Exploiting Graph Properties for Decentralized Reputation Systems

Dimitra Gkorou

Exploiting Graph Properties for Decentralized Reputation Systems

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 19 november 2014 om 12:30 uur

door **DIMITRA GKOROU**

Diploma on Computer Engineering and Informatics,
University of Patras, Patras, Greece

geboren te Athena, Greece

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. D.H.J. Epema

Copromotor:
dr.ir. J.A. Pouwelse

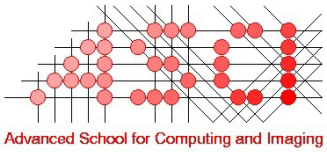
Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. D.H.J. Epema	Technische Universiteit Delft Technische Universiteit Eindhoven, promoter
Dr.ir. J.A. Pouwelse	Technische Universiteit Delft, copromotor
Prof.dr. R. W. van der Hofstad	Technische Universiteit Eindhoven
Dr. A. Iamnitchi	University of South Florida
Prof.dr.ir. R.E. Kooij	Technische Universiteit Delft
Prof.dr. R. Wattenhofer	Eidgenössische Technische Hochschule Zürich (ETHz)
Prof.dr. C. Witteveen	Technische Universiteit Delft
Prof.dr.ir. G.J. Houben	Technische Universiteit Delft, reservelid

Published and distributed by: Dimitra Gkorou
E-mail: dgkorou@gmail.com
ISBN: 978-94-6295-016-0
Keywords: Trust, decentralized reputation systems, social networks, network analysis, graph mining, network evolution, sybil attacks, random walks.

The cover is designed by Nastazia Tsoupa in collaboration with Dimitra Gkorou, 2014, and it is inspired by Marcel's Duchamp Network of Stoppages, 1914.

Printed in The Netherlands by: Uitgeverij BOXPress.



The work described in this thesis has been carried out in the ASCI graduate school. ASCI dissertation series number 320.

Acknowledgements

The accomplishment of my thesis reflects a precious journey resulting in a significant scientific as well as personal growth. Here, I take the opportunity to thank the people who have contributed to it along the way.

Dick, thank you for your nurturing guidance and your disciplined critical thinking, which have been key motivations for me. I have been fortunate to have you as my advisor, as you gave me the freedom to explore new ideas as well as the appropriate guidance and support when my steps stumbled. Your insightful comments on my research and your scrutiny of my technical writing have been invaluable on my attempts towards formulating questions and expressing my ideas. I also appreciate your ability of making interesting any casual conversation, even if it is about the most trivial topics.

Johan, thank you for your unceasing enthusiasm, your pursue towards research with practical implications, and your many interesting ideas. Your support on implementing my ideas to Tribler and your advices on making my technical writing more interesting have significantly influenced this thesis. I would also like to thank you for assembling the Tribler group with so many talented people.

Henk, thank you for our pleasant conversations and running up the incredible PDS group. I am also thankful to my committee members for their constructive feedback on my thesis.

I would like to to express my gratitude to my officemates for providing me with a pleasant and encouraging office atmosphere during all these years. The combination of sharing scientific ideas as well as having fun was inspiring. Nitin, thanks for your valuable input to all my ideas and Boxun, thanks for your advices on refining my plots. Dear friends, your appreciation of contemporary art, combined with Rahim's, offered us unforgettable moments such as #SpanishFresco, a scientific "graffiti" on a white board, and an Andy Warhol-inspired tower made of cans. Mihai, thanks for your encouragement on my attempts to implement my ideas, printing for me the appropriate cheat sheets, and the fancy marshmallow shots, together with Bodgan, in Amsterdam. Niels, thanks for your insights and explanations on Bartercast code and many other technical issues. Dear paranymphs, we had many stimulating and funny conversations, particularly after Mihai's suggestions of movies for the movie nights of our group.

Special thanks go to Tamás and Boudewijn for their fruitful collaboration in different phases of this thesis. Tamás, thank you for your positive attitude, your insightful comments, your invaluable contribution to the supervised random walks, and all our engaging talks. Boudewijn, I am thankful to you for all your support with the Bartercast code, your patience on answering even my most naive questions on it, and your contribution to the DAS experiments.

