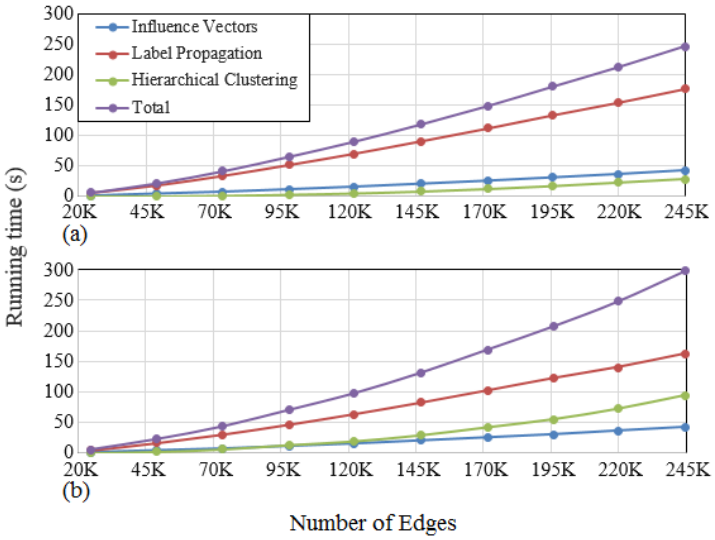

Figure 2.12: Time complexity: (a) IGLP-WE; (b) IGLP-DP

edges) dataset, IGLP-WE produces a community hierarchy of 510 levels in less than 250 seconds, and IGLP-DP delivers a more detailed hierarchical community structure of 1,031 levels within 300 seconds. They are actually faster than many existing algorithms (including *IGSK*), which provide only a single community partition. IGLP-WE and IGLP-DP incorporate the same depth-limited search algorithm as IGSK, which generates the influence vectors efficiently. Its time complexity is $O(m)$ as shown in Figure 2.12. For both IGLP-WE and IGLP-DP, the time complexity is dominated by label propagation. IGLP-DP runs slightly faster than IGLP-WE w.r.t label propagation since IGLP-DP requires no convergent iteration. Actually, the label-propagation process itself is fast with both weighted ensemble and direct passing. Most of the time is spent on calculating the SIN similarity of each node to each of its neighbors, which has a time complexity of $O(Lm)$. Recall L is the average length of influence vectors, which is determined by the average node out-degree b, depth limit d_{max} and and the community structure. It is independent of n and m in general, which explains why the time complexity of label propagation is linear in m as shown in Figure 2.12.

The time complexity of hierarchical clustering, however, is slightly superlinear. For each iteration, we need to visit those boundary nodes and their neighbors to calculate the cluster proximity, which leads to a time complexity of $O(m)$. This is done $(K-1)$ times to build the hierarchy of communities agglomeratively, where K is the number of communities in the initial configuration given by label propagation. Hence, our hierarchical clustering has a time complexity of $O(K_{we}m)$ and $O(K_{dp}m)$ for IGLP-WE and IGLP-DP, respectively. Note that: K_{we} and K_{dp} are much smaller than m, and the number of boundary nodes is a monotonically decreasing fraction of n. K_{we} and K_{dp} are closely related to K_{gt}. As shown in Table 2.6, K_{gt} increases monotonically as the network size increases when we fix both the topological mixing parameter and the min/max community size. While IGLP-WE consistently arrives at an initial community configuration that matches the ground truth closely, IGLP-DP zooms into deeper scales of the community structure and renders an initial configuration that includes many small sub-communities (roughly 2$\sim$3 times of the number of communities of the ground truth). Therefore, IGLP-DP takes more time on hierarchical clustering than IGLP-WE. It is worth pointing out that both IGLP-WE and IGLP-DP deliver a *complete* hierarchy of communities in the sense that no intermediate scale is missing above the initial configuration, as opposed to the well-known *Louvain method* [9]. From this point of view, $O(Km)$ is the best time complexity one can achieve to generate a community hierarchy of K levels.

Overlapping community and role detection. IGSK, IGLP-WE and IGLP-DP can be easily adapted to the detection of overlapping community and individual roles in each community using the approach described in Section 2.4.5. The influence-based belonging factor is a good fit for quantifying the strength of association of a node to a community, and the three community-level influence rankings provide us a new sensible perspective and tool to deal with the role detection. We take IGSK and the Zachary's karate-club network as an example, and list in Table 2.7 its *comprehensive-*

influence ranking, *internal-influence* ranking, *external-influence* ranking, belonging factors, and the community assignment of each node by IGSK with $dephLimit = 2$. It is noted that the partition by IGSK matches the ground truth perfectly. We group the nodes by their community assignments, and sort the nodes in each community by their comprehensive-influence ranking.

Table 2.7: Influence rankings and belonging factors of Zachary's karate club. Int-Rank denotes internal-influence ranking, Ext-Rank denotes external-influence ranking, BF denotes belonging factor, C1 and C2 denote community assignments by IGSK

Node	**Comprehensive Rank**	**Int-Rank**	**Ext-Rank**	**BF to C1**	**BF to C2**	**C1/C2 by IGSK**
1	1	1	3	0.795	0.205	1
3	2	4	1	0.542	0.458	1
2	3	2	5	0.797	0.203	1
14	4	5	2	0.630	0.370	1
4	5	3	6	0.859	0.141	1
8	6	6	7	0.849	0.151	1
20	7	11	4	0.569	0.431	1
6	8	7	10	0.946	0.054	1
7	8	7	10	0.946	0.054	1
5	10	9	10	0.938	0.063	1
11	10	9	10	0.938	0.063	1
18	12	12	8	0.903	0.097	1
22	12	12	8	0.903	0.097	1
13	14	14	10	0.929	0.071	1
12	15	15	10	0.895	0.105	1
17	16	16	16	1.000	0.000	1
34	1	1	2	0.190	0.810	2
33	2	2	5	0.155	0.845	2
9	3	6	1	0.419	0.581	2
32	4	4	2	0.306	0.694	2
24	5	3	9	0.073	0.927	2
31	5	7	4	0.309	0.691	2
30	7	5	10	0.063	0.938	2
28	8	8	7	0.234	0.766	2
29	9	14	6	0.286	0.714	2
15	10	9	10	0.086	0.914	2
16	10	9	10	0.086	0.914	2
19	10	9	10	0.086	0.914	2
21	10	9	10	0.086	0.914	2
23	10	9	10	0.086	0.914	2
10	15	16	7	0.333	0.667	2
27	16	15	16	0.074	0.926	2
26	17	16	18	0.043	0.957	2
25	18	18	16	0.091	0.909	2

BF to C1 and *BF to C2* list the belonging factors of each node associated with communities 1 and 2, respectively, which converts the original non-overlapping assignment into a *fuzzy* overlapping assignment. It is interesting to observe that each node has a higher belonging factor to its own community than to the other community, which follows our intuition. Moreover, we refer to a node with multiple membership as an overlapping node. The belonging factors enable us to identify overlapping nodes with respect to a *belonging threshold*. If a node's belonging factor to a community is great than the belonging threshold, the node is considered as a member of that community. In this case, if we set the belonging threshold to 0.3, we find the set of overlapping nodes are nodes 3, 9, 10, 14, 20, 31 and 32. If the belonging threshold is set to 0.4, then only nodes 3, 9 and 20 are regarded as overlapping nodes. In practice, we can use the *belonging threshold* as a tunable, application-dependent parameter to examine the association of a node to different communities at different scales.

Further, it is straightforward to use the three influence rankings to uncover the roles of individual members in each community. As shown in Table 2.7, nodes 1 (the instructor) and 34 (the president) are of the top ranking on both comprehensive influence and internal influence in communities 1 and 2, respectively. It is reasonable to identify them as *leaders* of their respective communities, which matches the ground truth. Moreover, if we refer to a node as a *core member* if its ranking on both comprehensive influence and internal influence is among top 5 (this number can be adjusted according to the size of the community in practice), nodes 3, 2, 14, and 4 can be regarded as the core members in community 1, and nodes 33, 32, and 24 are core members in community 2. Similarly, if we refer to a node as an inter-community *liaison* if its external-influence ranking is among top 3 (not including the leaders), we find that: node 3, 14, and 20 are liaisons of community 1, and nodes 9, 32, and 31 are liaisons of community 2. As we can see, our approach digs out rich connectivity information for us to probe the structural importance of a node in the network.

2.6 Conclusion

In this chapter, we provide a new, influence-based perspective on network graph topology, and propose a novel *reachability-based* influence-diffusion model. Using this model, we define a new *influence centrality* and a novel *shared-influence-neighbor* (SIN) similarity. Our *influence centrality* differentiates the node's comprehensive influence significance in a more detailed and precise manner, and SIN similarity is well-suited as a refined vertex-pair closeness metric.

For community detection, we first present an *influence-guided spherical K-means* (IGSK) algorithm, which achieves excellent performance in terms of high accuracy. More importantly, we develop two novel *influence-guided label-propagation* algorithms, IGLP-WE and IGLP-DP, for finding hierarchical communities. Both of them consistently uncover the community hierarchy with superior quality and high efficiency. All three algorithms are applicable to both undirected/directed and unweighted/weighted networks. Further, they can be easily adapted to identification of overlapping communities and individual roles in each community. All of these essential tasks are naturally integrated in one framework. Another advantage of IGLP-WE and IGLP-DP is that they can be easily parallelized. It is desirable to parallelize them for large-scale networks of millions of nodes or for online community detection. In addition, it is interesting to combine this approach with content analysis, namely, considering both network graph topology and node profiles for community detection.

Finally, we point out that our *influence centrality* and *SIN similarity* provide important implications for viral marketing and link prediction in social networks. A lot of work can follow.

CHAPTER 3
RESILIENCE ANALYSIS

In Chapter 2, we provide a new, influence-based perspective on network connectivity and propose a novel influence-diffusion model. In this chapter, we present our work on resilience analysis of supply networks. Previous work takes a top-down approach, in which the resilience is measured by some global-level metrics, such as the size of the largest connected component, the average/maximum shortest path length, etc. There are many issues on these metrics. We propose a novel bottom-up approach, in which we adapt our influence-diffusion model for topological resilience analysis of supply networks. We exploit the resilience by investigating the multiple-path reachability of each demand node to other nodes, and quantify the network resilience by aggregating the resilience of all demand nodes in the supply network. Our approach gives rise to a new, comprehensive resilience metric. Further, we adopt multiagent modeling framework and develop new supply-network growth mechanisms to build resiliency into the construction of supply networks.

3.1 Background and Related Work

The supply chain is the construction and management of the flow of goods among suppliers, distributors, retailers, and customers. Traditional supply chains usually maintain a hierarchical structure with a linear flow of goods from suppliers to customers via distributors and retailers. Due to the globalization and fast development of technology, the basic supply chain system has become much more sophisticated, and has rapidly evolved into various supply networks in which links can occur not only between units of different types, but also between units of the same type. For example, some large retailers may distribute goods to small retailers.

Supply networks play an important role in product distribution systems. However, they are often subject to various disruptions, such as unexpected accidents,

natural disasters, terrorist attacks, etc. When the disruption occurs, a few units or connections fail to operate at the onset, but the adverse impact on organizational performance may propagate in the network and eventually lead to devastating malfunction of a great component or even the entire supply system. The sustainability (or say, survivability) of supply networks becomes an important concern. Specifically, the resilience analysis of supply networks under random disruptions and targeted attacks has received considerable managerial attention and extensive studies. There are many challenging questions and open issues in this field. For example, what are the principles that govern how supply networks arise and develop? How can we improve the overall resilience of a supply network against random disruptions and targeted attacks? How can we build resiliency in the supply-network design? A more fundamental question is how to measure resilience.

While conventional disruption studies focus on risk mitigation and contingency planning strategies [73, 123], some researchers investigate the resilience of supply networks from a topological perspective. Previous research [119, 138] has revealed that supply networks share many characteristics that most real-world networks commonly have, and the graph topology of the supply network has great impact on its resilience against disruptions. Criado et al. [29] define a quantitative measure of network vulnerability related to the graph topology. Using a multiagent-based simulation framework, Thadakamalla et al. [121] examine how different network topologies affect the supply-network resilience against random disruptions and targeted attacks in terms of clustering coefficient, size of the largest connected component (LCC), characteristic path length in LCC, and maximum distance in LCC. Nair and Vidal [95] adopt the multiagent model and investigate topology-associated supply-network robustness from the perspective of performance impacts in terms of inventory levels, backorders, and total costs. Based on the complex network theory, Chen and Lin [18] study the characteristics of the complex supply chain and present an invulnerability analysis

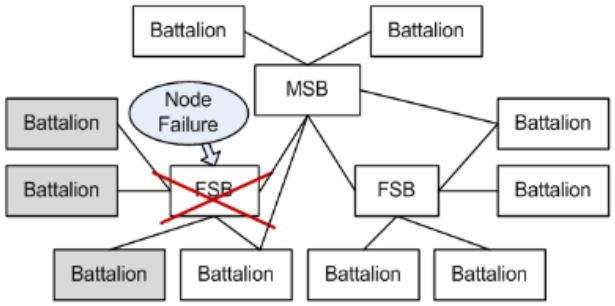

Figure 3.1: A hierarchical military supply chain (given in [143], Copyright by IEEE). FSBs are forward support battalions, and MSB stands for main support battalions

method to evaluate its robustness against random failures and deliberate assaults. Using a military logistic network as a case study (as shown in Figure 1 a hierarchical illustration of it), Zhao et al. [143] propose another taxonomy of resilience metrics that reflect the heterogeneous roles of different types of units in supply networks, and present a hybrid and tunable network growth model. Kim et al. [71] present a comprehensive conceptualization of the supply-network disruption and resilience from a network structural perspective.

Table 3.1 lists some generic resilience metrics used in previous literature [95, 121], which focus on three macro-level network characteristics, i.e., the size of the largest connected component (LCC) and the average and maximum path lengths in LCC. A significant issue of these metrics is that they fail to differentiate the hetergeneous roles of different types of units in supply networks. When using these metrics to evaluate the supply-network resilience, it is implicitly assumed that the roles of all nodes in the supply network are homogeneous. Obviously, this assumption is not realistic. For example, as illustrated in the military logistic network in Figure 3.1, the MSB can be regarded as the supplier or manufacturer, FSBs act as distributors or warehouses, and regular battalions as retailers or consumers. An LCC without the MSB or FSBs should not be considered resilient since the supply flow in such a

sub-network is limited and unsustainable. On the other hand, for a demand node, connecting (or being close) to a supply node is more important and more beneficial than connecting to a regular demand node.

Table 3.1: Some generic network resilience metrics

Name	**Description**
Size of LCC	Number of nodes in the largest connected component (LCC)
Average path length in LCC	Average shortest path length between any pair of nodes
Maximum path length in LCC	Maximum shortest path length between any pair of nodes

The resilience metrics proposed by Zhao et al. in Table 3.2 make more sense. They differentiate supply nodes from demand nodes by considering their heterogeneous roles in supply networks. For example, instead of conventional LCC, they focus on the *largest functional sub-network* (LFSN), which is the LCC with at least one supply node in it. Clearly, these resilience metrics are more realistic than the generic ones in Table 3.1. Unfortunately, they also have some significant limitations.

First, the average/maximum supply path lengths in LFSN are not defined in a sufficiently rigorous manner. We illustrate two simple examples in Figures 3.2 in the context of the military logistic network. Using the taxonomy defined in Table 3.2, the average/maximum supply path lengths are 1.5/2 for both examples. However, it is clear that Example 2 has better accessibility and is more resilient than Example 1. This inaccuracy is introduced because these two accessibility metrics are both based on *all* pairs of supply and demand nodes in the LFSN, which results in the inclusion of some *far-away* supply nodes and adversely decreases the overall accessibility. This is not reasonable. This issue can be addressed by simply defining the average/maximum

Table 3.2: Resilience metrics proposed by Zhao et al. [143]

Name	**Topology-level metric**	**Description**
Availability	Supply availability rate	Percentage of demand nodes that have access to supply nodes
Connectivity	Size of the largest functional sub-network (LFSN)	Number of nodes in LFSN, in which there is a path between any pair of nodes and there exists at least one supply node
Accessibility	Average supply path length in LFSN	Average of the shortest supply path length between *all* pairs of supply and demand nodes in LFSN
	Maximum supply path length in LFSN	Maximum shortest supply path length between *all* pairs of supply and demand nodes in LFSN

supply path lengths as the average/maximum shortest supply path lengths of all demand nodes to their *nearest* supply nodes. To some extent, this helps integrate the effect of the number of supply nodes in the overall resilience analysis. Applying the modified accessibility metrics to the two examples in Figures 3.2, we obtain the average/maximum supply path lengths of 1.5/2 for Example 1 and 1/1 for Example 2, which practically reflects that Example 2 is more accessible and more resilient than Example 1. Zhao et al. detect this issue and get it fixed in [144].

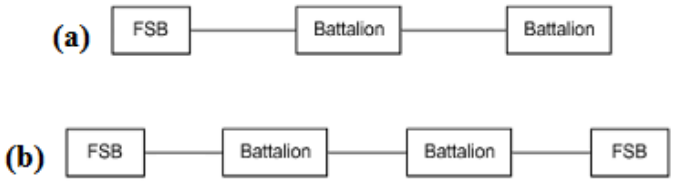

Figure 3.2: Two simple examples: (a) Example 1; (b) Example 2

There are some other issues. For example, how do we differentiate MSBs from FSBs? Should we consider the path length in supply availability rate? Why should

we only consider the LFSN but totally ignore the second largest and all other sub-networks? Should we also consider the number of supply nodes in LFSN? In addition, if the number of supply nodes is fixed, larger LFSN is usually associated with larger average/maximum supply path lengths in LFSN. In other words, better connectivity is associated with worse accessibility. How do we evaluate the overall resilience with such negatively correlated metrics? Further, now that we allow demand nodes to be connected with each other in supply networks, should we give this type of connection some resilience credit as well even though it is supposed to be much smaller than directly connecting to a supply node? All of these imply considerable loss of information in these metrics. While these metrics capture the intuitive concepts of netwrok resilience from some specific angles, it is hard to piece them together to provide a comprehensive and reliable judgement for resilience analysis.

In this chapter, we take a novel bottom-up approach, and propose a new resilience metric that captures the multiple-path reachability-based robustness encoded in supply-network topology in a more realistic and comprehensive manner. Then we present new supply-network growth models that incorporate the heterogeneous roles of units of different types into two fundamental network topologies with various attachment strategies. Using a military logistic network as a case study, we analyze the resilience of different growth models by simulating the network under random disruptions and targeted attacks. Experimental results verify the validity of our resilience metric, and demonstrate the effectiveness of our growth model. Our approach sheds light on the construction of more robust supply networks and more realistic supply-network modeling.

3.2 Methodology

We draw inspiration from the influence-diffusion model that we present in Chapter 2, and adapt it for topological resilience analysis of supply networks. We propose a new network resilience metric by exploiting the multiple-path reachability of each

demand node to other nodes. Then we examine various attachment strategies and perform resilience analysis of supply networks that are characterized with random-graph and scale-free topologies. Our growth models differ from previous work. The key idea is that the supply-network growth model (that is always driven and investigated from the supplier's perspective) should impose few constraints on demand nodes but focus on developing effective attachment rules on supply nodes instead since the supply-network designer has little direct control over demand nodes when they enter the network. In fact, when a demand node enters the network, it usually does not have a global scope of the whole network, such as the shortest paths between all pairs of nodes. It makes more sense to allow them to enter or grow on their own by simply following the basic attachment rule originally proposed in constructing the respective network topology. It is not realistic to apply the same attachment rules to both demand nodes and supply nodes.

3.2.1 Resilience Metric

Our resilience metric distinguishes from existing metrics in the literature. Instead of only focusing on the LCC or LFSN, we consider all nodes in the supply network. More precisely, we measure network resilience based on the resilience of each demand node, which is measured by the *multiple-path reachability* of the node of interest to other nodes. The higher multiple-path reachability a demand node has, the more routes through which it acquires supplies, and thus the more robust it would be when disruptions occur. We argue that the concept of multiple-path reachability captures the essence of supply-network resilience. We quantify the supply-network resilience by the aggregated resilience of all demand nodes in the network.

In Chapter 2, we propose a reachability-based influence-diffusion model. It considers multiple-path reachability, but is not directly fit for supply-network resilience analysis. For example, nodes in our influence-diffusion model are of the same type, but nodes in supply networks are characterized with different types and heterogeneous

roles. We elaborate below how to adapt the model to supply-network scenario. First of all, we develop a similar depth-limited search algorithm to exploit the multiple-path reachability of a demand node to other nodes, but the three important rules are interpreted differently to fit in supply-network resilience analysis.

1. *Cycles are avoided.* It makes sense since revisiting the same set of nodes in a loop or cycle does not improve the resilience.

2. *Revisits along different routes are explored independently.* This mechanism allows us to examine how many different ways the root node can reach other nodes, which captures the essence of resilience in terms of multiple-path reachability.

3. *The path is penalized by its length.* It is reasonable to penalize longer paths since they are usually associated with longer delivery time and more transportation cost in supply networks.