I would like to thank all the wonderful and talented people, with whom I had the opportunity to interact, in the PDS group: Riccardo, Rahim, Bogdan, David, Lucia, Adele, Arno, Rameez, Jie, Lipu, Siqi, Jiabin, Yong, Alexey, Naza, Victor, Ana, Alex, Flutra, Otto, Kefeng, Paulo. Many thanks to Cor-Paul for translating the summary of my thesis in Dutch and David for keeping sending me interesting emails. Paulo, Munire, and Stephen, thank you for the excellent technical support. Rina, Ilse, Esther, and Monique, thank you for your support on administrative issues. It has been a great pleasure meeting all of you.

I am grateful to all the people in my alma matter, the University of Patras, who have encouraged me to pursue a PhD, and in particular, my diploma thesis advisor Efstratios. Many thanks go to my friends in Patras, Athens, and London for proving that friendship eliminates geographical distance, obviously, when provided with the appropriate technical support. A special thanks goes to Nastazia for her contribution to the cover of this thesis and Maria for our endless conversations. I am also grateful to my friends in Utrecht, Amsterdam, and Eindhoven, for our stimulating discussions, our pleasant trips, and generally, our great days and nights.

I would like to express my gratitude to my family, my sister Angeliki, my brother Christos, and my parents Christina and Dimitrios. You are always there for me, full of unconditional love and support, encouragement to pursue my interests, and many hugs. Yiorgo, special thanks to you, for sincerely sharing my joys and difficulties, encouraging me, and keeping your sense of humour when I loose mine.

Contents

1 Introduction **1**
1.1 Enabling Trust through Reputation Systems 2
1.2 Online Reputation Systems 3
1.2.1 Decentralization 4
1.2.2 Problems due to Decentralization 5
1.3 Reputation Systems as Interaction Graphs 7
1.3.1 Studying and Building the Interaction Graphs 8
1.3.2 Online Communities under Study 9
1.4 Research Context: Tribler 12
1.5 Problem Statement 13
1.5.1 Computation of Reputation 13
1.5.2 Storage of Information 14
1.5.3 Collection of Information 14
1.6 Contributions and Thesis Outline 15

2 Betweenness Centrality Approximations **17**
2.1 Background 18
2.1.1 The Bartercast Reputation Mechanism 19
2.1.2 Betweenness Centrality 20
2.2 Dataset and Evaluation Metrics 20
2.2.1 Random Graphs 20
2.2.2 Scale-free Graphs 21
2.2.3 Bartercast Graphs 21
2.2.4 Evaluation Metrics 22
2.3 Evolution of Betweenness Centrality in Growing Networks 23
2.4 Approximations of Betweenness Centrality 27
2.4.1 Definition of BC Approximations 27
2.4.2 Experimental Results 29
2.5 Increasing Efficiently the Accuracy of the Computation of Reputation in Bartercast 30

2.6 Conclusion 32

3 Leveraging Node Properties in Random Walks for Robust Reputations 33
3.1 Problem Statement 34
3.1.1 Motivation 35
3.1.2 Definitions and Network Model 35
3.2 Datasets 37
3.3 Choosing the Time Window 39
3.4 Identifying Properties of Nodes Indicative of their Behavior 41
3.4.1 Introducing the Properties of Nodes 41
3.4.2 Evaluation 43
3.5 Biasing Random Walks in the Face of Uncooperative Nodes 45
3.5.1 Naive Random Walks 46
3.5.2 Supervised Random Walks 48
3.6 Biasing Random Walks in the Face of Sybil Attacks 51
3.7 Conclusion 56

4 Forgetting the Least Important parts of the History of Interactions 57
4.1 Problem Statement 58
4.2 Creating the Reduced History 59
4.2.1 The Parameters for Node Removal 59
4.2.2 The Parameters for Edge Removal 61
4.3 Datasets 62
4.4 Computation of Reputations and Evaluation Metrics 65
4.5 Evaluation 66
4.5.1 Experiments and Results 67
4.5.2 Discussion 71
4.6 Related work 72
4.7 Conclusion 73

5 Trust-based Collection of Information 75
5.1 Problem Statement 76
5.1.1 Resilience to Attacks 77
5.1.2 Scalability 77
5.1.3 Relevance of Information 78
5.1.4 Trade-off among the Requirements 78
5.2 Design Considerations 78
5.2.1 Collection of Reports 79
5.2.2 Incorporating Trust 80
5.2.3 Direction of Information 81

5.2.4 Type of Information Spread 81
5.3 Collecting Information Using Random Walks 82
5.3.1 Network Model and Definitions 82
5.3.2 Types of Random Walks 83
5.3.3 Implementation of Random Walks 83
5.4 Experiment Methodology 86
5.4.1 Datasets 86
5.4.2 The Restart Probability 88
5.4.3 Experiment Setup 89
5.4.4 Sybil Attack Model 90
5.5 Evaluation 91
5.5.1 Resilience against Sybil Attacks 91
5.5.2 Scalability 92
5.5.3 Relevance of Information 94
5.5.4 Discussion 96
5.6 Conclusion 97

6 Conclusion **99**
6.1 Conclusions 99
6.2 Suggestions for Future Work 101

Bibliography **103**

Summary **113**

Samenvatting **115**

Curriculum vitae **119**

Chapter 1

Introduction

Computers and connectivity pervade our lives providing us with many opportunities to interact with strangers. Nowadays, we participate in multiple online communities for buying and selling services and objects, educating ourselves, watching videos, keeping up with friends, and playing games. As members of those online communities, we are able to easily create, publish, and consume content on the Internet reaching audiences of unprecedented scale. Such rich activity on the Internet increases uncertainty and brings new forms of fraud, which are easier and more profitable than ever before. Establishing trust among strangers is essential for the prosperity of offline and online communities. While many traditional trust mechanisms such as contractual guarantees and repeated interactions are very costly or even infeasible in large-scale environments, online reputation systems are an effective way to protect users from fraud and provide incentives for them to cooperate and contribute to the system. Just like their counterparts in offline communities, they aggregate the history of user interactions in one reputation value per user. In many different contexts, online reputation systems function effectively on a worldwide scale and allow millions of users daily to take decisions about their future interactions with strangers. Examples of popular online reputation systems are those of eBay, Amazon, eLance, oDesk, Google, Booking, Youtube, topCoder, and CouchSurfing. All those online reputation systems rely on a single point of control.

In this thesis, we study reputation systems for decentralized networks such as distributed online social networks, online markets on mobile devices, and P2P networks. Growing privacy concerns in online communities and popularity of applications on mobile devices motivate the use of decentralized reputation systems where each user individually collects the history of user interactions, stores it, and aggregates it to one reputation value per user. Due to the highly dynamic behavior of users and the scarcity of resources, several challenging scalability and security issues arise. First, the increasing number of users in online communities makes it hard for those systems to scale up to a large number of users. Secondly, the large amount of available information challenges the users to iden-

tify relevant information for computing the reputations. Finally, the ease and the low cost of creating accounts in these communities enables malicious users to promote themselves by spreading false information about their interactions.

To face these challenges, we propose algorithms that exploit the graph structure induced by the user interactions in decentralized reputation systems. The socially rich available information of online communities allows us to study the intrinsic patterns of their user behavior and their evolution over time. We observe their static and temporal properties both at the local and the global level. Using the key insights of our analysis we develop scalable and effective algorithms in order to collect, store and aggregate information in reputation systems. The developed algorithms are computationally tractable and resilient to adversarial environments. To identify and maintain only relevant and trustworthy information, we leverage the history of user interactions since frequent and successful interactions indicate trust and similarity between the corresponding users.