As an example, we illustrate in Figure 3.3 a simple military logistic network, in which node 4 is a main support battalion, node 5 is a forward support battalion, and all other nodes are regular battalions. We show in Figure 3.4 the search tree of node 1 (with a depth limit of 3). The first two rules described above are implemented in the construction of the search tree. For example, when node 1 goes along nodes $2 \rightarrow 4$, the search path does not get back to itself at depth 3 since cycles are avoided. On the other hand, node 5 is visited 4 times along nodes $1 \rightarrow 2 \rightarrow 5$, nodes $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$, and so on.

We consider the heterogeneous roles of nodes of different types by assigning them different significance weights, and penalize the path length using a depth-associated penalty factor α $(0 < \alpha < 1)$. More precisely, we define the *depth-associated penalty* to be α^{d-1}, where d is the depth from the root node to a node of interest following a specific path. Whenever a node is reached from the root node, we multiply the

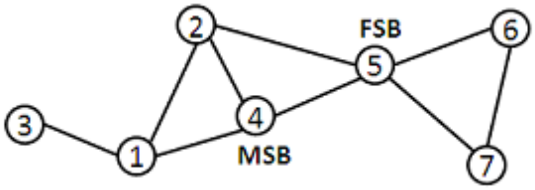

Figure 3.3: Example of a simple military logistic network

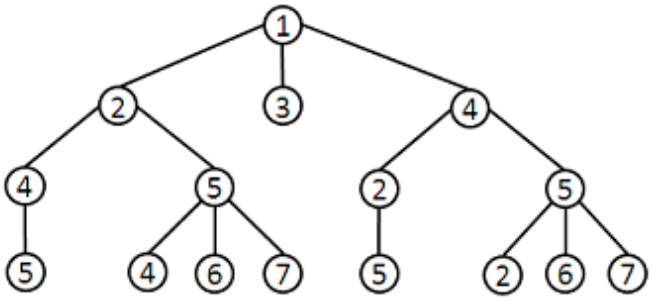

Figure 3.4: Search tree of node 1

significance weight of that node by the corresponding depth-associated penalty, and add it to the resilience score of the root node. In other words, the resilience score of the root node is quantified by the summation of the depth-associated penalized significance weights of all the nodes that the root node visits in its search tree (given a pre-specified depth limit). Using the military logistic network in Figure 3.3 as an example, if we assign a weight of 2 to the MSB, 1 to the FSB, and 0.2 to the regular battalion, then following the path from nodes $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$, the resilience score of node 1 increases by $(\alpha^0 \times 0.2 + \alpha^1 \times 2 + \alpha^2 \times 1)$.

Further, it is straightforward to compute the network resilience score by summing up the resilience scores of all demand nodes. Let D denote the set of demand nodes in the network, T_i denote all the nodes in the search tree of node i (a node may occur multiple times along different paths in the search tree), d_j denote the depth from node i to node j following a specific path in node i's search tree, and W_j denote the

significance weight of node j. The *network resilience score* S is computed as

$$S = \sum_{i \in D} \sum_{j \in T_i} \alpha^{d_j - 1} \times W_j.$$

The pseudocode is presented in Algorithm 3. This network resilience score gives rise to a new, more realistic and finer metric for supply-network resilience analysis. It differentiates the heterogeneous roles of different types of nodes, considers the resilience of all demand nodes, incorporates the reachability to both supply nodes and other demand nodes, includes not only the shortest paths but also other paths, and takes into account the depth-associated penalty. We leave the significance weights, the depth-associated penalty factor α, and the depth limit d_{max} as user-specified tunable parameters such that supply-network designers can adjust them to achieve the desired fit for different applications or different components/paths of the supply network. There is an interesting byproduct of our methodology. When we obtain the network resilience score, we have actually completed the resilience analysis of each individual demand node, which helps supply-network designers/managers identify vulernable demand nodes and components in the network.

Algorithm 3 Calculate Network Resilience Score

```
1:  procedure NRS(G, α, d_max)
2:      nrs ← 0
3:      for node i ← 1, n do
4:          if i is a demand node then
5:              i.score ← 0
6:              Do the modified depth-limited search
7:              for any node j visited in the search tree do
8:                  i.score ← i.score + α^(j.depth−1) × j.weight
9:              end for
10:             nrs ← nrs + i.score
11:         end if
12:     end for
13:     return nrs
14: end procedure
```

3.2.2 Attachment Strategies

Random-graph and scale-free networks [7] present two most fundamental and characteristically distinct topologies. Many researchers use these two network topologies to study network robustness against disruptions. It is observed that the scale-free network is highly robust against random failures but very vulnerable to targeted attacks while the random-graph network shows better performance against targeted disruptions [2]. We develop new attachment strategies to construct various random-graph-based and scale-free-based supply networks, and evaluate their performance using the new resilience metric.

We adopt the multiagent modeling framework. Without loss of generality, we also use the military logistic network as a case study. The military logistic network consists of three types of units: regular battalions, forward support battalions (FSB), and main support battalions (MSB). In a conventional logistic scenario, the MSBs can be regarded as manufacturers, the FSBs as distributors, and the regular battalions as retailers. Both MSBs and FSBs are supply nodes, and regular battalions are demand nodes. As listed in Table 3.3, we build the military logistic network using the same setting and parameters as those used in previous work [121, 143].

As noted earlier, it makes more sense to concentrate on the development of new attachment strategies for supply nodes only, and let demand nodes enter the network following the pure-random and pure-preferential attachment rules in the random-graph-based and scale-free-based supply network models, respectively. Moreover, we need to consider and maintain the basic functionalities of different types of nodes when developing attachment strategies. Specifically, we differentiate MSBs from FSBs. We allow an entering MSB to directly connect to regular battalions, but it has to directly connect to one or more FSBs. This is a sensible assumption to capture that the manufacturer is supposed to directly connect to at least one distributor. When an FSB enters the network, it connects directly to regular battalions but not to any other

Table 3.3: Basic setting and parameters

	Description
Nodes	Start with 10 unconnected battalions, and a new node enters the network at each step
	A total of 990 steps, which generate a network of 1,000 nodes
	The ratio for battalions/FSBs/MSBs is 25:4:1
Edges	An entering battalion initiates one edge to an existing node, and the 2nd edge is initiated with a probability of 0.5
	An entering FSB initiates three edges, and an entering MSB initiates 5 edges
	Neither multiple edges or loops are allowed when new edges are initiated
	The expected number of edges is 1,800 (average node degree is 3.6)

FSBs since a distributor is expected to directly connect to retailers instead of other distributors in general. In addition, we avoid attaching a new entering FSB/MSB to a battalion that already directly connects to an FSB or MSB. Further, we include both preferential-attachment and random-attachment rules for each entering FSB/MSB so as to balance the robustness against random disruptions and targeted attacks.

We list in Table 3.4 three attachment strategies. The first two are conventional random attachment and preferential attachment. The third is the *new supply-specific* attachment that we develop for FSBs and MSBs solely. Finally, as shown in Table 3.5, we create four growth models by applying/combining the three attachment strategies in different ways to investigate how the new supply-specific attachment strategy interplays with the two fundamental network topologies.

3.3 Simulation and Analysis

We develop a simulation program, in which we build military logistic networks based on the four growth models and evaluate the resilience of those networks under random disruptions and targeted attacks.

Table 3.4: Three attachment strategies

Attachment Strategies		**Description**
Random		Attach to a node selected uniformly at random
Preferential		Attach to a node i of degree d_i with the probability $p_i = \frac{d_i}{\sum_j d_j}$
New Supply-specific	FSB	The first edge attaches to a demand node that has the highest degree
		The second edge attaches to a demand node preferentially to its degree
		The third edge attaches to a demand node randomly selected
	MSB	Each of the first two edges attaches to a different FSB randomly selected from the four newly deployed FSBs
		The third edge attaches to a demand node preferentially to its degree
		Each of the last two edges attaches to a demand node randomly selected

3.3.1 Degree Distribution

We first study the degree distribution of nodes in each network to determine how the *new supply-specific* attachment strategy affects the network topology. Figure 3.5 shows the log-log scatterplot of the number of nodes of degree k w.r.t degree k of the four growth models. Comparing the *New-Random* with the *Pure-Random* in Figure 3.5(a), we can tell the *New-Random* changes the random-graph topology with quite a few high-degree hubs that are not commonly seen in random-graph networks. For the *New-ScaleFree* versus the *Pure-ScaleFree* in Figure 3.5(b), the *New-ScaleFree* also slightly changes the scale-free topology by decreasing the degree of high-degree hubs. Although these changes may not be significant, they help balance the robustness against random disruptions and targeted attacks and improve the overall resilience of the network, as shown in the next subsection.

Table 3.5: Four growth models

Models	**Description**
Pure-Random	Apply the random-attachment strategy for all nodes
New-Random	Apply the random-attachment strategy for all demand nodes, and apply the new supply-specific attachment strategy for all supply nodes
Pure-ScaleFree	Apply the preferential-attachment strategy for all nodes
New-ScaleFree	Apply the preferential-attachment strategy for all demand nodes, and apply the new supply-specific attachment strategy for all supply nodes

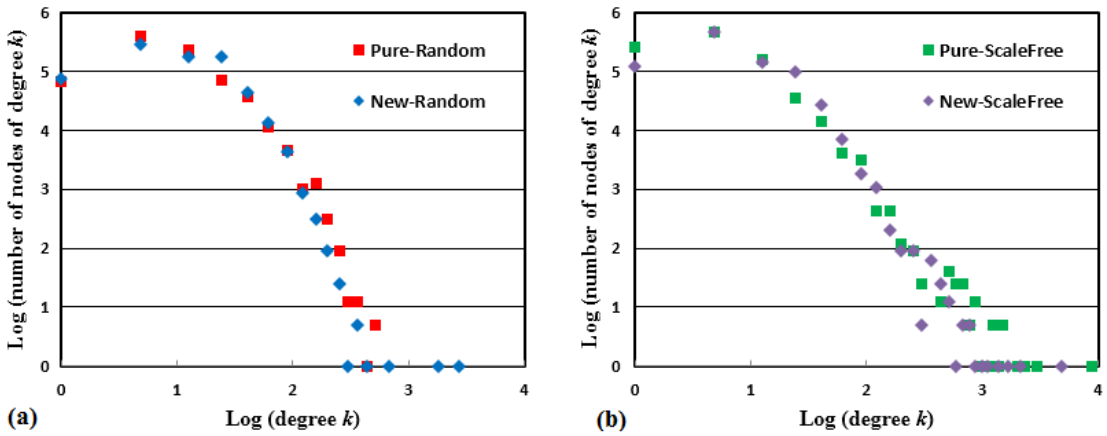

Figure 3.5: Log-Log of the degree distribution of the four models

3.3.2 Resilience Analysis

To evaluate the network robustness against disruptions, we simultaneously remove a batch of 50 nodes (5% of the total nodes) from the network each time and do it 16 times successively, i.e., until 80% of the total nodes are removed. When a node is removed, all of its edges are removed as well. For random disruptions, we remove the nodes uniformly at random each time, and we run the simulation 20 times for each network. For targeted attacks, we remove the nodes in descending order of the node degree. Each time after the batch of 50 nodes are removed, we measure the resilience

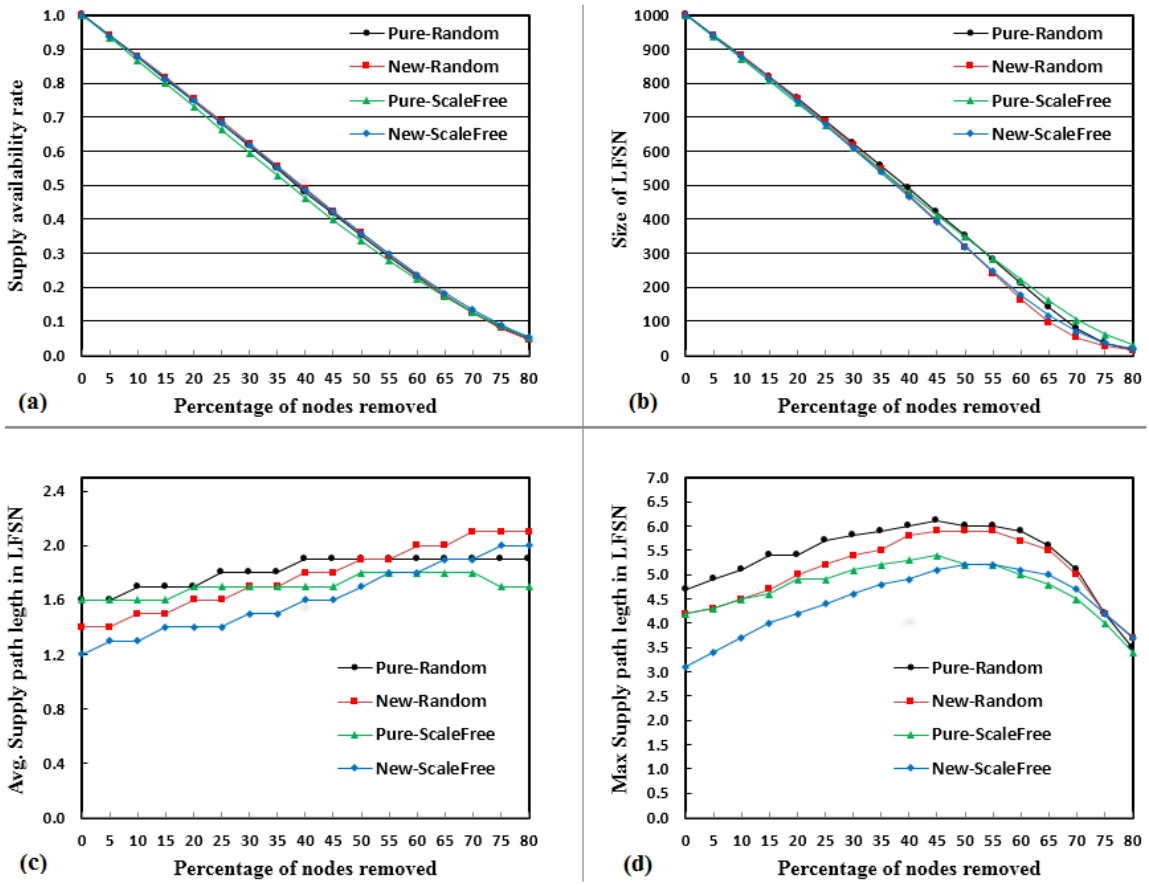

Figure 3.6: Responses of the four growth models to random disruptions, plotted as (a) supply availability rate, (b) size of LFSN, (c) average supply path length in LFSN, and (d) maximum supply path length in LFSN

of the disrupted network using different resilience metrics. Each data point is the average of 10 networks built from the respective growth model.

First, we evaluate the resilience using the four resilience metrics presented in Table 3.2. The first two, i.e., the *supply availability rate* and the *size of LFSN*, are used as they are originally defined. But as discussed earlier, for the *average/maximum supply path lengths in LFSN*, we use the average/maximum shortest supply path lengths of all demand nodes to their *nearest* supply nodes. As we can see in Figure 3.6, all four models are fairly resilient against random disruptions. The *New-Random* has almost the same supply availability rate as the *Pure-Random*, but is slightly

worse in terms of the size of LFSN. It outperforms the *Pure-Random* in terms of the average/maximum supply path lengths. However, given that the size of LFSN of the *New-Random* is smaller than that of the *Pure-Random*, it is expected that the average/maximum supply path lengths of the *New-Random* are shorter than those of the *Pure-Random*. Similarly, it seems that the *New-ScaleFree* dominates the *Pure-ScaleFree* in terms of the supply availability rate and the average/maximum supply path lengths, but it is inferior to the *Pure-ScaleFree* in terms of the size of LFSN. It turns out that we are not able to arrive at a definite or comprehensive resilience ranking of the four models. It even fails to demonstrate that the scale-free network is more robust against random disruptions than the random-graph networks.

For targeted attacks, as shown in Figure 3.7(a) and 3.7(b), both the supply availability rate and the size of LFSN decrease sharply, which reflects that targeted attacks are more damaging than random disruptions to all the four models and that the scale-free networks are even more vulnerable than the random-graph networks. While the *New-Random* shows similar performance to the *Pure-Random*, the *New-ScaleFree* clearly outperforms the *Pure-ScaleFree*. It is noticed that the supply availability rate and the size of LFSN both decrease monotonically as nodes are removed. However, the average/maximum supply path lengths initially increase and then decrease continuously. That is because the LFSN increasingly gets sparser and leaner, which leads to longer average/maximum supply path lengths. After the sparsity of LFSN reaches some threshold, the LFSN becomes fragmented. Then the average/maximum supply path lengths decrease and keep getting shorter. In addition, from Figure 3.7(c) and 3.7(d), one may arrive at a plausible argument that the two scale-free models exhibit better accessibility than the two random-graph models in terms of shorter average/maximum supply path lengths. However, this argument is misleading. As we can see from Figure 3.7(b), the scale-free models have much smaller LFSN (worse connectivity) than the random-graph models, which results in shorter average/maximum

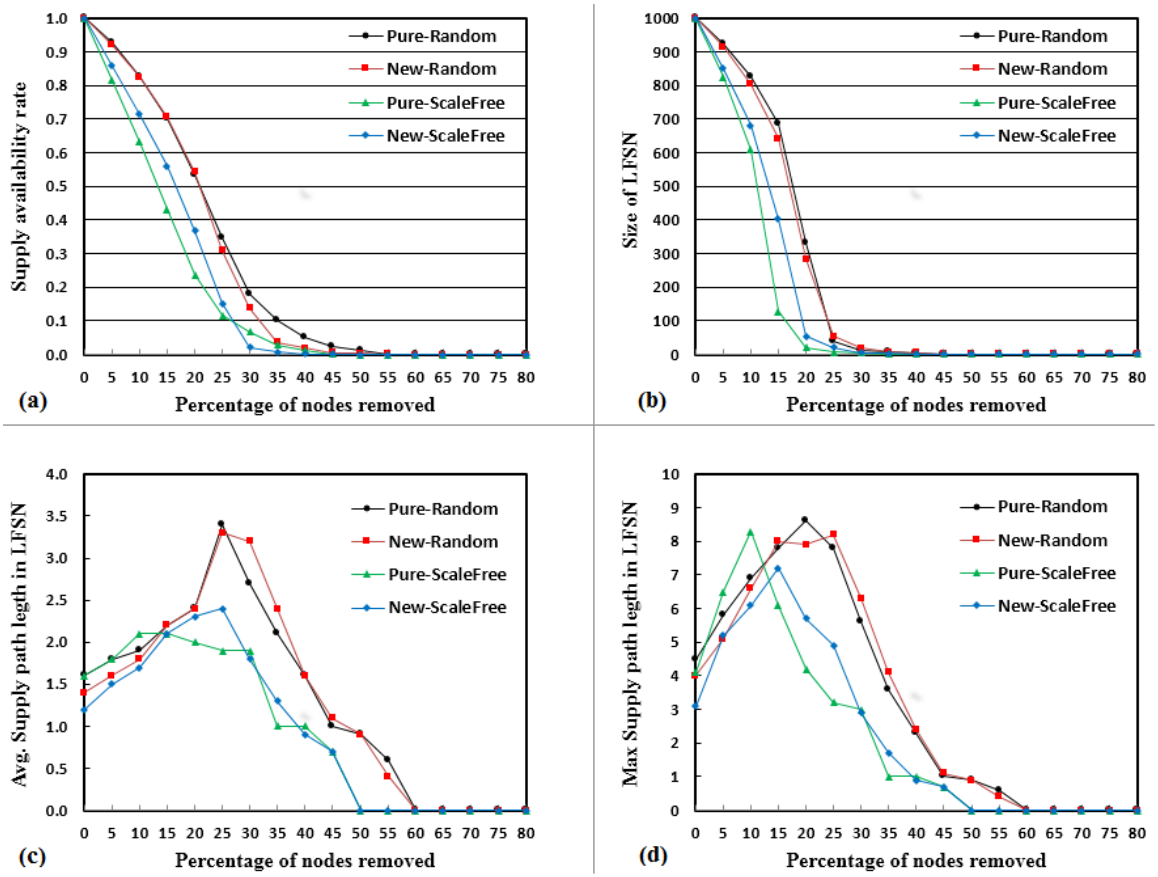

Figure 3.7: Responses of the four growth models to targeted attacks, plotted as (a) supply availability rate, (b) size of LFSN, (c) average supply path length in LFSN, and (d) maximum supply path length in LFSN

supply path lengths in both scale-free models. In fact, these four resilience metrics are closely correlated. It is hard to piece them together to provide a comprehensive and reliable judgement on the network resilience.

Next we evaluate the network resilience using our proposed metric, i.e., the network resilience score defined in Section 3.2.1. We assign a significance weight of 2 to each MSB, 1 to each FSB, and 0.2 to each regular battalion. We set the depth limit to 3 and the depth-associated penalty factor α to 0.5. Figures 3.8 and 3.9 illustrate the changes of the network resilience score along with the percentage of nodes removed in the event of random disruptions and targeted attacks, respectively.

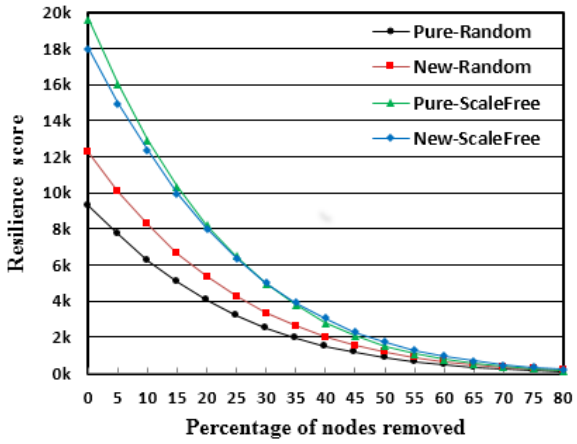

Figure 3.8: Resilience scores of the four growth models to random disruptions

In general, networks with high-degree hubs are more resilient since the hubs make the network diameter shorter. As we can see in Figure 3.8, the resilience scores of the four models vary considerably at the onset. The ranking agrees with the degree-distribution analysis on these models. The *Pure-Random* has the lowest resilience score since hubs rarely occur in random-graph networks. The *New-Random* achieves higher resilience score than the *Pure-Random* since it changes the random-graph topology with quite a few hubs. The *Pure-ScaleFree* reaches the highest resilience score due to its scale-free property. The resilience score of the *New-ScaleFree* is a little bit lower than that of the *Pure-ScaleFree*. That is because the hubs of the *New-ScaleFree* have relatively lower degrees than those of the *Pure-ScaleFree*.

As shown in Figure 3.8, the four models are all fairly resilient to random failures. The resilience scores show alignments with our observation that scale-free networks are much more robust than random-graph networks under random disruptions. The *New-Random* consistently outperforms the *Pure-Random*. The resilience scores of the *New-Random* are 32.6%, 32.5%, and 35.1% higher than those of the *Pure-Random* when 10%, 20%, and 40% of nodes are removed, respectively. The *New-ScaleFree* has almost the same performance as the *Pure-ScaleFree*. The *New-ScaleFree* is slightly

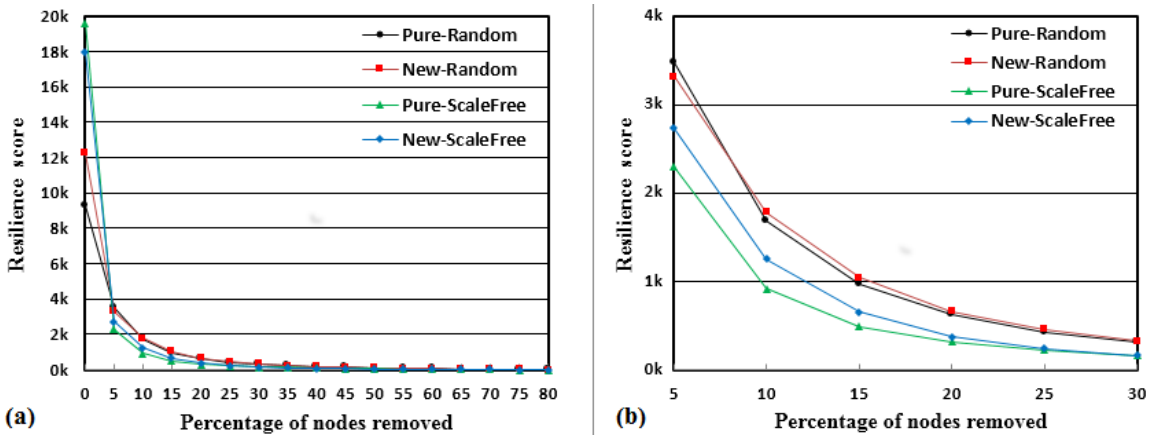

Figure 3.9: Resilience scores of the four growth models to targeted attacks, plotted as (a) 0-80 percent of nodes removed, and (b) 5-30 percent of nodes removed

worse when less than 20% of nodes are removed, but gets a little better when more than 35% of nodes are removed.

For targeted attacks, our network resilience scores support our observation that the damage due to targeted attacks is devastating to all the four models. As shown in Figure 3.9, when only 5% of nodes are removed, the resilience scores of the *Pure-Random*, the *New-Random*, the *Pure-ScaleFree*, and the *New-ScaleFree* drop 62.5%, 72.9%, 88.3%, and 84.8%, respectively. The scale-free networks are more vulnerable and become inferior to the random-graph networks immediately in spite of great advantages at the onset. While the *New-Random* shows slightly better performance than the *Pure-Random*, the *New-ScaleFree* obviously beats the *Pure-ScaleFree*. The resilience scores of the *New-ScaleFree* are 18.9%, 36.7%, 32.2%, and 18.9% higher than those of the *Pure-ScaleFree* when 5%, 10%, 15%, and 20% of nodes are removed, respectively.

Further, we can aggregate the random-disruption resilience score with the targeted-attack resilience score using different weights. Considering that targeted attacks are much more damaging than random disruptions, it is reasonable to assign a higher weight to the targeted-attack resilience score to favor stronger robustness against tar-

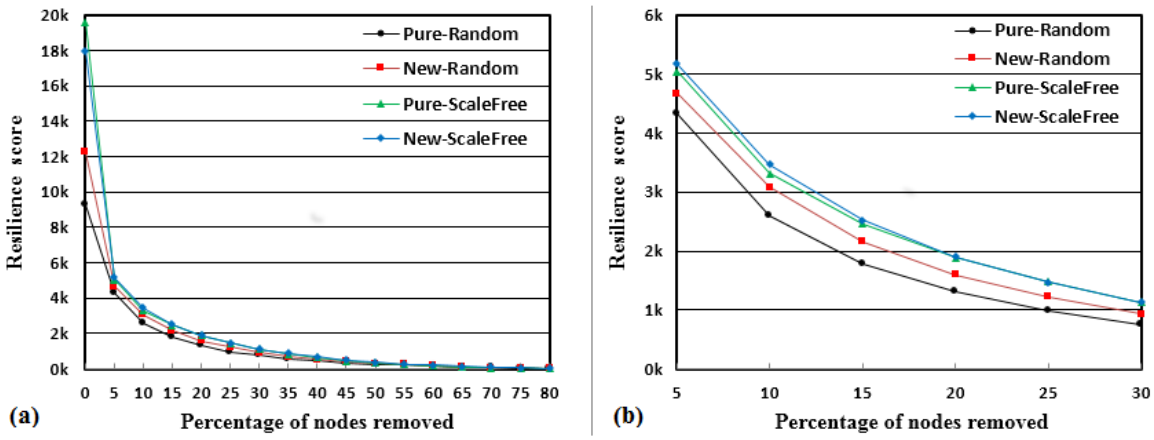

Figure 3.10: Aggregated resilience scores of the four growth models under random disruptions and targeted attacks, plotted as (a) 0-80 percent of nodes removed, and (b) 5-30 percent of nodes removed

geted attacks. We assign a weight of 0.2 to the random-disruption resilience score and 0.8 to the targeted-attack resilience score, and illustrate the aggregated resilience scores in Figure 3.10. As we can see, the *New-Random* obviously beats the *Pure-Random*, the scale-free networks outperform the random-graph networks, and the *New-ScaleFree* has the best performance overall.

3.4 Conclusion

In this chapter, we exploit the multiple-path reachability of each demand node to other nodes in the supply network, and propose a novel network resilience metric. We also develop new attachment strategies that differentiate the heterogeneous roles of different types of nodes. We incorporate them into two fundamental network topologies, and analyze the network resilience against random disruptions and targeted attacks. The experimental results demonstrate the validity of our resilience metric and the effectiveness of our growth model. Our work provides a novel, generic framework and important insights into the construction and resilience analysis of supply networks.

For the future work, it is desirable to take into account inventory levels, backorders, total costs, and inventory reassignment to arrive at a more realistic and more robust growth model. One promising direction is to convert those factors into the weights on the edges of the supply network. Our approach can be easily extended to weighted networks. Another interesting direction is to investigate the cascade-based attack vulnerability on supply networks and/or adapt this approach to the resilience analysis of other complex networks.

CHAPTER 4
VIRAL MARKETING

The influence-diffusion model we present in previous chapters is *reachability-based*. It measures a node's influence significance by the total amount information the node is able to spread out within a pre-specified depth limit. Roughly speaking, it focuses on information's visibility (e.g., brand awareness) regardless of information's effect (e.g., product sale). In this chapter, we investigate *activation-based* influence-diffusion modeling for viral marketing, in which a node's influence is measured by the number of nodes it can activate to adopt a new technology or purchase a new product. We adapt our *reachability-based* influence-diffusion model to the viral-marketing scenario and propose a novel *activation-based* diffusion model. We call it the *multiple-path asynchronous threshold* (MAT) model. It takes into account not only direct and indirect influence, but also influence attenuation along diffusion paths, influence decay over time, and individual diffusion dynamics. Further, we develop a set of heuristics to address the influence-maximization problem under our MAT model.

4.1 Introduction

All of us live in various social networks. We like to share information and ideas with our friends in the form of word-of-mouth (WOM hereafter) communication. Moreover, new technologies and various social media rapidly penetrate into every aspect of our daily life, and provide us new channels and great convenience to exchange information and express opinions. For instance, Twitter had more than 100 million users who posted 340 million tweets a day in 2012. As of May 2015, it had more than half a billion users worldwide. Facebook had over 1.59 billion monthly active users as of August 2015 [1]. People disseminate massive volume of information over different social media, and spread influence on others. As social media becomes prevalent, its

[1] https://www.wikipedia.org

influence on business, politics and society becomes evident and significant. How new innovations, behaviors, and diseases spread through social networks has a long history of study in social sciences. Research in this area has exploded and drawn considerable attention from many disciplines over the last decade. Many models of information and influence diffusion have been proposed for a wide variety of applications, such as viral marketing [8, 69, 85, 110], cascading behavior and prediction [1, 23, 83, 142], outbreak detection [81], etc. In this chapter, we focus on modeling influence diffusion for viral marketing in business.

The term "viral marketing" was coined by a Harvard Business School professor, Jeffrey Rayport, in 1996 in an article for Fast Company magazine called *The Virus of Marketing*. He wrote: *"Think of a virus as the ultimate marketing program. When it comes to getting a message out with little time, minimal budgets, and maximum effect, nothing on earth beats a virus. Every marketer aims to have a dramatic impact on thinking and behavior in a target market; every successful virus does exactly that"*. Viral marketing is termed to describe such a marketing technique that induces the users in a social network to pass on a marketing message (viral ad) to others so as to achieve a potentially exponential growth in brand awareness and product sales. It can be delivered by WOM communication and greatly enhanced by network effects. The commercial success of MSN Hotmail is a classic example. It simply appended a promotional pitch with a clickable URL, *"Join the world's largest e-mail service with MSN Hotmail. http://www.hotmail.com"*, to every message sent by a Hotmail user. It rapidly attracted 12 million users in 18 months with an advertising budget of only $50,000 [65]. Conversely, its competitors like Juno spent $20 million on traditional marketing in the same period, but achieved less effect. Hotmail achieved historically the fastest growth of any user-based media companies at the time.

Viral marketing is driven by WOM communication and hinges on the premise that a consumer's purchasing decisions are often strongly influenced by his family

and friends. This premise is quite reasonable and supported by studies in consumer behavior and marketing strategies. It is widely accepted in consumer behavior that WOM communication among friends plays an important role in shaping consumers' attitudes and behaviors. In an early study, Katz and Lazarsfeld [67] found that WOM was the most important source of influence in the purchase of household goods and food products. It was seven times as effective as newspapers and magazines in influencing consumers to switch brands. More recent studies also reveal that consumers increasingly rely on advice from others in their personal or professional networks [56,62,115]. While there are more widespread and instant-messaging marketing channels such as web pages and mobile phones, family and friends always have the highest level of trust. According to the Nielsen Global Trust in Advertising Report released in 2015, 83% of online respondents in 60 countries said they trusted the recommendations of friends and family. The most credible form of advertising comes straight from the people we know and trust, and the referral from a friend conveys a direct and strong endorsement of the product. To this end, WOM promotion is regarded as the most powerful and cost-effective marketing tool. With the widespread use of social media and mobile phones, WOM has become an even more powerful resource for both marketers and consumers.

Research has investigated WOM effects on viral marketing from various perspectives. Brown and Reinegen [15] presented a network analysis of WOM referral behavior and examined different roles of strong and weak social ties for the flow of referral information. Domingos and Richardson [32] mined the network value of customers using collaborative filtering systems, in which they modeled the customers' influence on each other as a Markov random field. Thomas [122] illustrated a conceptual framework to achieve the amplification of initial marketing efforts by third parties through their passive or active influence. Kozinets et al. [74] proposed a network coproduction model and presented a case study of a WOM marketing campaign, in which the prod-

uct (mobile phone) was seeded with prominent bloggers in online communities so that they could communicate favorably about it to other bloggers. Figure 4.1 illustrates three influence models presented in [74]. The earliest and simplest is the *organic interconsumer influence model*, in which WOM occurs *organically* among consumers without direct promoting and influence by marketers. The *linear marketer influence model*, however, includes marketers as important actors in the WOM communication. Marketers attempt to influence selected opinion leaders with advertisements and promotions, hoping they would spread the influence to other consumers. The *network coproduction model* is more sophisticated but more realistic. In this model, marketers are directly involved in managing WOM activity, and consumers are regarded as active coproducers of WOM messaging. There are two distinguishing characteristics built in this model. First, marketers directly target and influence the consumer or opinion leader through one-to-one seeding campaigns. Second, market messages do not flow unidirectionally from seeded consumers to others, but rather are exchanged among any connected consumers in the network. Based on this descriptive model, Kozinets et al. [74] generate some insights and propositions about WOM using qualitative data analysis and constant comparative method.

In essence, viral marketing is a process of influence diffusion over social networks. An effective viral marketing campaign requires that marketers identify individuals with high social networking potential. A general setting can be depicted as follows: A company would like to market a new product, in the hopes that it would be adopted by as many people as possible in a target social network. The company needs to choose a small number of influential individuals as the initial adopters/seeds by giving them free/discounted samples of the product and encourage them to recommend the product to their friends, hoping that their friends would be influenced to purchase the product and continue to influence their friends to buy the product. As a result, the influence would propagate through the network and trigger the widespread adop-

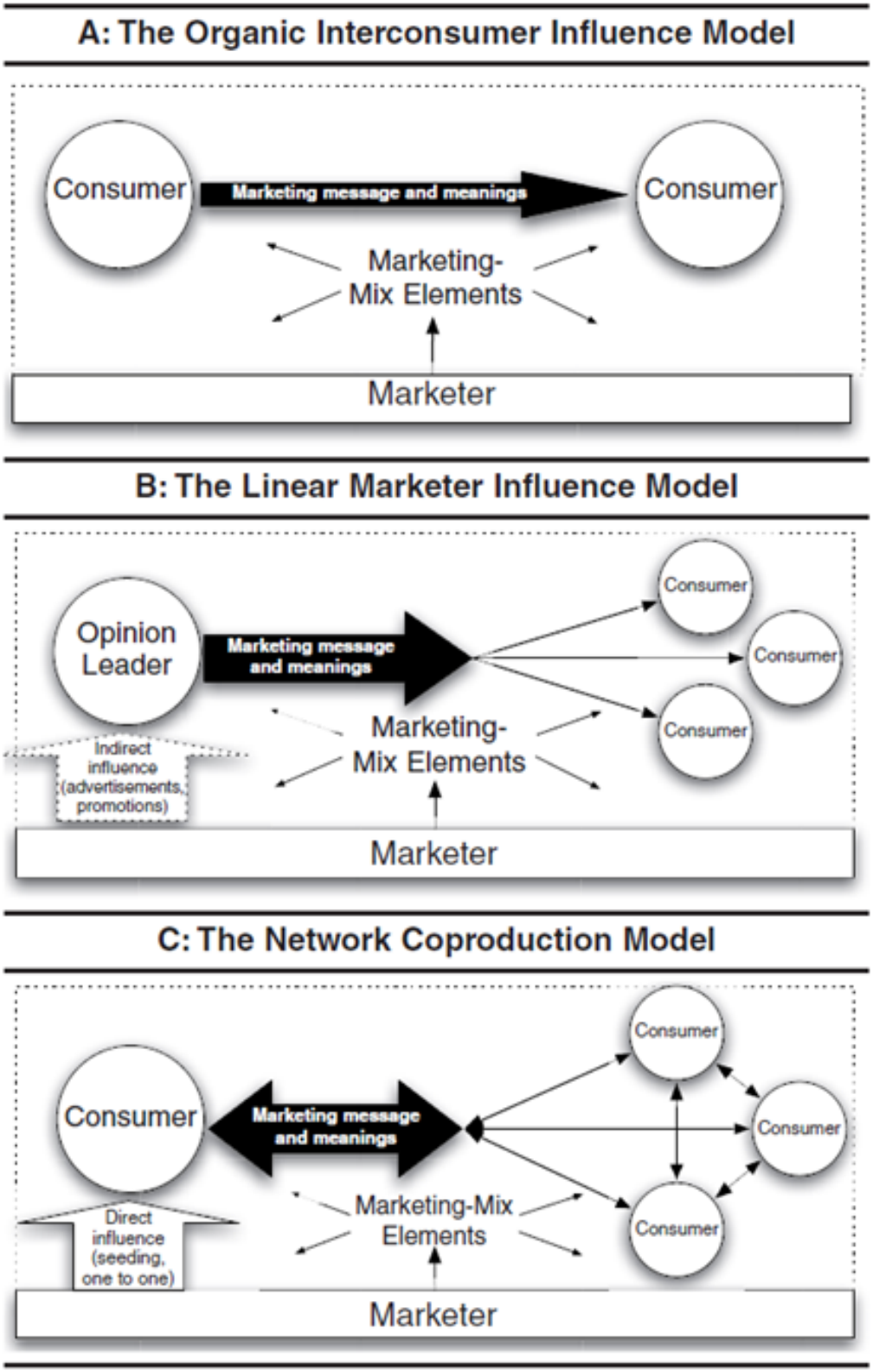

Figure 4.1: The evolution of WOM models (given in [74], Copyright by The American Marketing Association)

tion in the end of the diffusion process. Survey-based statistical research has shown very strong support for the hypothesis that network linkage can directly affect product/service adoption [56, 62]. The crucial factor to the success of viral marketing is for the marketers to identify the most influential set of initial adopters. Researchers investigated different approaches to seeding campaigns for marketing practice. Hinz et al. [57] compared four seeding strategies in two small-scale field experiments and one real-life viral marketing campaign of a mobile phone service provider. Their empirical results showed that seeding to well-connected individuals (social hubs) is most successful, which can be up to eight times more successful than other seeding strategies. Iyengar et al. [62] studied how opinion leadership and social contagion within social networks affect the adoption of a new prescription drug by physicians. They used hazard modeling as the main statistical approach to analyze the data and test the hypotheses. They found strong evidence of contagion operating over network ties and that the amount of contagion is moderated by both the recipients' perception of their opinion leadership and the sources' volume of product usage. Libai et al. [86] further explored the value of WOM seeding programs in terms of market expansion and purchase acceleration. They investigate how expansion and acceleration integrate to create programs' social value using an agent-based simulation model [109] and 12 social networks in various markets as input. They also ran a pooled regression across the 12 networks to examine the factors that affect the acceleration ratio, which include network size, average degree and clustering coefficient, etc. Using the same approach, Peres [105] present more findings on the impact of network characteristics on new product diffusion.

These studies mentioned above significantly enhance our understanding of WOM behavior and effects on promoting viral marketing, and provide substantial insights and implications for the design and evaluation of seeding strategies for both marketing practice and diffusion research. However, the approach these researchers commonly

use or rely on is explanatory modeling. They apply various field experiments and/or statistical models to data for testing some causal hypotheses or correlations between variables. For example, researchers find support for the hypothesis that referrals from strong ties are more influential in receivers' decision-making than those from weak ties [15], and that high-degree seeding remains the most successful strategy [57]. Nevertheless, how can we create a robust seeding strategy that outperforms the high-degree seeding? Is it possible to find the optimal or suboptimal seeding strategy? Can we estimate the cascade of adoption given any set of initial adopters? Most of these qualitative approaches and empirical studies fail to give answers to these important questions in an operational manner. One significant limitation is that an explicit information/influence diffusion model is missing.

Just as most marketing researchers focus on investigating WOM diffusion processes using various explanatory models and/or qualitative methods, researchers from computer science and other related fields aim to explicitly build influence-diffusion models so as to maximize and/or predict the influence spread. Influence maximization as an important algorithmic technique for viral marketing was first posed by Domingos and Richardson [32, 110], in which they applied Markov random fields to model the influence among customers and then choose the best marketing plan to maximize the profit. In their seminal paper [69], Kempe et al. formulate it as a discrete optimization problem: given a social network, a stochastic diffusion model, and a number K (also called budget), the objective is to find the seed set of K initial adopters (of a new product) who can trigger the largest cascade of further adoptions, in which the influence diffusion process unfolds in discrete time steps as described by the diffusion model.