1.1 Enabling Trust through Reputation Systems

We define an online community as any group of users interacting online with a loose common goal and direction. The level of *trust* in a community impacts its prosperity by reducing the risk and cost of interactions. Higher levels of trust decrease the interaction costs, thus facilitating the activity of a community [56]. To highlight the importance of trust in the development of a community, Fukuyama states that ”its well-being, as well as its ability to compete, is conditioned by a single pervasive cultural characteristic: the level of trust inherent in the society ” [56]. The sociological literature defines trust across three different levels as either a property of individuals, of social relationships, or of the social system [88]. At the individual level, an individual trusts other individuals to do something based on some knowledge about their disposition, ability, or reputation. At the collective level, an individual trusts others based on its trust in the agency or organization with which the others are affiliated. At the system level, individuals consider the background, culture, and social system of others when trying to determine whether to trust them.

Even though trust is necessary for the prosperity of a community, it is very challenging to enforce it. Communities have a set of interests associated with the goals and directions of the community. Every person in a community has one or more self-interests that may conflict with the group interest. A *societal dilemma* captures the conflict between the self-interest of a community member and the group interest, the choice whether to defect or to cooperate. Many types of societal dilemmas have have been studied in the literature: the collective action problem, the tragedy of the commons, the free-rider problem, the arms race, and the prisoner’s dilemma.

Reputation systems have evolved in societies as an efficient way to enforce trust and cooperation at a satisfying level in societal dilemmas via social pressure [106]. The rep-

utation of an individual is a score estimated based on his past interactions that is being disseminated among the members of a community informing them about his attitude. The expectation that other members will consider each other's history of interactions in future interactions, the "shadow of the future" according to Axelrod [8], constrains their present behavior forcing them to cooperate. To demonstrate the power of reputational pressure, we present how reputations resolve the well known Prisoner's Dilemma [106]. According to the Prisoner's Dilemma, two partners in a crime are interrogated separately by the police due to lack of evidence for their conviction. The criminals could cooperatively deny their crime and not testify against each other, which would result in a minor sentence for both. Fearing of being betrayed by their partner and serving the longest period in jail alone, they are both inclined to testify against each other spending a moderate period in jail. However, their reputation in the underworld depends on them not betraying each other and eventually, it enforces them to cooperate. No doubt, reputational pressure does not eliminate betrayals as depending on the country and the severity of the crime, the police can be very persuasive for the criminals to testify.

Reputation systems ensure that the rate of defectors stays small enough to allow communities to remain cohesive without imposing large cost or reducing the convenience of interactions [103]. In a reputation system, typical users are imperfectly informed about the past interactions of other users and the collection of perfect information is usually too expensive or in several cases impossible to obtain [57]. On the other hand, strict security mechanisms and contractual guarantees would raise prohibitively the cost of user interactions, still without ensuring that under any circumstance every user will cooperate.

1.2 Online Reputation Systems

In online communities, contractual guarantees are infeasible due to the global dispersion and the large number of their users. Repeated interactions between users are also rare due to their large scale, e.g., in eBay 89% of the transactions between a pair of buyer-seller [44] and in P2P systems about 92% of data transfers between a pair of peers [96] are conducted only once. Online reputation systems allow members of a community to take decisions about their online interactions with strangers in many different applications. They enable us to trust unknown products on eBay, unknown professionals on oDesk, unknown friends on Facebook, and unknown couches on Couchsurfing. In Table 1.1, we present the analogies in terminology among different types of applications using online reputation systems. Table 1.1 is a modification of a table presented in [32].

Online reputation systems differ from their offline counterparts in their large number of participants spread around the world, their explicit design, and the variety of defector strategies [44]. Online communities have hundreds of millions of users spread across many countries, e.g., Facebook has over a billion users in more than 70 countries [50]

Table 1.1: Analogies in the terminology of reputation systems

File-sharing	Marketplace	Service-Oriented	Social Media and Networks
download	transaction	subscription	interaction
file	product	service	video/photo/story
uploader	seller	provider	uploader
downloader	buyer	consumer	viewer

and Youtube more than 100 million in 61 countries [124]. Scale is critical and it causes the shift from trust based on personal relationships to impersonal trust. While offline reputation systems could never handle this scale, online reputation systems can efficiently collect, store and aggregate a very large amount of information. They have been explicitly designed to control the participants, the information spread, the computation of reputation, and the storage of information. However, depending on the reputation system, attackers can adopt a great range of strategies: attackers might be *traitors* who were previously cooperative users milking their reputation by exploiting other users, they might *whitewash* their badly reputed identities by leaving the system and reappearing under a new identity, or they might claim dishonest feedback by *collusion* or controlling fake identities, their *sybils*. In this section, we discuss the decentralization of reputation systems and the challenges due to decentralization.