This influence-maximization problem has attracted a great deal of attention and extensive studies. To address this problem, two key components are indispensable. First, we need to build a realistic influence-diffusion model that captures in detail the

diffusion process and the activation mechanism of WOM marketing. Second, we need to develop an effective and efficient algorithm that enables us to find the optimal or suboptimal seed set under the diffusion model. Kempe et al. [69] approach the problem under two widely-studied stochastic diffusion models: *independent cascade* (IC) model and *linear threshold* (LT) model. They show that this optimization problem is NP-hard for both the IC and LT models, and provide the first provable approximation guarantees for efficient algorithms. Their work laid theoretical and algorithmic foundations for understanding influence diffusion and addressing the influence-maximization problem. Since then, there has been a large amount of follow-up work in two main directions: (1) extensive research has been done to study various extensions to the IC and LT models [8, 13, 19, 44, 49, 55]; (2) a large number of algorithms have been proposed to improve efficiency in finding the K-seed set [22, 51, 64, 81, 125]. We will elaborate in the next section several representative models and algorithms most related to our work.

The IC and LT models in [69] are fundamental algorithmic models for influence diffusion and lie at the core of most generalizations. In the IC model, each node u that is newly activated in step t_i is given a single chance to independently activate each of its currently inactive neighbors v in step t_{i+1}. It succeeds with a pre-specified uniform propagation probability p. Whether or not it succeeds, u cannot make any further attempts to activate v in any subsequent steps. The diffusion process is repeated until no more activation is possible. The LT model is based on the use of node-specific thresholds. Each node v is assigned an activation threshold θ_v uniformly at random from $[0, 1]$. In any step of the diffusion process, an inactive node becomes active once the total weighted fraction of its active neighbors is greater than or equal to its threshold. Goyal et al. [49] develop data-driven models and algorithms for learning influence probabilities using an action log. Chen et al. [19] propose an extension to the IC model to incorporate the emergence and propagation of negative opinions.

Bhagat [8] extend the LT model by defining an objective function that distinguishes product adoption from influence. The LT model has also been extended to account for competition of influence diffusion in social networks [13,55]. Moreover, Gayraud et al. [44] extend both IC and LT models to evolving social networks. These extensions are the substantial complement of the classic IC and LT models. However, there exist some significant limitations in these models.

First, they approach a social entity's adoption likelihood from a rather confined scope, considering only the direct influence from the activated neighbors of the entity, which is in spirit similar to the *linear marketer influence model* in Figure 4.1. However, a social entity's adoption decision could be influenced by other adopters who are not social neighbors [37], e.g. through *structural equivalence* [16] and *three-degree-of-influence* [24–26]. In reality, as described in the *network coproduction model* in Figure 4.1, WOM messages do not flow unidirectionally from seeded consumers to others, but rather may be exchanged among any connected consumers in the network. This is an important feature of WOM communication. In other words, a node in the network could be influenced and activated by its inactive neighbors who pass on the influence from other influencers to the node of interest. For example, Jenna watched the movie "Zootopia" and told her friend Cindy that the movie is awesome. Cindy did not get a chance to watch the movie, but she passed the words to their friend Anna that Jenna watched the movie and liked it. Anna got influenced and watched the movie soon. The IC and LT models fail to capture such *indirect influence* passed along by messengers who may or may not be activated. Fang et al. [37] take a data-driven approach to study adoption behavior in social networks. Their findings suggest that the diffusion models relying exclusively on direct influence are limited in predictive power. We argue that messengers play an important role in influence diffusion for viral marketing, which should be taken into account to build a more realistic influence-diffusion model.

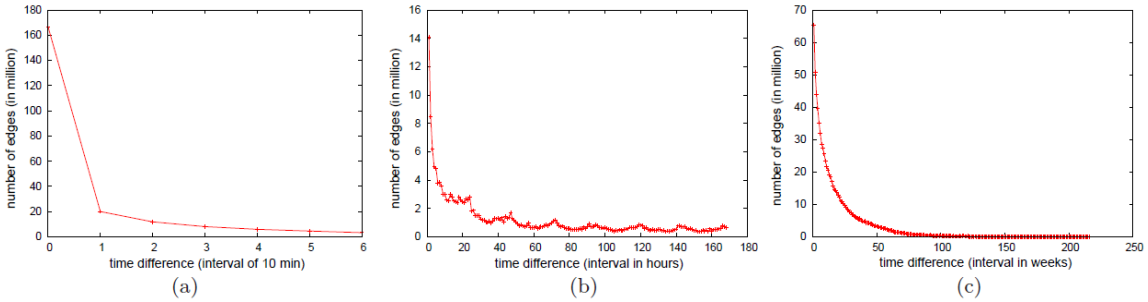

Figure 4.2: Frequency of common actions vs. the time difference between two users performing actions. (a) during the first hour at a granularity of 10 minutes; (b) during the first week at hourly granularity (without considering the cases in which the time difference is less than one hour, i.e., the cases in (a)); (c) the rest of the dataset with weekly granularity (given in [49], Copyright by ACM)

Second, the IC and LT models (and most generalizations) overlook the fact that influence does not remain static or constant, but rather attenuates along diffusion paths and decays with time. As indicated in the *three-degrees-of-influence* phenomenon, while it keeps propagating up to a social horizon of three hops, peer influence gradually dissipates along the diffusion path. Supportive experimental studies have continued to appear [12,75,87,94]. On the other hand, Goyal et al. [49] measure the number of actions that propagate between pairs of neighbors in Flickr at different time intervals. As shown in Figure 4.2, there is clearly an exponential decay in the number of actions propagated as the time elapses, confirming our intuition that influence decays with time.

Third, these models fail to capture the individual temporal diffusion dynamics. In the IC model, each newly activated node gets a single chance to simultaneously execute an activation attempt to each of its inactive out-neighbors in one time step. In the LT model, an inactive node simultaneously checks each of its in-neighbors to compute the total influence weight in any time step. Both of them are synchronous diffusion models. It turns out that delaying the activation attempts of any node in the IC model and delaying adding the weights of some out-links of a newly activated node

in the LT model would not change the distribution of the final active set [20]. This time-invariant property is not realistic or desired. In practice, the time constraint and spreading speed are always critical concerns of marketers since they are closely related to profit and competition. In fact, information diffusions generally take place in an asynchronous way in reality. People communicate much more frequently with their close friends than regular friends. Even if a person sends out a message simultaneously to all her friends on Twitter, they may log in and check out the message at different times. As shown by Iribarren and Moro in their viral email experiment [61], there exists large heterogeneity in human response time at the individual level, which has great impact on the dynamics of information diffusion at the collective level.

There is another subtle but more fundamental issue in existing models. WOM's effectiveness as an information source for consumers can be broken down into WOM's reach and WOM's impact. Unlike epidemic spreading in which each exposure acts independently, WOM's impact is usually derived from *social reinforcement*, where repeated exposures continue to have large marginal effects on adoption [17, 111]. We thus consider product adoption as an accumulation process of influence. More precisely, we describe influence as a three-stage process as shown in Figure 4.3. The first stage is *awareness*, where an inactive entity gets exposed to the WOM about the new product. The second stage is *aggregation*, in which the entity gets reinforced with more and more WOM exposures. The last stage is *activation*, where the entity adopts the new product when the accumulated influence is greater than a certain threshold of the entity. We refer to this procedure as the "3A process". To the best of our knowledge, no models in the literature explicitly reveal this process in an incrementally aggregate manner since they are not able to quantify the influence or measure the amount of influence accumulated and passed from one on to another.

We aim at developing a more realistic influence-diffusion model to address these issues. We adapt our reachability-based influence-diffusion model (developed for

Figure 4.3: 3A process of influence

community detection in Chapter 2) to the viral-marketing scenario, and arrive at a *multiple-path asynchronous threshold* (MAT) model. In this model, we naturally integrate the three stages of the 3A process, and quantify in an incremental manner the aggregate consequences from informational influence to activation on the basis of complex WOM communication. Our MAT model is the first model to this regard in the framework of the general threshold model. We consider both direct and indirect influence, take into account influence attenuation along diffusion paths and influence decay with time, and model the individual temporal diffusion dynamics using a contact-frequency-based Poisson process. Further, we develop two effective approximation algorithms to address the influence-maximization problem under our MAT model, and conduct experiments on four real-life networks. Our work provides preliminary but important insights and implications for viral marketing and diffusion research in other fields.

We note that *homophily* may also explain product adoption behaviors. In our viral-marketing setting, the influence is embedded in the WOM messages, and homophily may facilitate the spread of WOM-based influence. While some researchers work on distinguishing influence-based contagion from homophily-driven diffusion [4, 76], we focus on the comprehensive WOM effects on production adoption, without necessarily distinguishing the causes. Especially, like most existing influence-diffusion models, our model is built upon network connectivity and structure without using any node attributes or profiles.

4.2 Related Work

Viral marketing is an important and cost-effective marketing technique in business. One of the fundamental problems in viral marketing is to find a small set of initial adopters who will trigger the largest further adoptions through WOM-based influence propagation. To address this problem, we need a realistic influence-diffusion model and an effective influence-maximization algorithm. There are a large number of studies in the literature. Detailed surveys can be found in [3, 20, 70]. Here we focus on several representative models and algorithms most relevant to our work and considerations.

4.2.1 Diffusion Models

In 2003, Kempe et al. published a seminal paper on maximizing influence spread in social networks [69]. They formulate influence maximization as a discrete optimization problem, and investigate the step-by-step dynamics of adoption under two basic diffusion models. One is the *independent cascade* (IC) model, and the other is the *linear threshold* (LT) model. These two classic models lie at the core of a large number of generalizations in the subsequent work.

The IC model takes the social network $G = (V, E)$, the influence probability $p(\cdot)$ on each link, and the initial seed set S_0 as input, and the influence-diffusion process unfolds in discrete time steps according to the following randomized propagation rule. At every time step t_i ($i \geq 0$), each newly-activated node u is given a single chance to activate each currently inactive out-neighbor v independently. It succeeds with the pre-specified influence probability p_{uv}. If successful, then v becomes active in step t_{i+1}. But whether or not u succeeds, it cannot make any further attempts to activate v in subsequent steps. The diffusion process runs until no new nodes are activated. Such diffusion behavior is referred to as *simple contagion* in social science, in the sense that activation could be triggered by a single influencer independently. This model is well-suited for epidemic spread since an individual may get infected once

he gets exposed to the virus. For viral marketing, it does not fit very well. With respect to the 3A process as described in the previous section, the IC model reflects the brand-awareness stage, but fails to explicitly capture the aggregation or activation stages since most people cannot be convinced to adopt a new product right after one single WOM interaction. Kempe et al. also propose a *general cascade model* in [69]. They define an incremental function $p_v(u, S) \in [0, 1]$, where u is a newly activated in-neighbor of v and S is the set of active in-neighbors of v that have already tried (but failed) to activate v in previous steps. When u attempts to activate v, it succeeds with probability $p_v(u, S)$. They give the active in-neighbors of v multiple chances to activate v as long as v has a newly activated in-neighbor at any time step.

In the LT model, every link $(u \to v)$ of the social network G is associated with an influence weight $b_{uv} \in [0, 1]$, indicating the influence significance of u on v. Before the diffusion process starts, each node v is assigned a threshold θ_v uniformly at random in the range $[0, 1]$. The threshold can be interpreted as the personal latent tendency of a node to adopt a new product. Once the initial seed set S_0 is given, the diffusion process then unfolds deterministically in discrete time steps. In any step, an inactive node v becomes active if and only if the total influence weight of its active in-neighbors N_v^{in} is greater than or equal to its threshold θ_v, i.e.,

$$\sum_{u \in N_v^{in}} b_{uv} \geq \theta_v.$$

The diffusion process continues until no further activation is possible. As opposed to the IC model, the LT model captures *complex contagion*, in which an individual is usually not activated until she receives positive reinforcement from multiple influencers. The LT model is more suitable for viral marketing than the IC model in this regard. Similarly, Kempe et al. also propose a *general threshold model* in [69] by lifting the assumption that the influence from individual in-neighbors can only be aggregated linearly. In the general threshold model, each node v is associated with

an arbitrary monotone threshold function $f_v(S) \in [0,1]$, where S is the set of v's active in-neighbors and $f_v(\phi) = 0$. Now v becomes active if and only if $f_v(S) \geq \theta_v$. Clearly, the LT model is a special case of the general threshold model, in which each threshold function has the form

$$f_v(S) = \min(1, \sum_{u \in S} b_{uv}).$$

Surprisingly, while the IC model and the LT model are two specific models for influence diffusion from different perspectives, the general cascade model and the general threshold model are proved to be equivalent and can be converted to each other [69]. Both the influence probability in the IC model and the influence weight in the LT model are assumed to be given as input. However, neither of them is directly available in reality. In fact, how to accurately estimate them is an important challenge. Kempe et al. did not address the issue of inferring actual influence parameters. In the IC model, they simply assigned a uniform probability of p to each edge, choosing p to be 0.01 and 0.1 in separate trials. In the LT model, they treat the ratio of multiplicity of parallel edges between a pair of nodes to the degree of the node of interest as its influence weight.

Goyal et al. [49] devise three probabilistic influence models to estimate the influence probabilities on edges, using the action log of every user. They first propose a static model with three variations. For instance, they define the *maximum likelihood estimator* (MLE) of success probability as the ratio of number of successful attempts over the total number of trials that an active user performs to influence its inactive neighbor. More interestingly, they measure the number of actions that propagate between pairs of neighbors at different time intervals, and demonstrate that influence decays with time exponentially as shown in Figure 4.2. Thus, they present a *continuous-time* (CT) model and a *discrete-time* (DT) model to capture the temporal dynamics of influence probability. They also propose an instance of the general

threshold model, in which the threshold function is defined as

$$p_u(S) = 1 - \prod_{v \in S}(1 - p_{vu})$$

where p_{vu} is the learned influence probability of v on u. They validate their methodology in a case study of a Flickr dataset consisting of $1.3M$ nodes, $40M$ edges, and an action log of $35M$ tuples referring to $300K$ distinct actions. Obviously, this data-driven model requires an action log that is accurately collected and large enough (from tens to hundreds of millions of tuples) to learn the model parameters, which is not readily available or may be difficult to obtain in most social networks.

Bhagat et al. [8] bring up an interesting point. They argue that it is important to distinguish product adoption from influence. They show that there exist *tattlers* who are influenced but without adopting the product themselves. These tattlers serve as information bridges in influence propagation and make a significant difference to product adoption. They adapt the LT model to what they called *linear threshold with color* (LT-C) model, as shown in Figure 4.4 in the form of a state diagram. Let A be the set of active in-neighbors of an inactive node v. They instantiate the threshold function $f_v(A)$ as

$$f_v(A) = \frac{\sum_{u \in A} w_{u,v}(r_{u,i} - r_{min})}{r_{max} - r_{min}}$$

where $w_{u,v}$ is the influence weight of u on v, $r_{u,i}$ is u's rating on product i, and r_{max} and r_{min} represent the maximum and minimum ratings in the system, respectively. Node v becomes active if $f_v(A) \geq \theta_v$. Then v may adopt the product with some probability λ_v or simply tattle about the product. A tattler can either enter the *promote* state with a probability μ_v or enter the *inhibit* state.

The LT-C model requires that the social network under consideration is associated with a large number of user ratings on different products, such as Netflix, Amazon

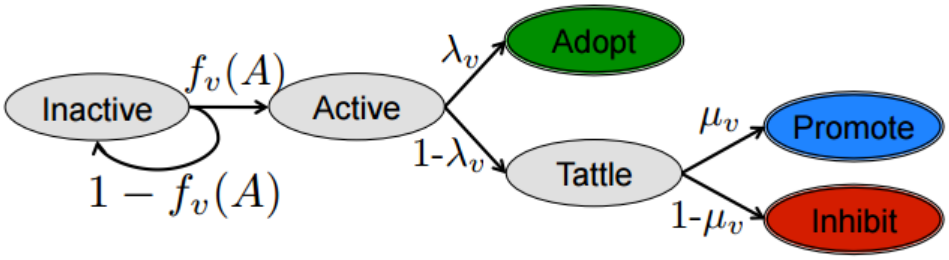

Figure 4.4: LT-C model with colored end states: adopt-green, promote-blue, inhibit-red (given in [8], Copyright by ACM)

and Flixster. Not only is the threshold function f_v computed based on the user rating log, detailed user log information is needed to determine whether an activated user becomes an adopter or a tattler so as to learn λ_v and μ_v. They rely on these user rating logs to compute the ratings matrix. However, the ratings matrix is commonly very sparse with a large number of missing values corresponding to users who have not rated a product, especially for a new product just or to be launched in the market. Therefore, they have to assume that the new product has been adopted by some small number of early adopters who have assigned ratings to the product, and then use *matrix factorization* to predict these missing ratings. All of these confine the applicability of this model, and may affect its accuracy. More importantly, there is a subtle but significant issue in this model. We agree that it is a step towards a more realistic model by distinguishing tattlers from adopters. However, a tattler does not have to be activated to be able to spread influence in reality. Many tattlers are simply *messengers* passing WOM messages on to their friends. They do not have to form opinions of their own. This is an important feature of WOM communication. For example, Tom bought an iPhone, and told his friend Jeff that it is cool and he likes it. When Jeff chats with their friend Nick at lunch, he simply tells Nick the fact that Tom just bought an iPhone and he really likes it. Jeff does not have to be activated to buy an iPhone of his own or form his opinion about iPhone at all, but he does

pass *indirect/passive/informational* influence on to Nick. In fact, messengers play an important role in WOM marketing, and should be distinguished and included in influence-diffusion modeling.

To incorporate the temporal aspects observed in diffusion dynamics, Chen et al. [21] investigate time-critical influence maximization. The goal is to maximize influence spread within a given deadline. They propose an *IC model with meeting events* (IC-M), in which each edge (u, v) is associated with a meeting probability $m(u, v)$ to reflect the time-delayed diffusion dynamics. They present two methods to determine the meeting probabilities. One is the weighted method with $m(u, v) = \frac{c}{d_u^{out}+c}$, where c is a user-defined smoothing constant and d_u^{out} is the out-degree of u; the other is a random method in which $m(u, v)$ is selected from $\{0.2, 0.3, \ldots, 0.7, 0.8\}$ uniformly at random. It is no doubt a realistic step to incorporate temporal diffusion dynamics of individual nodes. However, this model has limitations on the following two respects. First, the meeting probability would be meaningless if there is no deadline constraint or the deadline constraint is set to a relatively large number. As noted in previous section, the IC model has a time-invariant property, i.e., delaying the activation attempts of any nodes would not change the distribution of the final active set. In other words, the meeting probability has an effect in IC-M model only because of the deadline constraint, which apparently underestimate the impact of individual temporal dynamics on influence diffusion. We argue that it would make more sense to examine it by associating the temporal diffusion dynamics with the temporal influence decay. Second, they fail to capture the *contact frequency* in their meeting probability. For most weighted social networks, the weight on a link is a natural and important indicator that describes the strength of the relationship and/or the contact frequency. A realistic diffusion model should incorporate the weight information to better capture the individual temporal diffusion dynamics.

There are many other extensions of the classic IC and LT models to different

scenarios. Chen et al. [19] extend the IC model to incorporate the emergence and propagation of negative opinions. The LT model has been extended to address influence maximization under competition [13, 55]. Gayraud et al. [44] extend both the IC and LT models to study the influence diffusion in dynamic networks.

However, all the aforementioned models are probabilistic models. None of them explicitly quantify the influence or directly compute the aggregation of influence to reveal the 3A process from *awareness* $\rightarrow$ *aggregation* $\rightarrow$ *activation*. The model proposed by Ma et al. [90] can be regarded as an attempt in this direction. They model the diffusion of innovations as *heat-diffusion processes*. The seed nodes are regarded as heat sources of a very high temperature, and their heat flows throughout the network following the heat diffusion theory from physics. This heat-diffusion model incorporates influence diffusion and aggregation in terms of heat flow. Unfortunately, there are several crucial issues associated with this model. First, influence is not like heat. When a node transmits its heat to its neighbors, its temperature drops. The more heat it spreads out, the faster the temperature drops. This is not true for influence. A node does not lose its influence after it diffuses its influence to its neighbors. Second, heat always flows from a high-temperature node to a low-temperature node. However, influence does not have this directionality, especially in WOM communication. Any node can pass WOM messages and influence on to its neighbor. Third, it fails to capture an important feature that should be included in the influence-diffusion model, that is, a newly activated node should have a jump of its temperature and increase its capability to influence others. Finally, the initial heat of seed nodes and the activation threshold are not set at the same or reasonable scale.

4.2.2 Influence Maximization

The influence-maximization problem is formally described as follows: Given a social network $G = (V, E)$ with a total of n nodes (i.e., $|V| = n$), a stochastic diffusion model on G, and a number $K (K \ll n)$ that specifies the number of seed nodes (e.g.,

the first batch of people who have been convinced to adopt a new product), find the seed set of K nodes to be activated first so that they can trigger the largest cascade of further adoptions in the social network. More precisely, let S_0 denote a seed set, $\phi(S_0)$ denote the final active set generated by the stochastic diffusion model, and $\sigma(S_0) = E(|\phi(S_0)|)$ denote the *influence spread* of seed set S_0, which is the expected size of the final active sets of all random runs of influence diffusion under the given diffusion model. Then the influence-maximization problem is to find the optimal seed set S^* such that

$$S^* = \underset{S_0 \subseteq V,\, |S_0|=K}{\arg\max}\ \sigma(S_0).$$

The influence-maximization (hereafter referred to as IM) problem is NP-hard for both the IC and LT models. Kempe et al. [69] prove the hardness by considering an instance of the NP-complete *Set Cover* problem and the NP-complete *Vertex Cover* problem as a special case of the IM problem under the IC model and the LT model, respectively. Moreover, they show that the influence spread function $\sigma(\cdot)$ in both the IC and LT models has two important properties: *monotonicity* and *submodularity*. A set function $f : 2^V \to R$ is monotone if for any subsets $S \subseteq T \subseteq V$, $f(S) \leq f(T)$, which in this context means that adding more nodes to a seed set cannot reduce the size of the final activated set. The set function f is submodular if for any subsets $S \subseteq T \subseteq V$ and any element $u \in V \setminus T$, $f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T)$. Submodularity can be understood as diminishing marginal return, which in this context means the marginal contribution of node u when added to a seed set T cannot exceed the marginal contribution when adding it to a subset $S \subseteq T$. They prove these properties by formulating an equivalent view of the diffusion process in terms of *live-edge graph models*. Further, it is shown that the influence spread function $\sigma(\cdot)$ for any general cascade model or general threshold model is always monotone, but the general models may not satisfy the submodularity property [20].

Fortunately, as conjectured by Kempe et al. [69] and later proved by Mossel and Roch [93], the following theorem holds: in the general threshold model, if the threshold function $f_v(\cdot)$ is submodular for all $v \in V$, then the influence spread $\sigma(\cdot)$ of the model is also submodular. This theorem not only reveals that local submodularity is preserved globally under the general threshold model, but also provides an alternative approach to examining the submodularity property of a broad class of the general threshold models.

After exploiting these properties, Kempe et al. present a simple greedy hill-climbing algorithm to find the *"optimal"* K-node seed set. As it relies on Monte Carlo (MC) simulation to compute the influence spread, we refer to it as MC-Greedy hereafter. It is easy to implement. As shown in Algorithm 4, MC-Greedy starts with an empty seed set (Line 2), and runs K rounds to generate a seed set of K seed nodes. In each round, it sweeps over each node $u \in V \setminus S$ to compute the influence spread $\sigma(S \cup \{u\})$. Since the exact computation of influence spread under the IC and LT models is $\sharp P$-hard [22], they run MC simulation R times to estimate the influence spread (Line 6-10) . Finally, the node that together with current seed set S generates the maximum influence spread is added to S (Line 12).

For any model (such as the IC and LT models) in which the influence spread function $\sigma(\cdot)$ is monotone and submodular, MC-Greedy approximates the optimal within a factor of $(1 - 1/e - \varepsilon)$ for any $\varepsilon > 0$ [96]. This is a performance guarantee at least 63% of optimal. However, this algorithm suffers from a very high computational cost. It requires $O(nK)$ evaluations of influence spread, and each evaluation needs to run R simulations of the diffusion process. Not only is each simulation generally time-consuming, but R also has to be large enough to maintain the effectiveness of this greedy algorithm. This performance inefficiency makes it infeasible even for medium-sized networks of tens of thousands of nodes and edges.

To improve its efficiency, one strategy is to reduce the number of influence-

Algorithm 4 MC-Greedy

```
1: procedure MC-Greedy(G = (V, E), K, R)
2:     Initialize S ← φ
3:     for i = 1 to K do
4:         for each node u ∈ V \ S do
5:             g_u ← 0
6:             for j = 1 to R do
7:                 σ_u ← MC-simulation(S ∪ {u})
8:                 g_u += σ_u
9:             end for
10:            g_u ← g_u/R
11:        end for
12:        S ← S ∪ {arg max_{u∈V\S} g_u}
13:    end for
14:    Return S
15: end procedure
```

spread evaluations. Leskovec et al. [81] exploit the submodularity and present a *Cost-Effective Lazy Forward* (CELF) scheme that significantly reduces the number of evaluations. The key idea is that a node u's marginal gain $\sigma(u|S)$ in the current iteration cannot be more than its marginal gain $\sigma(u|S')$ in any previous iteration since $\sigma(\cdot)$ is submodular and $S' \subset S$. Thus, if we have evaluated a node w's marginal gain $\sigma(w|S)$ in current iteration, there is no need to evaluate in this iteration any node u that was evaluated in previous iteration with $\sigma(u|S') \leq \sigma(w|S)$. It is shown empirically that CELF improves the time-complexity of MC-Greedy by up to 700 times. Goyal et al. [50] propose CELF++ to further reduce the number of evaluations, which is approximately 35-55% faster than CELF as reported by the authors. Unfortunately, CELF and CELF++ are still quite slow and not scalable due to the overheads of Monte Carlo simulation.

Another way to improve the efficiency is to develop new heuristic algorithms that avoid Monte Carlo simulation. This strategy requires exploiting specific aspects of the diffusion model and the network structure. Wang et al. [125] propose a *maximum influence arborescence* (MIA) algorithm for the IC model. The main idea is to use local

arborescence structures of each node to approximate the influence propagation. They first compute the *maximum influence path* (MIP) between each pair of nodes (u, v), which is the path with the maximum influence probability among all paths from u to v. A user-specified threshold is used to ignore MIPs with too small probabilities. Then they union the MIPs starting or ending at each node into the arborescence structures, which represent the local influence region of each node. They only consider the influence propagation within these local regions. Their experiments show that MIA is several orders of magnitude faster than greedy algorithms with competitive influence spread. Chen et al. [22] use the same approach and propose a *local directed acyclic graph* (LDAG) algorithm for the LT model, in which they construct a LDAG for each node and restrict influence diffusion to a node only through its LDAG. They show that LDAG improves running time by three orders of magnitude while closely matching the influence spread of the greedy algorithm. Goyal et al. [51] present a more efficient heuristic called SIMPATH for influence maximization under the LT model. This algorithm is built upon the CELF optimization that iteratively selects seeds in a lazy forward manner. However, they avoid the expensive MC simulation by enumerating the *simple paths* starting from the seed nodes within a small neighborhood. They show that SIMPATH outperforms LDAG in terms of running time, memory consumption and influence spread.

There are some other interesting algorithms. An appealing idea is to take advantage of the community structures in social networks. Wang et al. [130] propose a *community-based greedy algorithm* (CGA) for finding the top-K influential nodes in mobile social networks under the IC model. They first detect communities, and then implement a dynamic programming algorithm for selecting communities to mine influential nodes. Their experimental results show that CGA is more than an order of magnitude faster than MC-Greedy with small loss in influence spread. Jiang et al. [64] propose a new *simulated annealing* (SA) heuristic to maximize influence spread un-

der the IC model. They estimate the influence spread using *expected diffusion values* instead of running MC simulation, which significantly improve the efficiency. They show in their experiments that SA outperforms the state-of-the-art greedy algorithms in terms of both efficiency and influence spread.

4.3 Multiple-path Asynchronous Threshold (MAT) Model

In this section, we elaborate our MAT model for viral marketing. In this model, we quantitatively measure the aggregated influence on each node, rather than just differentiating the nodes with a binary status on whether they are activated. Our model integrates direct and indirect influence, and instantiates the 3A process to model complex WOM communication and its effects. Moreover, we explicitly model influence attenuation along diffusion paths, influence decay with time, and individual temporal diffusion dynamics related to the relationship strength or contact frequency between a node of interest and its neighbors.

4.3.1 Model Description

Our model is applicable to both undirected/directed and unweighted/weighted networks. Without loss of generality, let $G = (V, E, W)$ be a directed and weighted network with $|V| = n$ and $|E| = m$. The weight $w_{uv} \in W$ associated with a directed edge $(u \rightarrow v)$ represents the relationship strength of node u over node v, e.g., the frequency that u calls or emails v. The relationship strength is usually asymmetric in directed networks. For any unweighted network, we regard it as a weighted network with a weight of 1 on each link. The influence is transmitted through out-links. For any undirected edge between two nodes, we replace it with a pair of directed links pointing to each other, associated with the same weight as the original weight on the undirected edge.

We categorize all the nodes in the network into two types: influencers and mes-

sengers. An *influencer* is an active node that originates and spreads its influence in the network. A *messenger* is an inactive node that acquires influence and passes the influence it receives from influencers or other messengers. Once a messenger acquires enough influence (greater than or equal to its threshold), it is activated and turns into an influencer who starts to spread out its own influence to others. It is noted that an influencer not only actively diffuses its influence to others but also acts as a messenger passing along the influence from other influencers or messengers. Our model captures an important characteristic of WOM communication in that: *anyone (either active or inactive) can pass along WOM messages and potentially influence the recipient*. In other words, a node can be activated by not only *direct influence* from its active neighbors (influencers) but also *indirect influence* passed along by its inactive neighbors (messengers). This is a distinguishing feature built in our MAT model. In addition, there are two important mechanisms implemented in the MAT model. First, *cycles are prohibited*. It makes sense since no one would repeatedly send out the same message to her friends. Second, *an influencer may influence the same person multiple times* since her message may be delivered to that person along different diffusion paths. This is a more realistic imitation of WOM messaging and social reinforcement in social networks.

For example, in Figure 4.5, A is an influencer. She bought an iPhone at $t = 0$ and told her friends that it was awesome at $t = 1$. In Figure 4.5(a), B got the message and direct influence from A. Suppose that B sent the message back to A at $t = 2$. A would simply ignore it. Even if A is inactive, the message would have no influence on A since it was exactly the message she sent out. It is sensible to avoid such cycles. In Figure 4.5(b), when A passed her message on to B and C at $t = 1$, B and C acquire the direct influence from A. Further, when B told C that *"I have a friend who bought an iPhone and she thought it is so cool"* at $t = 2$, C receives A's indirect influence via B (triadic transitivity). Similarly, in Figure 4.5(c) and (d), both D and

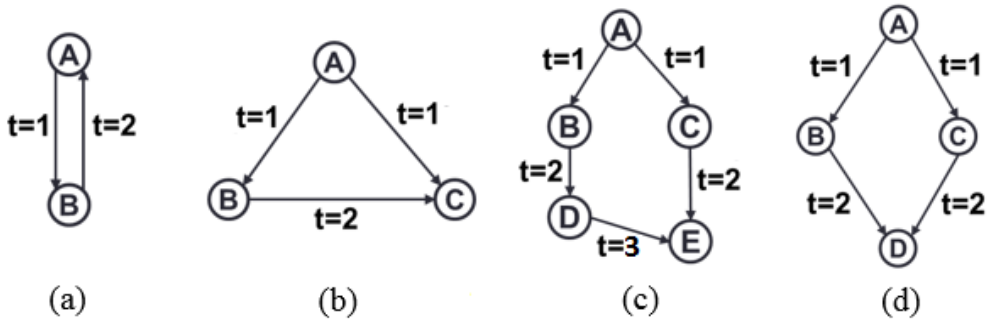

Figure 4.5: Several types of direct and indirect influence

E received A's indirect influence from multiple paths. Especially, in Figure 4.5(d), D is structurally equivalent to A. Our MAT model captures the well-known *structural-equivalence* property, which may be crucial to social contagions [37]. This model fits well in a variety of scenarios, such as Twitter/consume/citation networks.

The relationship strength of a person, as indicated by the weight on edges, may vary significantly among family members, close friends, work colleagues, casual acquaintances, and so on. Stronger relationship implies stronger influence in general. To quantify the relationship strength on influence, we introduce a *weight normalization scheme* that measures the fraction of influence a node receives from a specific in-neighbor relative to the total influence it may receive from all of its in-neighbors. Given a directed edge $u \to v$ with a raw weight w_{uv} , and letting N_v^{in} denote the set of node v's in-neighbors, the normalized weight $\hat{w}_{uv}$ is defined as

$$\hat{w}_{uv} = \frac{w_{uv}}{\sum_{k \in N_v^{in}} w_{kv}}.$$

As discussed in the introduction, influence does not remain static or constant, but rather dissipates rapidly along diffusion paths [24, 87] and decays with time [49]. To quantify the influence attenuation along a diffusion path, we use a depth-associated attenuation coefficient $\alpha = d^{-2}$, where d is the depth (number of hops) from an influ-

encer to the node of interest along the diffusion path. It is the same as what we define in Chapter 2, and can be interpreted as a compounding factor that incorporates the trustworthiness decay, information corruption, and decreasing reaching probability. Specifically, we set the depth limit d_{max} to 3 (for each influencer) to capture the *three-degrees-of-influence* phenomenon. On the other hand, we model the temporal influence decay as an exponential function of time

$$I(t) = e^{-\lambda t}$$

where λ is a user-specified tunable parameter of decay rate. It can be used to account for different products/topics or various types of social media, such as blogs, newspaper, TV, etc. When a node is newly activated, an independent timer is attached to this new influencer and the time is set to 0 at that moment. These mechanisms are proposed to reflect WOM communication in reality. Right after we purchase a new product, we feel the excitement and urge to tell our friends (potentially with stronger attempt to influence them), but our enthusiasm and influence potential fade away monotonically over time. It is noted that the timing for that node acting as a messenger of other influencers has no change.

In general, people communicate much more frequently with their family members and close friends than casual acquaintances. The contact frequency among friends is strongly correlated with the relationship strength. As a result, WOM communications generally take place in an asynchronous manner at the individual level. To capture the individual temporal diffusion dynamics, we model the heterogeneity of WOM messaging from one to her friends as a *Poisson process*. For a directed edge $(u \to v)$, let X_{uv} denote the number of times node u makes contact with node v during a unit interval of time. We assume X_{uv} follows a Poisson distribution with a rate of μ_{uv}. Its probability mass function can be written as

$$p_{uv}(x) = \begin{cases} \frac{\mu_{uv}^{x} e^{-\mu_{uv}}}{x!}, & x = 0, 1, 2... \\ 0, & \text{elsewhere.} \end{cases}$$

μ_{uv} is computed as

$$\mu_{uv} = 1 + \tilde{w}_{uv} + \mu_u,$$

where $\tilde{w}_{uv}$ and μ_u can be interpreted as the *local activeness* of $u \to v$ and *global activeness* of node u. Let $w_{u:max}$ and $w_{u:min}$ denote the maximum and minimum weight among node u's out-links, respectively. Let w_u denote the sum of all out-link weights of node u, and define $w_{max} = \max_{i \in V} w_i$ and $w_{min} = \min_{i \in V} w_i$. Then, we measure $\tilde{w}_{uv}$ and μ_u as follows

$$\tilde{w}_{uv} = \frac{w_{uv} - w_{u:min}}{w_{u:max} - w_{u:min}},$$

$$\mu_u = \frac{w_u - w_{min}}{w_{max} - w_{min}}.$$

These normalization schemes enable us to differentiate a node's activeness at both the local and global levels. If $w_{u:max} = w_{u:min}$ (e.g., unweighted networks), we set $\tilde{w}_{uv}$ to 0.5. In case $w_{max} = w_{min}$ (which rarely occurs), we set μ_u to 0.5 for all $u \in V$. Thus, μ_{uv} falls in the range $[1, 3]$ for any $(u \to v) \in E$. At each time step, the WOM message is propagated one hop from node u to its out-neighbor v with a probability

$$p_{uv} = 1 - P(X \leq 0).$$

If the propagation is not realized at time step t_i, then the probability for the message to be transmitted at time step t_{i+1} is unchanged due to the memoryless property of Poisson distribution. If u has delayed passing the message on to v for $delay_{max}$ time

steps (*maximum delay*), it is assumed that u has no intention to pass the message on to v at all. This is a sensible mechanism reflecting the fact that not everyone is actively engaged in WOM messaging with each of its neighbors at any time steps. In addition, we set $delay_{max}$ to 3 time steps in alignment with the depth limit d_{max} of 3, which implies that the message would have no noticeable influence even if it is finally propagated after it has been held for 3 or more time steps.

Now we can quantify the influence along a diffusion path at a specific time step. Let $\sigma_{i:x\to y}^{t,d}$ denote the amount of influence (originated from an influencer i) that node x passes on to node y at time step t and depth d. Then following a diffusion path from an influencer $i \to j \to k \to l$, the influence that nodes j, k and l acquire from node i is respectively calculated as

$$\sigma_{i:i\to j}^{t_1,1} = e^{-\lambda t_1} \times \frac{1}{1^2} \times \hat{w}_{ij}$$

$$\sigma_{i:j\to k}^{t_2,2} = e^{-\lambda t_2} \times \frac{1}{2^2} \times \hat{w}_{ij} \times \hat{w}_{jk}$$

$$\sigma_{i:k\to l}^{t_3,3} = e^{-\lambda t_3} \times \frac{1}{3^2} \times \hat{w}_{ij} \times \hat{w}_{jk} \times \hat{w}_{kl}.$$

In each equation above, the first term on the right hand side is the temporal influence decay, the second term is the depth-associated influence attenuation, and the rest is the chain product of the normalized weights on the corresponding links that constitute the diffusion path from the influencer to the node of interest. It is noted that since we allow a delay of propagation for up to 3 time steps for each node, t_1 can take any element in $\{1, 2, 3\}$, t_2 in $\{2, 3, 4, 5, 6\}$, t_3 in $\{3, 4, 5, 6, 7, 8, 9\}$, and $t_1 < t_2 < t_3$.

The diffusion process starts with an initial set of influencers (seed nodes) S_0 with $|S_0| = K$, and unfolds in discrete time steps. At each time step, the influence is propagated one hop from a node u (parent node) to each out-neighbor v (child node) with a probability p_{uv}. Like the IC and LT models, we focus on the *progressive case* in which an inactive node can turn into an active node but not the other way

around. Each seed node is assigned an influence value of 1. Each inactive node v is initialized with an influence value of 0, and selects an activation threshold θ_v uniformly at random in the range $[0, 1]$. Let A denote the set of v's in-neighbors who pass influence on to v, and let $f_v(A)$ denote the threshold function of v, that is, the total influence that v receives from nodes in A. Formally, we define $f_v(A)$ as

$$f_v(A) = \min(1, \sum_{u \in A} b_{uv}), \quad A \subseteq N_v^{in},$$

where b_{uv} represents the amount of influence that v receives from u, and N_v^{in} denotes the set of in-neighbors of v. Whenever $f_v(A) \geq \theta_v$, v is activated and turns into an influencer with an influence value of 1. Then it not only continues passing other influencers' influence as a messenger, but also starts to spread out its own influence as an influencer. The diffusion process stops when the number of hops of influence diffusion of each influencer (including the seed nodes and all activated nodes) reaches the depth limit ($d_{max} = 3$ by default) and no new activation is possible.

4.3.2 Discussions

It is interesting to observe how the MAT model relates to the general threshold model and the classic LT model. The definition of the general threshold model can be described as follows: For a social network $G = (V, E)$, each node $v \in V$ has an *arbitrary* threshold function $f_v(A)$: $A \subseteq N_v^{in} \rightarrow [0, 1]$, subject to the condition that $f_v(\phi) = 0$ and $f_v(\cdot)$ is monotone. Initially, each node independently selects a threshold θ_v uniformly at random in the range $[0, 1]$. Starting from a given seed set S_0, at every time step $t \geq 1$, first set S_t to be S_{t-1}, then for any inactive node $v \in V$, if $f_v(S_{t-1} \cap N_v^{in}) \geq \theta_v$, v becomes activated at time step t, and add it into S_t. This diffusion process stops when no new activation is possible. The LT model is a special

case of the general threshold model, in which each threshold function is defined as

$$f_v(S \cap N_v^{in}) = \min(1, \sum_{u \in S \cap N_v^{in}} b_{uv}),$$

where b_{uv} is the pre-specified influence weight on link $u \to v$. Especially, for the LT model, the notation $S_{t-1} \cap N_v^{in}$ can be extended to N_v^{in} only by a simple transformation: for each link $(u \to v)$, define $b'_{uv} = b_{uv}$ if u is an active node and $b'_{uv} = 0$ if u is an inactive node.

As we can see, our MAT model can be regarded as a general LT model and a more general threshold model. The main generalization comes from the following two directions. First, the general threshold model considers only the active in-neighbors who exert direct influence on the node of interest. The MAT model considers not only the active in-neighbors but also those inactive in-neighbors who serve as messengers passing on to the node of interest the indirect influence originated from influencers two or three hops away. In other words, we have $b'_{uv} > 0$ even if u is an inactive messenger. Second, b_{uv} in the LT model is a pre-specified constant. However, b_{uv} in the MAT model is a variable that depends on the diffusion path, temporal decay of influence, and individual diffusion dynamics. One way to think of it is that when u passes either direct or indirect influence on to v, the amount of influence is incrementally accumulated in influence weight b_{uv} and stored on the link $u \to v$. When checking whether v can be activated, its threshold function f_v linearly sums up the influence weights on all v's in-links. In particular, if we set the temporal decay rate $\lambda = 0$, the depth limit $d_{max} = 1$, and the diffusion probability $p_{uv} = 1$ if u is active and $p_{uv} = 0$ if u is inactive, our MAT model reduces to the classic LT model.