1.2.1 Decentralization

Decentralization of online communities and their reputation systems is driven mainly by the growing concerns of privacy and the popularity of applications on mobile devices. In Facebook, Twitter, and Youtube, hundreds of millions of active users share content on their websites revealing their personal information. Privacy concerns are attracting increasing attention from users. Even though most online communities offer privacy settings, their centralized nature facilitates tracking and censorship. Malicious users can keep track of particular users, classify personalities from their individual usage behavior, track their locations even when offline, and censor their content. To expose the ease of crawling data from online communities, we note that researchers very often crawl online communities to collect datasets for their studies — which they usually anonymize — and several books guide data crawling in social networks for research purposes [105]. An example of the privacy implications is the website pleaserobme.com which by crawling social networks for checks in restaurants, bars, and concerts, and holiday announcements, suggests houses whose dwellers are on leave to potential robbers.

Decentralized social networks have been proposed to protect users' privacy. In decentralized social networks, users have the ownership of their data and perform secure com-

munication with their friends without passing any information through central servers. In Diaspora [2], users setup their own servers to host content, form friendships, and share content with others. Safebook [37] combines a peer-to-peer architecture with trust relationships to preserve privacy. In Peerson [22] users keep control of their data through encryption, key management and access control in a decentralized way. In P2P filesharing networks, decentralized reputation mechanisms have been proposed against malicious users and content such as Bartercast [43], EigenTrust [78], Havelaar [67] and PET [32].

Concurrently with the concerns of privacy in online communities, an increasing number of users are engaged in applications on mobile devices by creating and sharing content at low cost. Half of Facebook users access it through their mobile devices [50]. Youtube users watch more than 6 billion hours of video each month with 40% of this watch time spent on mobile devices [124]. To identify relevant and quality content, researchers have proposed several reputation systems to rate and locate content producers in applications on mobile devices. Using centralized reputation systems, mobile devices have to go through central servers. For scalability and mobility, researchers have concluded that reputation systems on mobile devices have to be decentralized [86], [101]. Examples of such decentralized reputation systems include Mobile Bazaar [27] proposed for markets on mobile devices, MobID against malicious users on social applications [102], and the reputation system proposed by McNamara et al. [86] for content rating on media applications.

Allowing each user to collect, store and aggregate information about other users on his own device is an efficient way of decentralizing reputations systems as indicated by many researchers [86], [101], [43], [27]. Consequently, a decentralized reputation system is naturally divided into three components: the collection of the history of user interactions, its storage, and the computation of reputations [109]. During the *collection* of the information, users have to identify and collect information about the history of interactions of other users in order to compute their reputations. The *storage* of information deals with the type of stored data, the storage methodology, and the choice of the stored information. In the *computation* of reputations, each user aggregates the history of user interactions in one reputation value per user. Decentralization aggravates the fundamental problem of reputation systems which consists in collecting, storing, and aggregating *relevant* and *trustworthy* information. Below, we describe this problem.

1.2.2 Problems due to Decentralization

The burst of user interactions in online communities and the low cost of creating accounts pose the challenge of collecting, storing, and aggregating *relevant* and *trustworthy* information. The ease of account creation and the freedom to share content has attracted millions of users to contribute in online communities. For instance, Facebook has over

a billion of users uploading over 250 million photos everyday [50]. Twitter has over 500 millions users sending over 58 million tweets and issue over 2.1 billion queries each day [113]. In Youtube, over 100 million users upload every minute more than 100 hours of videos [124]. With this information overload, collecting and storing *relevant* information about past interactions for the computation of reputations is challenging. This problem is also known in the literature as the Babel objection [16]. Classical data mining techniques and recommendation algorithms, such as k-Nearest Neighbor [75], are not able to deal with the scale of the number of users in online communities and the sparsity of their interactions. For example, in Facebook a typical user is connected to about 100 other users out of one billion users in the system. Thus, a good but useless recommendation to a new user is to recommend no new friends. Due to the large number of users and the sparsity of their interactions, this useless recommendation achieves near perfect predictive accuracy, namely only 100 mistakes out of one billion possible predictions [9]. The effect of information overload is even more intense in decentralized reputation systems, as in these systems each user has very limited resources available for the collection, the storage and the processing of information.