4.4 Influence Maximization under MAT Model

We formally define the influence-maximization (IM) problem under our MAT model in a general framework as follows: Given a social network $G = (V, E, W)$,

a budget K denoting the seed-set size, and the MAT model on G associated with parameters λ (temporal decay rate), d_{max} (depth limit), p_{uv} for each $(u \to v) \in E$ (individual diffusion probability) and $delay_{max}$ (maximum delay), find a seed set $S_0 \subseteq V$ with $|S_0| \leq K$ such that the expected influence spread $\sigma(S_0)$ is maximized under the MAT model. For viral marketing, the seed nodes are the initial adopters of a new product, and the influence spread is measured by the total number of people in the network who adopt/purchase the product in the end of the diffusion process.

4.4.1 Complexity and Properties

We have thus far proposed the MAT model and defined the IM problem under the MAT model. Now we show below the problem is NP-hard and the influence spread function $\sigma(\cdot)$ is monotone. It is desirable to show that $\sigma(\cdot)$ is also submodular such that the greedy hill-climbing algorithm provides a $(1 - 1/e - \varepsilon)$-approximation to the optimum for any $\varepsilon > 0$. However, it turns out that it is hard to rigorously prove it. We present it as a conjecture.

Theorem 4.4.1. The IM problem is NP-hard under the MAT model.

Proof. Consider the class of instances of the problem with restrictions that $\lambda = 0$, $d_{max} = 1$, $delay_{max} = 0$, and $p_{uv} = 1$ if u is active and $p_{uv} = 0$ if u is inactive for any $u \in V$. As discussed in previous section, the MAT model then reduces to the classic LT model. Therefore, the IM problem over this class of instances is equivalent to the classic IM problem under the LT model, which is known to be NP-hard [69]. This concludes the proof. □

Theorem 4.4.2. The influence spread function $\sigma(\cdot)$ for the MAT model is monotone.

Proof. Recall that the threshold function $f_v(S) = \min(1, \sum_{u \in S \cap N_v^{in}} b_{uv})$, where b_{uv} is the amount of influence that v receives from u. It is straightforward to verify that $f_v(S)$ is monotone. For each run of the diffusion process, the threshold θ_v is randomly

selected and fixed for each $v \in V$. Then it is easy to show that when the seed set S_0 grows, the final active set $\phi(S_0)$ also grows under these fixed thresholds, due to the monotonicity of $f_v(S)$. Finally, $\sigma(S_0)$ is simply the average of the size of all final active sets among all possible threshold values selected in different runs, and thus is monotone. □

Conjecture 4.1. *The influence spread function $\sigma(\cdot)$ for the MAT model is submodular.*

Submodularity is a desirable property in the influence-maximization problem. With this property, the greedy hill-climbing algorithm produces a solution that has approximation factor $(1 - 1/e)$ of the optimal. However, it does not hold for all diffusion models. Even if it does for some model (such as the IC and LT models), the proof is nontrivial.

The most important and widely-used approach is to construct an equivalent *live-edge graph model* [20, 69] to the diffusion model of interest. A live-edge graph is generated as follows: Given a graph $G = (V, E)$, each edge $e \in E$ is marked as either *live* or *blocked* based on certain randomized rule, and the subgraph consisting of all nodes in V and all live edges is called a live-edge graph. Two stochastic diffusion models for a social network $G = (V, E)$ are equivalent if for any given seed set $S_0 \subseteq V$, for any time step $t \geq 1$ and any subset $A_1, \ldots, A_{t-1} \subseteq V$, the event $S_1 = A_1, \ldots, S_{t-1} = A_{t-1}$ has either zero probability or non-zero probability in both models, and in the latter case, the conditional distributions of active set S_t under seed set S_0 conditioned on the event $S_1 = A_1, \ldots, S_{t-1} = A_{t-1}$ are the same for the two models. In other words, when two models are equivalent, the joint probability distributions of all active sets $S_1, S_2, \ldots$ under any given seed set S_0 for the two models must be the same [20]. It is usually a challenging task to find a live-edge graph model equivalent to the diffusion model under consideration (and hard to prove the equivalence). An important feature embedded in the live-edge graphs is that any active node should

be reachable from at least one seed node along at least one *live-edge path* consisting entirely of live edges. It implies that any active node (except seed nodes) should have at least one active neighbor, which makes it infeasible to apply this approach to our MAT model since a node without any active neighbors may still be activated by indirect influence in the MAT model.

The theorem conjectured by Kempe et al. [69] and later proved by Mossel and Roch [93] provides an alternative approach to examining the submodularity property of the general threshold models. The theorem states that if the threshold function $f_v(\cdot)$ is submodular for all $v \in V$, then the influence spread function $\sigma(\cdot)$ of a general threshold model is also submodular. As it manifests, it is only applicable to the general threshold model, in which $f_v(\cdot)$ depends only on active neighbors. In our MAT model, $f_v(\cdot)$ considers both active neighbors (influencers) and inactive neighbors (messengers) as long as they pass influence on to v. It is possible to circumvent the obstacle by including those messengers in the active set as *dummy* influencers. It is needed to aggregate and store their influence on the corresponding links, and keep track of the influence each messenger receives. At some time step, a messenger might be activated if the influence it receives is greater than or equal to its threshold. In the end of the diffusion process, we remove those *dummy* influencers who are not activated. This might be a promising direction, but it requires rigorous proof. For now, we conjecture that the influence spread function $\sigma(\cdot)$ for our MAT model is submodular, and leave the proof as future work.

4.4.2 Algorithms

The final step to address the IM problem is developing effective and efficient algorithms to find the "optimal" seed set. Intuitively, the ideal selection of the K seed nodes would be the K most influential nodes such that: each of them achieves individual influence spread as large as possible, and they should be far away from each other to minimize the potential overlaps, but close enough to result in aggregation effects.

We develop a set of six approximation algorithms with different seeding strategies for influence maximization under our MAT model, which include four generic baseline heuristics and two novel algorithms designed specifically for the MAT model.

4.4.2.1 Baseline Heuristics

The simplest and naive baseline is to select nodes uniformly at random (hereafter referred to as RANDOM). Not surprisingly, this algorithm is very unstable and does not perform well in terms of influence spread. The most frequently used is the *degree-centrality* heuristic (hereafter referred to as DEGREE), in which the seed nodes v are chosen in descending order of out-degrees d_v^{out}. This seeding strategy is simple but effective. Empirical studies [57,69] show that DEGREE results in larger influence spread than other centrality-based heuristics, such as *distance centrality* and *betweenness centrality*. A common feature of these centrality-based algorithms is that they rely solely on one specific structural property of the network without considering the diffusion dynamics. For example, many of the highest-degree nodes may be clustered and have potentially large overlaps of influence spread, which leads to deterioration in performance.

The next baseline is the Top-K algorithm. We present its pseudocode in Algorithm 5. It sweeps over each node $u \in V$ to compute the influence spread of each node individually (Lines 2-9), using Monte Carlo simulation (Lines 4-7). Then, it selects the top K nodes with the largest *individual* influence spread (Lines 10-13). It is worth noting that the top-K nodes that produce the largest influence spread *individually* is not the same as the K seed nodes that produce the largest influence spread *together*. For example, if two top influencers are so close to each other that their influence spreads have a large overlap, it is not a good idea to select both of them as seed nodes. While the DEGREE algorithm relies solely on the structural properties of the network without considering the diffusion dynamics, the Top-K algorithm relies solely on the diffusion dynamics without considering the network structure.

Algorithm 5 Top-K

```
1: procedure TOP-K(G = (V, E), K, R)
2:     for each node u ∈ V do
3:         g_u ← 0
4:         for j = 1 to R do
5:             σ_u ← MC-simulation(S ∪ {u})
6:             g_u ← g_u + σ_u
7:         end for
8:         g_u ← g_u/R
9:     end for
10:    S ← φ
11:    for i = 1 to K do
12:        S ← S ∪ {arg max_{u∈V\S} g_u}
13:    end for
14:    Return S
15: end procedure
```

MC-Greedy (as shown in Algorithm 4) is the last baseline heuristic. It considers both the diffusion dynamics and the network structure (implicitly). One may notice that the Top-K algorithm is actually the first round of the K rounds in MC-Greedy except that MC-Greedy selects only one top influencer instead of K top influencers. It continues exhaustively running MC simulation to find the next seed node that produces the maximum *marginal gain* in influence spread. MC-Greedy usually achieves the largest influence spread of the K seed nodes at the collective level. However, as discussed in Section 4.2.2, It suffers from a very high computational cost, which makes it infeasible even for medium-sized networks.

4.4.2.2 IV-Greedy

MC-Greedy demonstrates the best performance in terms of influence spread under the IC and LT models among all state-of-the-art algorithms. Regardless of its low efficiency due to the overhead of MC simulation, the *greedy* strategy used in MC-Greedy makes sense and is highly effective. We develop IV-Greedy borrowing the greedy strategy of MC-greedy but replacing MC simulation by using the *influence vector* of each node, which greatly improve efficiency.

In our MAT model, we set $d_{max} = 3$ and $delay_{max} = 3$ by default and treat them as built-in parameters. The individual diffusion probability p_{uv} is determined by the weight on edges. The only user-specified parameter is the temporal decay rate λ, which is tunable and can be used to account for different products, topics and events on different types of social media. However, we do not want to include it in our influence-maximization algorithm so that our algorithm is robust and fairly comparable with other algorithms. Therefore, we directly call InfluenceMatrix-Builder (see Algorithm 1; IM-Builder for short hereafter), i.e., the algorithm we developed to generate the influence matrix for community detection in the *reachability-based* diffusion model. The only difference exists in the weight normalization schemes. In the reachability-based diffusion model, we normalize each in-weight of a node by the *maximum* in-weight of that node, which captures the *relative susceptibility* of the node to its in-neighbors. In our MAT model (*activation-based* diffusion model), we normalize each in-weight of a node by the *total* in-weight of that node, which quantifies the fraction of influence diffused from each in-neighbor. In fact, IM-Builder can be regarded as a *static* version of the influence-diffusion process, in which it explores various diffusion paths of each node individually, but ignores the temporal diffusion decay, individual diffusion dynamics, and the aggregate effect of multiple influencers and the newly activated nodes. We use it as a proxy for the real diffusion process so as to avoid the expensive MC simulation.

The influence vector of a node captures where and how much influence it spreads over its neighborhood. We define the *influence score* of a node as the sum of all elements in its influence vector, which represents an estimate of its total influence spread. Similar to the influence centrality we defined for community detection in Chapter 2, this influence score can be used to differentiate the influence significance of the nodes in the network. An intuitive solution is to select the top-K nodes of the highest influence scores as the seeds. Unfortunately, this solution does not work well.

Like those high-degree nodes, the nodes of high influence scores may be clustered or too close to each other, leading to large overlaps of influence spread. We need to somehow separate them from each other to minimize overlaps. The strategy that we use in IV-Greedy is the greedy strategy used in MC-Greedy, in which we sweep over the influence vector of each node to repeatedly pick the node with the maximum marginal gain and add it to the seed set until all K seeds are found. The pseudocode is shown in Algorithm 6.

Algorithm 6 IV-Greedy

procedure IV-GREEDY($G = (V, E, W), n = |V|, d_{max}, K$)
  $S \leftarrow \phi$
  Call IM-Builder(G, n, d_{max}) to generate IV_u of each node u
  for each node $u \in V$ **do**
    $R_u \leftarrow \sum_j IV_u(j)$
  end for
  $v \leftarrow \arg\max_{u \in V} R_u$
  $S \leftarrow S \cup \{v\}$
  $AR \leftarrow IV_v$
  for $k = 2$ to K **do**
    for node $i = 1$ to n and $i \in V \setminus S$ **do**
      $g_i \leftarrow 0$
      $c \leftarrow 1 - AR(i)$
      for $j = 1$ to n **do**
        $p \leftarrow c \times IV_i(j)$
        $q \leftarrow p + AR(j)$
        if $q > 1$ **then**
          $p \leftarrow 1 - AR(j)$
        end if
        $g_i \leftarrow g_i + p$
      end for
    end for
    $v \leftarrow \arg\max_{i \in V \setminus S} g_i$
    $S \leftarrow S \cup \{v\}$
    $c \leftarrow 1 - AR(v)$
    for $j = 1$ to n **do**
      $AR(j) \leftarrow \max(1, AR(j) + c \times IV_v(j))$
    end for
  end for
  Return S
end procedure

In IV-Greedy, each node is indicated by its node index from 1 to n. IV-Greedy starts with an empty seed set (Line 2), and then calls IM-Builder to generate the influence vector IV_u for each node $u \in V$. IV_u is an n-element array, in which element $IV_u(j)$ represents the amount of influence node u exerts on node j. For each node u, we get its influence score R_u by summing up all elements in its influence vector IV_u (Lines 4-6). We find the node v that has the highest influence score (Line 7), and add it to the seed set S as the first seed (Line 8). Then we make a copy of IV_v to an array AR, which is used as a representation of the *collective* influence distribution of currently selected seeds. To find the next seed, we sweep over each node $i \in V \setminus S$ and compute the marginal gain g_i of adding i to S individually. We compute its *benefit factor* $c = 1 - AR(i)$ (Line 13). Recall that $AR(i)$ represents the amount of influence i receives so far, which can be also regarded as the probability activating i. For example, if $AR(i) = 1$, it means that i has been activated, then selecting i as a seed has no benefit to the overall influence spread. Then for each element $IV_i(j)$, we get $p = c \times IV_i(j)$ (Line 15), which represents the marginal gain that node j gets. However, remember that the threshold function $f_v(S)$ is defined within the range $[0, 1]$. If the amount of influence that node v receives is greater than 1, it is set to 1 since v has been activated and more influence on v is of no more effect or benefit. Therefore, we add p to $AR(j)$ to get q (Line 16), which represents the accumulated influence that node j receives. If q is greater than 1, we set the actual marginal gain $p = 1 - AR(j)$ (Lines 17-19). Then we add p to g_i, which represents the total marginal gain of adding node i to S. Once we get the marginal gain of each node $i \in V \setminus S$ (Lines 11-22), we select the node that produces the maximum marginal gain as a seed and add it to S (Lines 23-24). Then we update AR by calculating the benefit factor c and adding the marginal gain to each element in AR (Lines 25-28). We repeat the process described above $(K-1)$ times to find the $(K-1)$ seeds, which along with the first seed node constitute a full seed set S.

4.4.2.3 IV-Community

The network structure is a crucial factor in influence diffusion. It is well studied and shown that the hubs have higher influential power. Moreover, community structure is a prominent property of social networks, and strong communities facilitate *internal* influence diffusion. It makes sense since people within a community are more densely connected, and thus have higher rate of WOM messaging and influence spread on each other. On the other hand, community structure may hinder WOM communication and influence diffusion between communities [34]. Therefore, an effective seeding strategy should consider the underlying community structure of the network, including the size and cohesiveness of individual communities.

We propose a community-based algorithm for influence maximization under our MAT model. The basic idea is to first identify the communities and then select the most influential nodes in different communities as the seed nodes. It seems straightforward at first glance, but turns out to be much more complicated than our initial thought. Intuitively, we can directly use our IGLP algorithms to find the community hierarchy and split the dendrogram at the level of K separated communities, and then select the most influence node in each community to form the K-node seed set S. Unfortunately, it does not work well since the K communities may vary significantly in size and cohesiveness. In addition, it does not work for those networks whose number of communities is less than K. Therefore, we have to define an appropriate community-level measure such that we can use it to allocate the "*right*" number of seeds to different communities. Ideally, this measure should incorporate both the size and the cohesiveness of the communities. In fact, if we have that measure, there is no need to agglomeratively merge the communities. We can directly allocate the seeds to the communities in the initial community configuration produced by the IGLP algorithm since the initial configuration captures the most compact community structure. Another benefit is that we do not have to do hierarchical clustering and thus reduce

the time complexity. There is another issue that needs to be addressed. When we need to select a seed node in a community, what is the measure we can use to identify the *most influential* node in that community?

Our strategy is to use the influence score as described in IV-Greedy subsection. Recall that the influence vector IV_u for each node $u \in V$, captures where and how much influence node u spreads in its neighborhood. Its influence score R_u is the sum of all elements in IV_u. It is straightforward to use the influence score as a measure to find the most influential node in each community. Further, we compute the influence score of a community C by summing up the influence scores of all of its community members, i.e.,

$$R_C = \sum_{u \in C} R_u.$$

R_C is not only directly defined in the context of influence but also naturally integrates both the size and the cohesiveness of the community. Then we define an *allocation protocol* as follows: *Allocate a quota of seed nodes to each community by the ratio of that community's influence score to the total influence score of all communities. If none of the communities is assigned to have even one seed (it is possible when the network consists of many small communities or the network is very large), allocate one seed to the community that has the largest influence score, and disregard it from future allocation.* When a community is assigned to have only one seed node, it is straightforward to select the node of highest influence score as the seed in that community. However, when we have to find two or more seed nodes in one community, we have to deal with the problem that is the same or as hard as the original one, just switching from the network level to a community level. Our solution is to use the same strategy as IV-Greedy to find the seed nodes using the influence vectors. We refer to this algorithm as IV-Community hereafter. A description of IV-Community is presented in Algorithm 7.

Algorithm 7 IV-Community

1: **procedure** IV-COMMUNITY($G = (V, E, W), n = |V|, d_{max}, K$)
2: $S \leftarrow \phi$
3: Call IM-Builder(G, n, d_{max}) to generate IV_u of each node u
4: **for** each node $u \in V$ **do**
5: $R_u \leftarrow \sum_i IV_u(i)$
6: **end for**
7: Call IGLP-WE(G, d_{max}) to find communities $\mathbb{C} = \{C_1, \ldots, C_l\}$
8: **for** each community $C \in \mathbb{C}$ **do**
9: $R_C \leftarrow \sum_{u \in C} R_u$
10: **end for**
11: Allocate seeds to the communities in $\mathbb{C}$ according to the allocation protocol
12: **for** each community C that gets i ($i > 0$) seed(s) **do**
13: **if** $i = 1$ **then**
14: $v \leftarrow \arg\max_{u \in C} R_u$
15: $S \leftarrow S \cup \{v\}$
16: **else**
17: Use IV-Greedy strategy to find $s = \{s_1, \ldots, s_i\}$
18: $S \leftarrow S \cup s$
19: **end if**
20: **end for**
21: Return S
22: **end procedure**

Similarly, IV-Community starts with an empty seed set (Line 2), and calls IM-Builder to generate the influence vector for each node (Line 3). The influence score of each node is calculated as the sum of all elements in its influence vector (Lines 4-6). Then we call IGLP-WE to find the communities in the initial configuration (Line 7). It is noted that we use IGLP-WE instead of IGLP-DP since we focus on finding strong communities in the initial configuration of the community structure of the network. For each community, we compute its influence score by summing up the influence scores of all members in that community (Lines 8-10). Then we allocate seeds to the communities using the allocation protocol (Line 11). If a community is assigned with a single seed, we select the node of highest influence score in that community, and add it to the seed set (Lines 14-15); if it has more than one seed, we use IV-Greedy strategy to find the seed nodes (Lines 17-18).

4.4.3 Experiments

We conduct experiments on four widely-used real-life network datasets to evaluate our MAT model and the performance of IV-Greedy and IV-Community, and compare them against the baseline algorithms on both influence spread and time efficiency. The code is written in Visual Basic and all experiments are carried out on a regular desktop PC with Intel(R) Core(TM) i5-4670 CPU @ 3.40 GHz and 8.0 GB memory under Windows 7 64-bit OS.

4.4.3.1 Network Description

To evaluate the applicability of our model and algorithms, we employ four real-life networks with different combinations of link directionality and weights. We list their statistics in Table 4.1. **PGP** [10] is an undirected/unweighted network of users of the Pretty-Good-Privacy algorithm for secure information interchange. Each node represents a user, and each edge connects a pair of users of interest who have assigned public keys of another based on trust between them. It is a single connected component with relative clear community structure. **NetHEPT**[2] is an undirected/weighted collaboration network from the paper lists extracted from "High Energy Physics (Theory)" section of the e-print arXiv from 1991 to 2003. Each node represents a unique author, and each edge represents co-authorship of the two authors of interest, weighted by the number of papers they have co-authored. This network has been frequently used in previous work [18, 22, 51]. **WikiVote**[3] [82] is a directed/unweighted who-vote-whom network from Wikipedia. Nodes in the network represent Wikipedia users and a directed edge from nodes i to j represents that user i voted on user j. This link direction does not reflect the direction of influence flow. User i voting on user j is actually because j has influence on i, as analogous to that of citing paper i and cited paper j. Therefore, we reverse the link direction to reflect the actual influence flow. This

[2]http://research.microsoft.com/en-us/people/weic/graphdata.zip
[3]https://snap.stanford.edu/data/wiki-Vote.html

network has a giant component and a set of 23 small ones. The last network dataset is **C.elegans** [132, 134]. It is a directed/weighted neural network of the nematode worm C.elegans.