Not only detecting relevant information is challenging in online communities, detecting trustworthy information is challenging as well. In most of these communities, anyone is able to create an account easily which enables malicious nodes to perform *sybil attacks* by creating fake accounts under their control to spam, collude or perform link farming. The attacker manages to gain a disproportionally large influence on the network in order to determine the result of a voting process, to spam other users, to monopolize system resources, or even to sell its sybils to other attackers as has occurred in Facebook [25]. Facebook estimated that about 5% to 11% of its accounts are illegitimate [49] and Twitter about 5% [112]. Renren and Tuenti also have such fake accounts [123], [25]. Classical machine learning techniques, such as Support Vector Machine [75] and logistic regression [75], have been used to classify malicious accounts based on their properties such as activity level and frequency of interactions [123]. However, these techniques are effective only when the characteristics of sybil accounts deviate clearly from normal users. Typically, malicious users tend to mimic normal users, making those techniques inefficient. Most of the proposed solutions against sybil attacks in the literature [117] are based on the observation that honest users form an area in the network well-separated from sybil identities, as illustrated in Figure 1.1. As interacting with honest nodes requires high engineering cost, sybil nodes are able to create only a limited number of connections with honest users, the *attack edges*. As a result, the sybil nodes constitute the *sybil area* which is well separated from the *honest area*. Due to this characteristic of sybil nodes, community detection algorithms such as the Louvain method [17], tend to be effective in detecting the sybil area. However, when the honest are consists of multiple communities, the identification of sybil area is challenging.

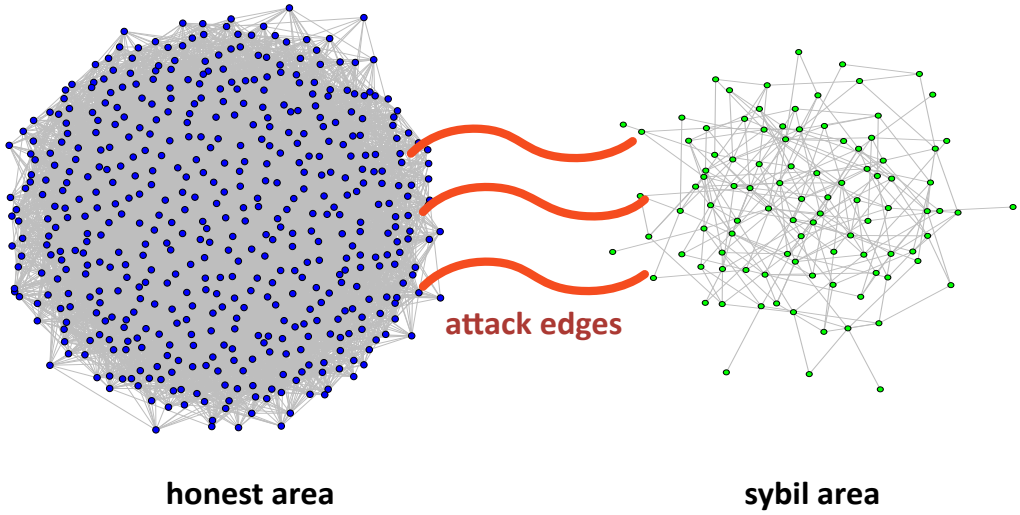


Figure 1.1: Sybil attacks in a reputation system. The sybil nodes connect to the honest nodes via the attack edges.

To identify relevant and trustworthy information, many proposed schemes leverage social networks by interpreting social links among users as similarity and trust relationships. However, in many systems such as P2P networks no social network is available. Furthermore, in social networks many social connections between nodes are superficial or of very low strength and thus, they do not indicate trust. For instance, many users in Facebook have many more friendship connections than 150, which is roughly the number of people they can regularly interact with [46]. As a result, these superficial connections among users correspond to ”Familiar Strangers” [94] rather than real friendships. In this thesis, instead of using social connections among users as trust indicators, we propose the usage of their interactions. Regular and successful interactions between nodes are strong indicators of trust and similarity among users [65], [89]. As a result, our approach is not only useful for systems without any social network available, but it is more resilient against sybil attacks, as well. Below, we describe the usage of interactions in online reputation systems and its benefits.

1.3 Reputation Systems as Interaction Graphs

Representing an online community as a graph, whose vertices are the users and edges are relationships between the users, is a powerful model to observe and analyze their complexity at the micro-level, meso-level and macro-level. In other words, we are able to

study the interconnections of individual nodes in networks, their relative positions in the graph, as well as the structure of the graph itself. Furthermore, the availability of socially rich information of millions of users in online communities enables us to ask questions that were impossible to answer before. This allows computer science to reach towards other sciences such as social sciences, economics, and physics of complex systems [33].

1.3.1 Studying and Building the Interaction Graphs

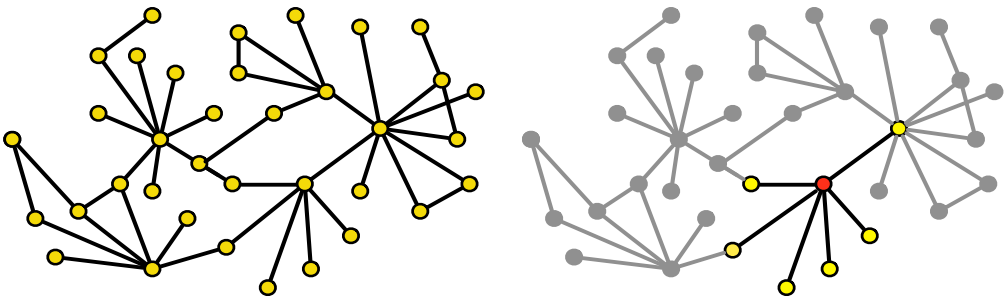
In reputation systems, the subject of interest is user interactions. Thus, we study a reputation system through the corresponding *interaction graph* whose edges represent the user interactions. This graph represents the collective user behavior in an online community, and it is directed indicating the direction of interactions, and weighted indicating the strength of interactions. We observe the interaction graph over time.

At the micro-level, we study the local properties of nodes over time, such as their connectivity, their activity, and their interactions with their immediate neighbors. The frequency and the strength of past interactions among users reveal their similarity and trust relationships as frequent interactions between two users imply strong common interests and trust, respectively.

Meso-level graph properties indicate a user's relative position in a given graph, e.g., its centrality and the transitivity of trust. Users of high centrality participate in more interactions and a major information flow passes through them. Their interactions are very important as they keep the graph connected. These users can be exploited to increase the accuracy of the computation of reputations but they risk being overloaded.

At the macro-level, we study the global properties of the interaction graph, e.g., its clustering in communities, its diameter, and its density. By studying the properties of a graph, we are able to infer characteristics of the underlying system. Furthermore, we are able to explore the influence of graph topology on the performance of a proposed algorithm for collecting, storing, or aggregating the information in reputation systems. We observe that graph properties such as clustering in communities, diameter, and the average path length between any pair of nodes influence a lot the performance of such algorithms.

These three levels of properties are closely related. When implementing an algorithm, we determine a user's interactions with its neighbours and so, we act at the micro-level. However, the system is affected at the macro-level, namely the collective user behavior is also affected. On the other hand, the macro-level properties of an already existing community affect the performance of an algorithm applied to it. For example, it was the power-law structure of WWW that made Google's Pagerank successful while in a random network Pagerank does not perform that well [59]. Pagerank is a ranking algorithm identifying the highest-ranked nodes in a graph using random walks. In a power-law



(a) The complete interaction graph

(b) The interaction subgraph of the red node; the part of the interaction graph drawn with grey indicates the unknown area from the perspective of this node

Figure 1.2: The complete interaction graph and the subgraph of a node (black edges)

graph, there are a few highly ranked nodes deviating clearly from the other nodes, and those highly ranked nodes have many walks leading to them. Consequently, it is easy to identify them.