Table 4.1: Statistics of network datasets

Dataset	**PGP**	**NetHEPT**	**WikiVote**	**C.elegans**
Directed	No	No	Yes	Yes
Weighted	No	Yes	No	Yes
Num. of nodes	10,680	15,233	7,115	453
Num. of directed links	0	0	97,835	2,025
Num. of undirected edges	24,316	31,376	2,927	0
Avg. out-degree	4.6	4.1	14.6	4.5
Max. out-degree	205	64	457	145
Avg. weight	1	1.9	1	2.3
Max. weight	1	119	1	114
Num. of connected components	1	1,781	24	1
Avg. component size	10,680	8.6	296.5	453
Largest component size	10,680	6,794	7,066	453

4.4.3.2 Performance Comparison

We run experiments on the four network datasets to compare the performance of IV-Greedy and IV-Community against the four baseline algorithms in terms of influence spread and time efficiency. As discussed before, we set $d_{max} = 3$ and $delay_{max} = 3$, and treat them as built-in parameters of the MAT model. The only user-specified parameter is the temporal decay rate λ, which can be tuned to account for different products, topics or events on different types of social media. Its sensible range is $(0, 0.5]$. The larger λ, the faster temporal decay of influence. We will evaluate its effect on influence spread in next subsection. For now, we set it to 0.2 for

all algorithms. In addition, both MC-Greedy and Top-K needs to run Monte Carlo (MC) simulation R_1 times in each round of influence-spread estimation. Once an algorithm produces the seed set it finds, we also rely on MC simulation (R_2 times) to estimate the influence spread of that algorithm for comparison. In our experiments, we set both R_1 and R_2 to 1000. In particular, due to the extremely low efficiency of MC-Greedy, we only report its results for C.elegans dataset.

Influence spread. We illustrate in Figure 4.6 the experimental results on the four network datasets. Each curve shows the variation of the influence spread with respect to the seed-set size. It is not surprising that RANDOM has the worst performance on all datasets, which indicates selecting seed nodes at random is not a good idea. The degree-centrality heuristic, DEGREE, greatly improves the influence spread, which is 4-8 times better than RANDOM. It makes sense that the well-connected nodes (social hubs) facilitate influence diffusion with their high reach to others. Top-K is analogous to DEGREE in the sense of targeting the group of most influential nodes who achieve largest influence spread *individually*. It achieves an improvement of 26.7%, 4.6%, 6.5%, and 7.2% over DEGREE on PGP, NetHEPT, WikiVote, and C.elegans, respectively. However, its improvement on influence spread comes with a huge sacrifice on efficiency since it relies on expensive MC simulation. Both DEGREE and Top-K fail to capture the fact that many of the highest-degree or most influential nodes may be clustered, which results in large overlaps of influence spread. Our IV-Community takes advantage of the community structure to naturally separate the seed nodes from each other to minimize the potential overlaps. It consistently outperforms DEGREE on all datasets with the largest improvement of 27.9% on PGP. It is about 4.6% better than Top-K on NetHEPT, and exhibits some advantages over Top-K on WikiVote and PGP when the seed-set size is relatively large. But it cannot beat Top-K on C.elegans. IV-Greedy achieves the best performance overall in the comparison with IV-Community, Top-K, DEGREE and RANDOM. It outperforms IV-Community,

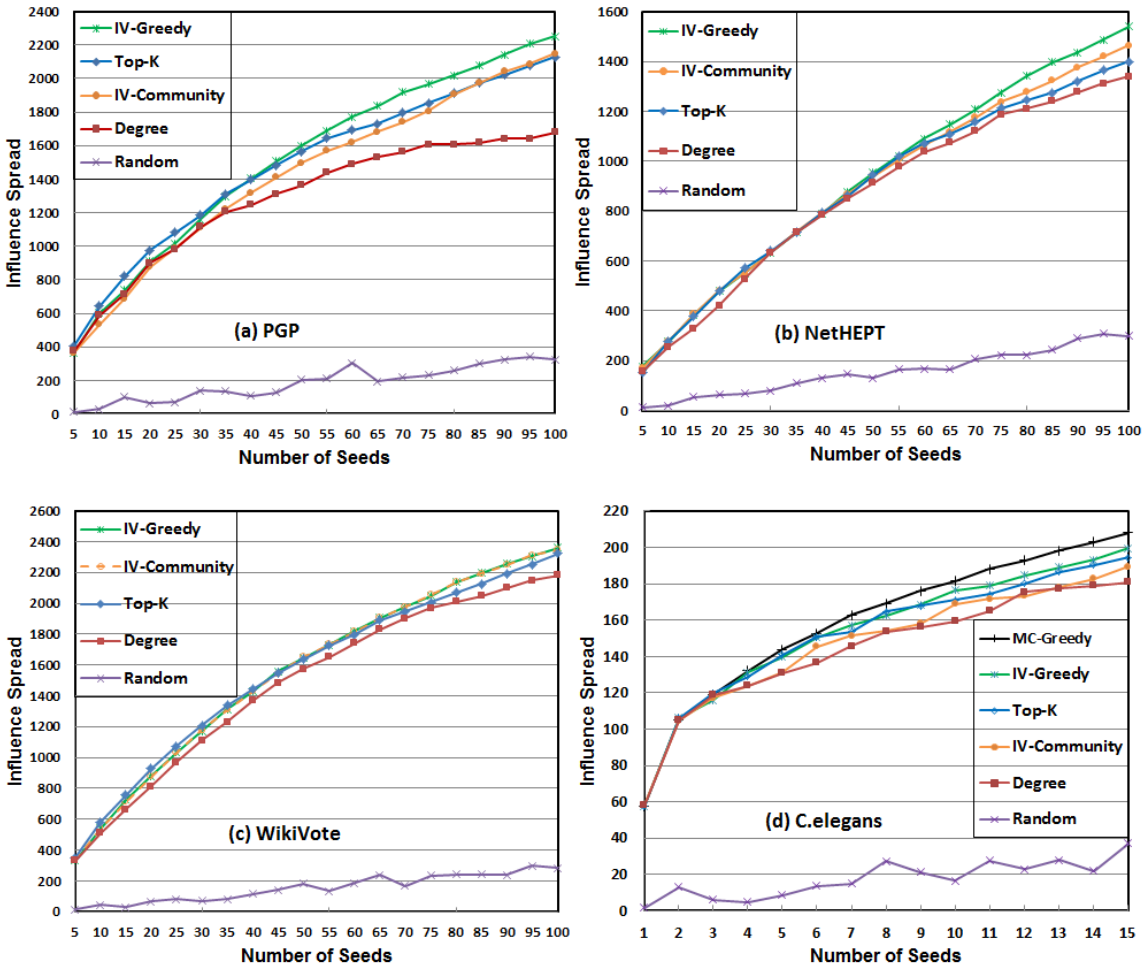

Figure 4.6: Performance comparison on influence spread

DEGREE and RANDOM consistently, except that it shows exactly the same performance as IV-Community on WikiVote. That is because IV-Community clusters all the nodes in the giant component into one single community and allocates all seeds to it. IV-Greedy achieves larger influence spread than Top-K when the seed set gets larger. It outperforms Top-K by 5.7%, 10.1%, 1.6%, and 3.1% on PGP, NetHEPT, WikiVote, and C.elegans, respectively. As expected, it is inferior to MC-Greedy, but only by 4% on C.elegans. However, as shown in the comparison of time efficiency, IV-Greedy and IV-Community significantly outperform MC-Greedy and Top-K.

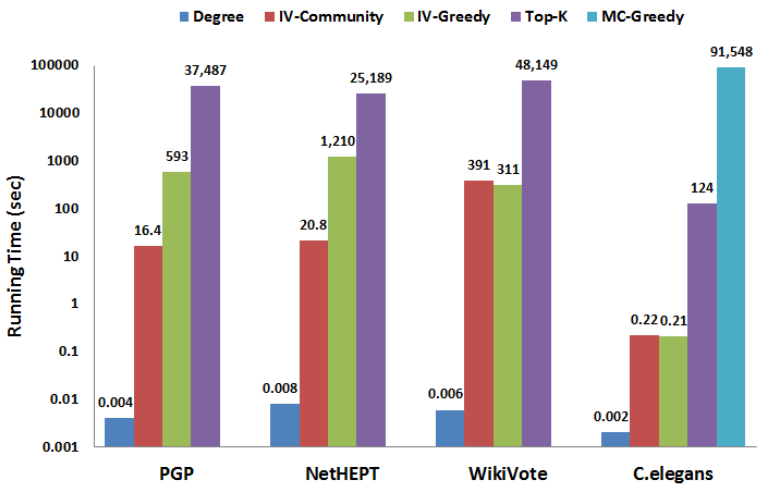

Figure 4.7: Performance comparison on running time (CPU seconds)

Running time. We start with a brief analysis of the time complexity of each algorithm. Let n and m denote the number of nodes and edges in the network, respectively. Each run of Monte Carlo (MC) simulation is $O(m)$, and thus Top-K has a time complexity of $O(nRm)$, where R is the number of repetitions of MC simulation. As we can see, Top-K already has a very high computational cost. Unfortunately, MC-Greedy is even much worse than Top-K. Top-K is just the first round in MC-Greedy finding the first seed. This process has to repeat K times to find the K-node seed set. Moreover, in the k^{th} repetition, the seed-set size increases to k. The time that each run of MC simulation in the k^{th} repetition takes is more than k times of that in the first round. Therefore, the time complexity of MC-Greedy is $O(K^2nRM)$, which makes it infeasible even for medium-sized networks of thousands of nodes.

IV-Greedy needs to generate an influence vector for each node, which takes $O(m)$ time as discussed in Chapter 2. We do not have to maintain each influence vector in an n-element array, but only keep the non-zero influence values in a compact dynamic array. Let L denote the average length of influence vectors. L is determined by the average node out-degree b, depth limit d_{max} and and the community structure. Then

in Line 11 of IV-Greedy (see Algorithm 6), the number of iterations can be reduced to L from n. Therefore, the time complexity of IV-Greedy is $[O(m) + O(KLn)]$. The worst case is $[O(m) + O(Kn^2)]$ when L is comparable to n or if we simply use an n-element array to store the influence vector. As for IV-Community, it has to generate the influence vectors and calculate the influence score of each node, which takes $O(Ln)$ time. Then it calls IGLP-WE to find the communities in the initial configuration of community structures, which takes $O(Lm)$ time. Finally, it applies IV-Greedy strategy to find seed nodes in communities with seed assignments. The best case is that each seed is assigned to a different community, which leads to a time complexity of $[O(Lm) + O(Ln) + O(KC)]$, where C is the average size of those communities with seed assignments. The worse case is that IGLP-WE clusters all nodes in the network into one single community, which leads to a time complexity of $[O(Lm) + O(KLn)]$ or $[O(Lm) + O(Kn^2)]$. In general cases, it is expected that IV-Community is more efficient than IV-Greedy.

In Figure 4.7, we illustrate the running time of the algorithms on the four network datasets (the Y-axis is in logarithmic scale). A seed-set size of K=100 is set for the PGP, NetHEPT, and WikiVote datasets. K is set to 15 for the C.elegans dataset. Clearly, DEGREE is the fastest algorithm. It finishes almost instantly on all datasets. IV-Community is the second fastest overall. It is 36 and 58 times faster than IV-Greedy on PGP and NetHEPT, respectively. It takes a little bit more time than IV-Greedy on WikiVote since all the seeds are assigned to the giant component as a single community. Both IV-Community and IV-Greedy are up to three orders of magnitude faster than Top-K. Not surprisingly, MC-Greedy is the slowest one. As we can see on the C.elegans dataset of only 453 nodes and 2025 edges, while IV-Greedy and IV-Community take only 0.2 seconds to find a set of 15 seed nodes, Top-K takes 2 minutes 4 seconds, but MC-Greedy takes more than 25 hours. Due to such a high computational cost, we are not able to run it on the other three datasets.

4.4.3.3 Adoption Rate

Marketers are concerned about not only the influence spread but also the adoption rate, which is usually measured by the number of adopters in a given time period. How fast the influence spreads is a critical factor for a viral marketing campaign, especially when there exist competing products in the same social network. It is interesting to observe the rate of adoption under our MAT model. We show in Figure 4.8 the influence spread over time on the four network datasets with IV-Greedy as the seeding strategy. The seed-set size is 100 for PGP, NetHEPT and WikiVote, and 15 for C.elegans. As we can see, the adoption rate shows similar pattern on all datasets. The influence diffuses at a high-speed rate at early stage, and the speed decreases monotonically. The influence spread reaches the saturation level in about 10 to 15 time steps. At $t = 5$, the influence spread arrives at 87%, 84%, 93.4%, and 98.3% of the maximum on PGP, NetHEPT, WikiVote, and C.elegans, respectively. At $t = 10$, this percentage increases to 97.4%, 97.3%, 99.5%, and 100% on PGP, NetHEPT, WikiVote, and C.elegans, respectively. It is observed that WikiVote and C.elegans achieve higher adoption rates than PGP and NetHEPT, which can be roughly explained by the high cohesiveness of WikiVote and the small network size of C.elegans. This follows our intuition. In practice, marketers need to carefully determine the period of a time step. It could be a day, a week, or a month, etc. The influence decay rate and individual contact frequency will vary accordingly.

4.4.3.4 Parameter Analysis

The temporal decay rate λ is the only user-specified parameter in our MAT model. The value of λ can be varied to account for different products, topics or events on different types of social media. For a particular application, it may be possible to use a data-driven approach to estimate the appropriate value of λ. We evaluate the impact of λ on influence spread using different decay rates under the MAT model. We also include the classic LT model as a baseline in the comparison. DEGREE is

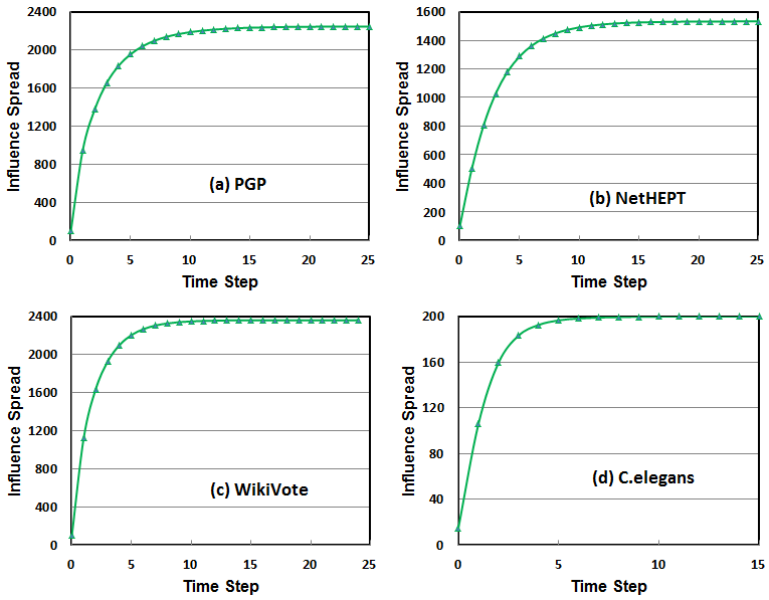

Figure 4.8: Adoption rate achieved by IV-Greedy

a fairly effective and generic heuristic that is applicable to both the MAT model and the LT model. Moreover, it always produces the same seed set, which enables us to perform a fair comparison on different models. Therefore, we use DEGREE in this experiment. The result is illustrated in Figure 4.9.

As expected, λ in our MAT model has significant impact on influence spread. A larger λ indicates faster temporal decay of influence, and thus results in smaller influence spread. It is good to see that λ in the MAT model enables us to gauge the influence spread in a reasonably large range on all datasets. When λ increase from 0.1 to 0.3, influence spread drops 45.4%, 48.1%, 36.7% and 25.4% on PGP, NetHEPT, WikiVote and C.elegans, respectively. Bhagat et al. [8] evaluate the influence-spread prediction of several models. Their findings suggest that the classic LT model over-estimates the influence spread by large amounts. As we can see, when $\lambda = 0.2$, the influence spread under the MAT model is already 15%, 24%, 21.3% and 19.9% smaller than the LT model on PGP, NetHEPT, WikiVote and C.elegans, respectively. When $\lambda = 0.3$, the influence spread under the MAT model drops to 35.4%, 43.2%, 36.4%

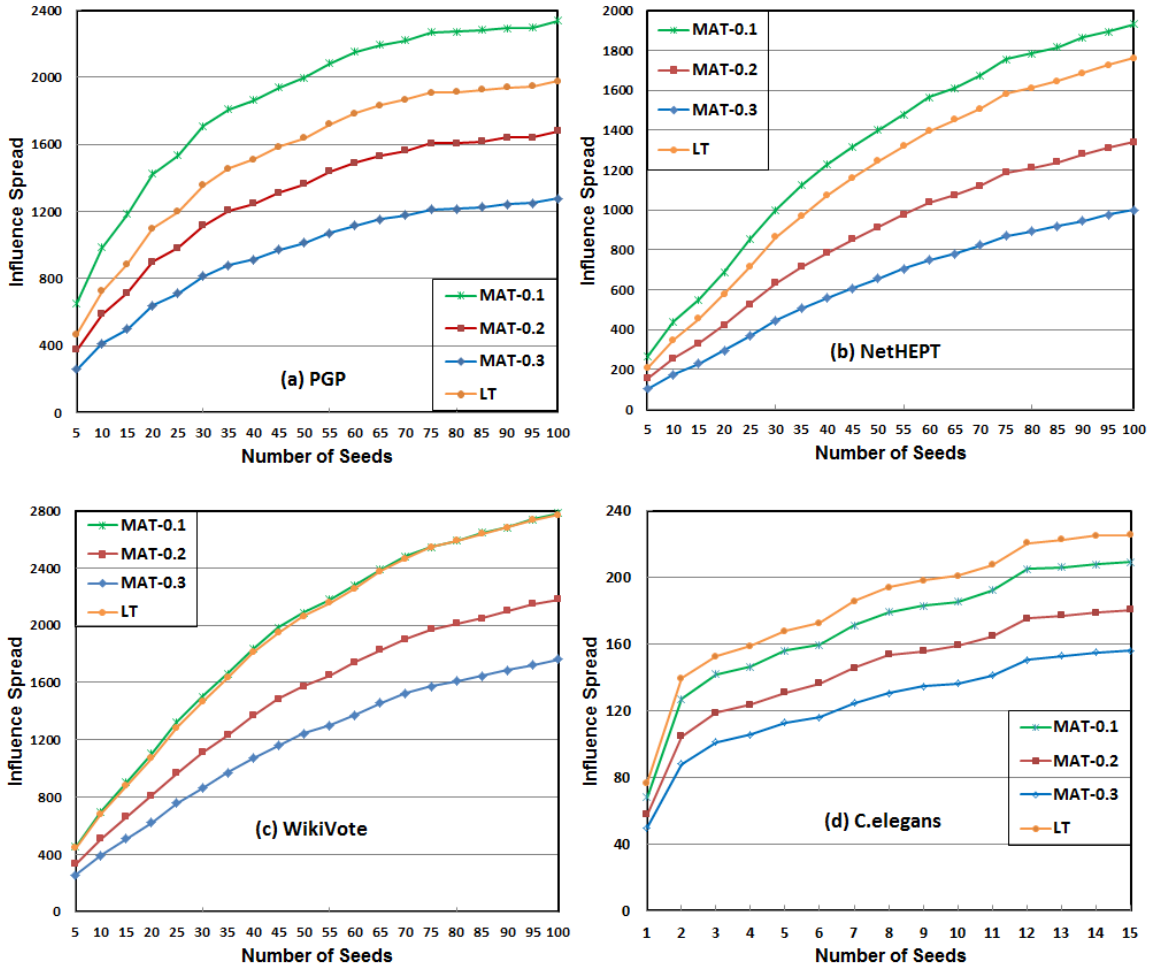

Figure 4.9: Influence spread achieved by DEGREE under the classic LT model and the MAT model with different temporal influence decay rates

and 31% smaller than the LT model on PGP, NetHEPT, WikiVote and C.elegans, respectively. We believe that our MAT model is more powerful than the LT model on prediction of influence spread. It is our future work to develop an effective data-driven approach to learn λ from training data and then use it in the MAT model to predict the diffusion.

4.5 Conclusion

In this chapter, we propose a novel *multiple-path asynchronous threshold* (MAT) model for viral marketing. It differs from existing diffusion models (such as the IC and LT models) in several aspects. We quantitatively measure influence and keep track of its diffusion and aggregation during the diffusion process. Our MAT model captures both direct and indirect influence, depth-associated influence attenuation, temporal influence decay, and individual diffusion dynamics. Our work is an important step toward a more realistic diffusion model. We prove that the influence-maximization problem under the MAT model is NP-hard, and show that the influence-spread function under the MAT model is monotone. Its submodularity property is left as a conjecture. Further, we develop two effective and efficient heuristics (IV-Greedy and IV-Community) to tackle the influence-maximization problem. Our experiments on four real-life networks demonstrate their excellent performance in terms of influence spread and efficiency. Our work provides preliminary but significant insights and implications for diffusion research and marketing practice.

This research opens up several directions for future work. First, the conjecture on submodularity of the influence-spread function under the MAT model needs rigorous proof. Second, scalable heuristics need to be developed for large-scale networks. Third, it is important to validate the MAT model with more real datasets from diverse domain. Last but not the least, it is desirable to develop an effective method to learn the temporal influence decay rate and use it in the MAT model for prediction of influence diffusion.

CHAPTER 5
CONCLUSION AND FUTURE WORK

In this dissertation, we investigate influence-diffusion modeling in networks for community detection, resilience analysis and viral marketing. First, we propose a novel *reachability-based* influence-diffusion model to decode the implicit knowledge of connectivity and proximity embedded in the network graph topology. Based upon this model, we define a new *influence centrality* and a novel *shared-influence-neighbor (SIN) similarity* as vertex-pair closeness metric. We then develop three influence-guided algorithms for community detection: *IGSK*, *IGLP-WE* and *IGLP-DP*. In particular, IGLP-WE and IGLP-DP render a complete community hierarchy and enable us to examine the community structure at different scales. They manifest high quality and promising scalability on both undirected/directed and unweighted/weighted networks. Second, we adapt the reachability-based diffusion model to topological resilience analysis of supply networks under random disruptions and targeted attacks. We propose a novel *resilience metric* by exploring the multiple-path reachability of each demand node. Further, we examine different attachment strategies and develop new *supply-network growth mechanisms* that enable us to build resiliency into the construction of supply networks. Finally, we propose a novel *activation-based* influence-diffusion model, *MAT model*, for viral marketing. It captures both direct and indirect influence, and incorporates influence attenuation along diffusion paths, influence decay with time, and individual diffusion dynamics. Moreover, we develop two effective and efficient approximation algorithms, *IV-Greedy* and *IV-Community*, to address the influence-maximization problem.

Besides the future work discussed in Chapters 2-4, there are some other exciting extensions in our research plan. First, most existing work on community detection and influence diffusion only focuses on network structure without using node profiles.

This may imply considerable loss of information since the node profile is not only an essential ingredient of a network but also an important complement to network structure. It is desirable to develop algorithms that integrate network structure with node profiles. For community detection, one straightforward solution is to define a unified vertex-pair closeness metric combining SIN similarity and attribute-based similarity with different weights, which will lead to more precise community detection and better interpretation of detected communities. For viral marketing, we can use node profiles to study homophily-driven diffusion and distinguish it from influence-based contagion, which will enable us to develop a deeper understanding of adoption behaviors and arrive at more effective seeding strategies. Second, social networks are dynamic with social interactions changing constantly, and so are the underlying communities. Communities may grow, shrink, merge, and split. It is interesting to develop an adaptive algorithm that tracks community evolution over time and enables us to predict future trends of communities in dynamic networks. A promising direction is to do community detection at different snapshots so as to find community evolutionary patterns, and then use classification techniques to build an effective learning model for community-evolution prediction. Third, there are many signed networks with both positive (friend) and negative (foe) relationships, and it is common that competing items concurrently diffuse in the same social network. It is interesting to extend the MAT model and influence-maximization algorithms to signed networks and/or in a competitive setting. Finally, many complex systems are modeled by multiplex networks, in which the same set of nodes are connected via different types of links in multiple interacting network layers. For instance, some people who are so inactive in offline social networks, are very active and influential in online social networks. Ignoring such information may yield misleading results. It is desirable to generalize our influence-diffusion models and algorithms for community detection and influence maximization to multiplex networks.

REFERENCES

[1] E. Adar and L. Adamic. Tracking information epidemics in blogspace. In *ACM International Conference on Web Intelligence*, 2005.

[2] R. Albert, H. Jeong, and A. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406(27):378–382, 2000.

[3] L. AlSumaidan and M. Ykhlef. Toward information diffusion model for viral marketing in business. *International Journal of Advanced Computer Science and Applications*, 7(2):637–646, 2016.

[4] S. Aral, L. Muchnik, and A. Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *PNAS*, 106(51):21544–21549, 2009.

[5] A. Arenas, J. Duch, A. Fernandez, and S. Gomez. Size reduction of complex networks preserving modularity. *New J of Physics*, 9, 2007. 176.

[6] A. Arenas, A. Fernandez, and S. Gomez. Analysis of the structure of complex networks at different resolution levels. *New J of Physics*, 10, 2008. 053039.

[7] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[8] S. Bhagat, A. Goyal, and L. Lakshmanan. Maximizing product adoption in social networks. In 5^{th} *ACM International Conference on Web Search and Data Mining*, pages 603–612, 2012.

[9] V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J of Statistical Mechanics: Theory and Experiment*, 10:P10008, 2008.

[10] M. Boguna, R. Pastor-Satorras, A. Diaz-Guilera, and A. Arenas. Models of social networks based on social distance attachment. *Phys. Rev. E*, 70:056122, 2004.

[11] P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *J of Mathematical Sociology*, 2:113–120, 1972.

[12] R. M. Bond, C. J. Fariss, J. J. Jones, A. D. I. Kramer, C. Marlow, J. E. Settle, and J. H. Fowler. A 61-million-person experiment in social influence and political mobilization. *Nature*, 489:295–298, 2012.

[13] A. Borodin, Y. Eilmus, and O. J. Threshold models for competitive influence in social networks. In 6^{nd} *Workshop on Internet and Network Economics*, pages 539–550, 2010.

[14] U. Brandes and D. Fleischer. Centrality measures based on current flow. In 22^{nd} *Annual Conference on Theoretical Aspects of Computer Science*, pages 533–544, 2005.

[15] J. Brown and P. Reinegen. Social ties and word-of-mouth referral behavior. *Journal of Consumer Research*, 14(3):350–362, 1987.

[16] R. S. Burt. Social contagion and innovation: Cohesion versus structural equivalence. *American J. of Sociology*, 92(6):1287–1335, 1987.

[17] D. Centola. The spread of behavior in an online social network experiment. *Science*, 329(5996):1194–1197, 2010.

[18] H. Chen and L. A. Complex network characteristics and invulnerability simulating analysis of supply chain. *J. of Networks*, 7(3):591–597, 2012.

[19] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan. Influence maximization in social networks when negative opinions may emerge and propagate. In *SDM*, 2011.

[20] W. Chen, L. Lakshmanan, and C. Castillo. *Information and Influence Propagation in Social Networks.* Morgan & Claypool Publishers, 2013.

[21] W. Chen, W. Lu, and N. Zhang. Time-critical influence maximization in social networks with time-delayed diffusion process. In *AAAI*, 2012.

[22] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In 10^{th} *International Conference on Data Mining*, 2010.

[23] J. Cheng, L. Adamic, P. Dow, J. Kleinberg, and J. Leskovec. Can cascades be predicted?. In *WWW*, 2014.

[24] N. A. Christakis and J. H. Fowler. The spread of obesity in a large social network over 32 years. *New England J of Medicine*, 357:370–379, 2007.

[25] N. A. Christakis and J. H. Fowler. *Connected: The Surprising Power of Our Social Networks and How They Shape Our Lives - How Your Friends' Friends' Friends Affect Everything You Feel, Think, and Do.* Little, Brown & Company, 2011.

[26] N. A. Christakis and J. H. Fowler. Social contagion theory: Examining dynamic social networks and human behavior. *Statistics in Medicine*, 32:556–577, 2013.

[27] A. Clauset, M. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, 2004.

[28] P. Clifford and A. Sudbury. A model for spatial conflict. *Biometrika*, 60(3), 1973.

[29] R. Criado, J. Flores, B. Hernandez-Bermejo, J. Pello, and M. Romance. Vulnerability of complex networks under random and intentional attacks. In *CMMSE*, pages 1–8, 2004.

[30] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *J of Stat. Mech.*, page P09008, 2005.

[31] I. Dhillon and D. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[32] P. Domingos and M. Richardson. Mining the network value of customers. In 7^{th} *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2001.

[33] L. Donetti and M. Muñoz. Detecting network communities: A new systematic and efficient algorithm. *J of Stat. Mech.*, page P10012, 2004.

[34] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World.* Cambride University Press, 2010.

[35] E. Estrada and N. Hatano. Communicability in complex networks. *Phys. Rev. E*, 77:036111, 2008.

[36] E. Estrada and N. Hatano. Communicability graph and community structures in complex networks. *J of Applied Mathematics and Computation*, 214:500–511, 2009.

[37] X. Fang, P. J. Hu, L. Li, and W. Tsai. Predicting adoption probabilities in social networks. *Information Systems Research*, 24(1):128–145, 2013.

[38] G. Flake, S. Lawrence, and C. Giles. Efficient identification of web communities. In 6^{th} *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 150–160, 2000.

[39] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3–5):75–174, 2010.

[40] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences of USA*, 104(1):36–41, 2007.

[41] F. Fouss, P. A., J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[42] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[43] 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.

[44] N. Gayraud, E. Pitoura, and P. Tsaparas. Diffusion maximization in evolving social networks. In *ACM Conference on Online Social Networks*, 2015.

[45] J. Gehrke, P. Ginsparg, and J. M. Kleinberg. Overview of the 2003 KDD cup. *SIGKDD Explorations*, 5:149–151, 2003.

[46] J. Gil-Mendieta and S. Schmidt. The political network in Mexico. *Social Networks*, 18(4):355–381, 1996.

[47] M. Girvan and M. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of USA*, 99(12):7821–7826, 2002.

[48] B. H. Good, Y. de Montjoye, and A. Clauset. The performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81:046106, 2010.

[49] A. Goyal, F. Bonchi, and L. Lakshmanan. Learning influence probabilities in social networks. In 3^{rd} *ACM International Conference on Web Search and Data Mining*, 2010.

[50] A. Goyal, W. Lu, and L. Lakshmanan. CELF++: Optimizing the greedy algorithm for influence maximization in social networks. In *WWW*, 2009.

[51] A. Goyal, W. Lu, and L. Lakshmanan. Simpath: An efficient algorithm for influence maximization under linear threshold model. In *IEEE International Conference on Data Mining*, 2011.

[52] R. Guimera and L. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.

[53] R. Guimera, L. Danon, D.-G. A., F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68:065103, 2003.

[54] R. Guimera, M. Sales-Pardo, and L. Amaral. Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E*, 70:025101, 2004.

[55] X. He, G. Song, W. Chen, and Q. Jiang. Influence blocking maximization in social networks under the competitive linear threshold model. In *SIAM Conference on Data Mining*, pages 463–474, 2012.

[56] S. Hill, F. Provost, and C. Volinsky. Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, 21(2):256–276, 2006.

[57] O. Hinz, B. Skiera, C. Barrot, and J. Becker. Seeding strategies for viral marketing: An empirical comparison. *Journal of Marketing*, 75(6):55–71, 2011.

[58] P. N. Howard and A. Duffy. Opening closed regimes, what was the role of social media during the arab spring?. *Project on Information Technology and Political Islam*, pages 1–30, 2011.

[59] J. Huang, H. Sun, J. Han, H. Deng, Y. Sun, and Y.Liu. Shrink: A structural clustering algorithm for detecting hierarchical communities in networks. In 19^{th} *ACM International Conference on Information and Knowledge Management*, pages 219–228, 2010.

[60] A. Hughes and P. L. Twitter adoption and use in mass convergence and emergency events. *International Journal of Emergency Management*, 6(3):248–260, 2009.

[61] J. Iribarren and E. Moro. Impact of human activity patterns on the dynamics of information diffusion. *Phys. Rev. Letters*, page 103:038702, 2009.

[62] R. Iyengar, C. Van den Bulte, and T. Valente. Opinion leadership and social contagion in new product diffusion. *Marketing Science*, 30(2):195–212, 2011.

[63] R. A. Jarvis and E. A. Patrick. Clustering using a similarity measure based on shared nearest neighbors. *IEEE Transactions on Computers*, C-22(11):1025–1034, 1973.

[64] Q. Jiang, G. Song, G. Cong, Y. Wang, W. Si, and K. Xie. Simulated annealing based influence maximization in social networks. In 25^{th} *AAAI Conference on Artificial Intelligence*, 2011.

[65] S. Jurvetson. What exactly is viral marketing?. *Red Herring*, pages 110–111, 2000.

[66] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

[67] E. Katz and L. P.F. Personal Influence. *Glencoe, IL: Free Press*, 1955.

[68] L. Katz. A new status index derived from sociometric index. *Psychometrika*, pages 39–43, 1953.

[69] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In 9^{th} *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[70] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147, 2015.

[71] Y. Kim, Y. Chen, and K. Linderman. Supply network disruption and resilience: A network structural perspective. *J. of Oper. Manag.*, 33(34):43–59, 2015.

[72] R. Kindermann and J. L. Snell. Markov random fields and their applications. *American Mathematical Society*, 1980.

[73] P. R. Kleindorfer and G. H. Saad. Managing disruption risks in supply chains. *Production and Operations Management*, 14(1):53–68, 2005.

[74] R. Kozinets, K. de Valck, A. Wojnicki, and S. Wilner. Networked narratives: Understanding word-of-mouth in online communities. *Journal of Marketing*, 74(2):71–89, 2010.

[75] A. D. I. Kramer, J. E. Guillory, and J. T. Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111:8788–8790, 2014.

[76] T. La Fond and N. J. Randomization tests for distinguishing social influence and homophily effects. In *WWW*, 2010.

[77] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E*, 80:016118, 2009.

[78] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80:056117(1–11), 2009.

[79] A. Lancichinetti, S. Fortunato, and J. Kertesz. Detecting the overlapping and hierarchical community structure in complex networks. *New J of Physics*, 11, 2009. 033015.

[80] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithm. *Phys. Rev. E*, 78:046110, 2008.

[81] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In 13^{th} *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2007.

[82] J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In *WWW*, 2010.

[83] J. Leskovec, M. McGlohon, Faloutsos, G. N., and H. M. Cascading behavior in large blog graphs. In *SIAM International Conference on Data Mining*, 2007.

[84] I. Leung, P. Hui, P. Liò, and J. Crowcroft. Towards real-time community detection in large networks. *Phys. Rev. E*, 79:066107, 2009.

[85] Y. Li, W. Chen, Y. Wang, and Z. Zhang. Influence diffusion dynamics and influence maximization in social networks with friends and foe relationships. In 6^{th} *ACM International Conference on Web Search and Data Mining*, pages 657–666, 2013.

[86] B. Libai, E. Muller, and R. Peres. Decomposing the value of word-of-mouth seeding programs: Acceleration vs. expansion. *Journal of Marketing Research*, 50:161–176, 2013.

[87] L. Liu, J. Tang, J. Han, and S. Yang. Learning influence from heterogeneous social networks. *Data Mining and Knowledge Discovery.*, 25:511–544, 2012.

[88] W. Liu, M. Pellegrini, and W. X. Detecting communities based on network topology. *Scientific Report*, 4:5739, 2014.

[89] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54:396–405, 2003.

[90] H. Ma, H. Yang, M. Lyu, and I. King. Mining social networks using heat diffusion processes for marketing candidates selection. In 17^{th} *ACM International Conference on Information and Knowledge Management*, 2008.

[91] F. Malliaros and M. Vazirgiannis. Clustering and community detection in directed networks: A survey. *Phys. Reports*, 533:95–142, 2013.

[92] J. Michael and J. Massey. Modeling the communication network in sawmill. *Forest Products Journal*, 47:25–30, 1997.

[93] E. Mossel and S. Roch. Submodularity of influence in social networks: From local to global. *SIAM Journal on Computing*, 39(6):2176–2188, 2010.

[94] M. Moussaid, H. Brighton, and W. Gaissmaier. The amplification of risk in experimental diffusion chains. *Proceedings of the National Academy of Sciences*, 112:5631–5636, 2015.

[95] A. Nair and J. M. Vidal. Supply network topolgy and robustness against disruptions - An investigation using multiagent model. *Int. J. Production Research*, 49(5):1391–1404, 2011.

[96] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[97] M. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, 2004.

[98] M. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, 2004.

[99] M. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27:39–54, 2005.

[100] M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[101] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

[102] J. D. Noh and H. Rieger. Random walks on complex networks. *Phys. Rev. Letters*, page 92:11870, 2004.

[103] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, Stanford University, 1999.

[104] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

[105] R. Peres. The impact of network characteristics on the diffusion of innovations. *Physica A*, 402:330–343, 2014.

[106] P. Pons and M. Latapy. Computing communities in large networks using random walks. *J of Graph Algorithms Applications*, 10(2):191–218, 2006.

[107] R. Radicchi, C. Castellano, F. Cecconi, and D. Parisi. Defining and identifying communities in networks. *Proceedings of National Academy of Sciences of USA*, 101:2658–2663, 2004.

[108] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76:03106, 2007.

[109] W. Rand and R. Rust. Agent-based modeling in marketing: Guidelines for rigor. *International Journal of Research in Marketing*, 28(3):181–193, 2011.

[110] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In 8^{th} *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2002.

[111] D. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on twitter. In 20^{th} *International Conference on World Wide Web*, pages 695–704, 2011.

[112] M. Rosvall and C. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of National Academy of Sciences*, 104:7327–7331, 2007.

[113] M. Rosvall and C. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of USA*, 105:1118–1123, 2008.

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

[115] P. Schmitt, B. Skiera, and C. Van den Bulte. Referral programs and customer value. *Journal of Marketing*, 75(1):46–59, 2011.

[116] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[117] K. A. Stephenson and M. Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11:1–37, 1989.

[118] H. Sun, J. Huang, J. Han, H. Deng, P. Zhao, and B. Feng. gSkeletonClu: Density-based network clustering via structure-connected tree division or agglomeration. In 10^{th} *International Conference on Data Mining*, pages 481–490, 2010.

[119] H. Sun and J. Wu. Scale-free characteristics of supply chain distribution networks. *Modern Physics Letter B*, 19(17):841–848, 2005.

[120] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. RankClus: Integrating clustering with ranking for heterogeneous information network analysis. In 12^{th} *International Conference on Extending Database Technology*, pages 565–576, Saint Petersburg, Russia, 2009.

[121] H. P. Thadakamalla, U. N. Raghavan, S. Kumara, and R. Albert. Survivability of multiagent-based supply networks: A topological perspective. *IEEE Intelligent Systems*, 19(5):24–31, 2004.

[122] G. Thomas. Building the buzz in the hive mind. *Journal of Consumer Behavior*, 4(1):64–72, 2006.

[123] B. Tomlin. On the value of mitigation and contingency strategies for managing supply chain disruption risks. *Management Science*, 52:639–657, 2006.

[124] S. van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, 2000.

[125] C. Wang, W. Chen, and Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3):545–576, 2012.

[126] W. Wang, R. E. deMatta, and W. N. Street. Topological resilience analysis of supply networks under random disruptions and targeted attacks. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 250–257, 2015.

[127] W. Wang and W. N. Street. A novel algorithm for community detection and influence ranking in social networks. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 555–560, 2014.

[128] W. Wang and W. N. Street. Finding hierarchical communities in complex networks using influence-guided label propagation. In *15th International Conference on Data Mining Workshops*, pages 547–556, 2015.

[129] W. Wang and W. N. Street. Modeling influence diffusion to uncover influence centrality and community structure in social networks. *Social Network Analysis and Mining*, 5(1), 2015.

[130] Y. Wang, G. Cong, G. Song, and K. Xie. Community-based greedy algorithm for mining top-k influential modes in mobile social networks. In 16^{th} *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2010.

[131] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications.* Cambridge University Press, 18 edition, 1994.

[132] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[133] Y. Wei and C. Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *IEEE International Conference on CAD*, pages 298–301, 1989.

[134] J. White, E. Southgate, J. Thomson, and S. Brenner. The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London Series B*, 314(1165):1–340, 1986.

[135] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4):1–35, 2013.

[136] J. Xie and B. K. Szymanski. LabelRank: A stablized label propagation algorithm for community detection in networks. In *IEEE Network Science Workshop*, pages 138–143, 2013.

[137] Y. Xing, F. Meng, Y. Zhou, M. Zhu, M. Shi, and G. Sun. A node influence based label propagation algorithm for community detection in networks. *The Scientific World Journal*, 2014. 627581.

[138] Q. Xuan, F. Du, Y. Li, and T. Wu. A framework to model the topological structure of supply networks. *IEEE Transactions on Automation Science and Engineering*, 8(2):442–446, 2011.

[139] Y. Yang, Y. Sun, S. Pandit, N. Chawla, and J. Han. Is objective function the silver bullet? A case study of community detection algorithms on social networks. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 394–397, 2011.

[140] L. Yen, F. Fouss, C. Decaestecker, P. Francq, and M. Saerens. Graph nodes clustering with the sigmoid commute-time kernel: A comparative study. *J of Data and Knowledge Engineering*, 68:338–361, 2009.

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

[142] J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li. Social influence locality for modeling retweeting behaviors. In 23^{rd} *International Joint Conference on Artificial Intelligence*, pages 2761–2767, 2013.

[143] K. Zhao, A. Kumar, T. P. Harrison, and J. Yen. Analyzing the resilience of complex supply network topologies against random and targeted disruptions. *IEEE Systems Journal*, 5(1):28–39, 2011.

[144] K. Zhao, A. Kumar, and J. Yen. Achieving high robustness in supply distribution networks by rewiring. *IEEE Transactions on Engineering Management*, 58:347–362, 2011.

[145] H. Zhou and R. Lipowsky. Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. In *International Conference on Computational Science*, pages 1062–1069, 2004.