In a decentralized reputation system, each node cannot have perfect information and thus, it keeps its own subgraph of the interaction graph, its *interaction subgraph*. In Figure 1.2, we illustrate an interaction graph and a corresponding interaction subgraph. A node aggregates its subgraph in one reputation score per node. Initially, a node only knows its own *direct interactions* in its interaction subgraph and so, it is able to compute only the reputations of nodes with which it has previously interacted. Nodes need to expand their interaction subgraphs in order to compute the reputations of many other nodes in the network. For this purpose, they should periodically contact each other acquiring information about the interaction subgraphs of other nodes. Ideally, after a node has contacted several other nodes, its interaction subgraph converges to the complete interaction graph.

1.3.2 Online Communities under Study

In this thesis, we propose algorithms for collecting, storing, and aggregating trustworthy and relevant information in decentralized reputation systems. We evaluate the proposed algorithms on three online communities with different graph properties: the Bartercast reputation system, the author-to-author Citation network, and Facebook in New Orleans. Bartercast is a decentralized reputation system with high population turnover and very few users staying long in and contributing to the system. On the other hand, the Citation network of authors forms a tightly connected community. Finally, Facebook is a social network of tightly connected users with bursty interaction patterns among them. In this section, we present the basic characteristics of the datasets derived from these three online

communities. In each chapter of this thesis, we use different subsets of those datasets with different sizes and time spans depending on the corresponding evaluation. Each of the corresponding subsets is described before the evaluation in each chapter. All the networks are growing and have timestamps associated with their interactions. Furthermore, for comparison reasons, we use synthetic random and power law graphs whose characteristics we describe in each chapter separately.

Bartercast

Bartercast [43] is the reputation mechanism of the BitTorrent-based client Tribler [99]. Bartercast is the motivation of the research presented in this thesis, as Tribler has been developed by the Parallel and Distributed Systems group of Delft University of Technology. Reputations in Tribler are used as an incentive mechanism for users to contribute to the system, as free riding in BitTorrent is easy to perform [82]. Bartercast was initially proposed by Meulpolder et al. [87] and was further improved in terms of its accuracy, security, and scalability by Delaviz et al. [43]. In this thesis we analyze and leverage the graph properties of the network induced by user interactions. We apply our algorithms to multiple online communities gaining key insights about the similarities and the differences of Tribler with other popular online communities. Using relevant and trustworthy information, our algorithms improve the collection, the storage, and the aggregation of the history of user interactions in Bartercast in terms of computational complexity, scalability, and tolerance against attacks.

In Bartercast, when a peer exchanges content with another peer, they both store a *Bartercast record* with the amount of data transferred and the identity of the corresponding peer. Each peer is associated with a unique identifier (UID). Regularly, peers contact other peers to exchange Bartercast records using a gossip-like protocol. From the records it receives, every peer dynamically creates a weighted, directed interaction subgraph, the nodes of which represent the peers about whose activity it has heard through Bartercast records, and the weights of the edges represent the total amount of data transferred between two nodes in the corresponding directions.

In Bartercast, the reputation of a node from the perspective of another node is computed as the difference between the flows passing between those nodes using max-flow. Particularly, each peer i computes the reputation of another peer j by applying the Ford-Fulkerson algorithm [35] to its interaction subgraph to find the maximum flows f_{ij} from itself to j and f_{ji} in the reverse direction, as in Figure 1.3. To limit the computational cost, only paths of length at most two hops from node i are considered. Within two hops, very few nodes are connected on average in Bartercast due to the sparsity of user interactions, resulting in limited accuracy and coverage. The accuracy of the two hop limitation has been studied in [43]. We elaborate on the computation of reputations in Bartercast in

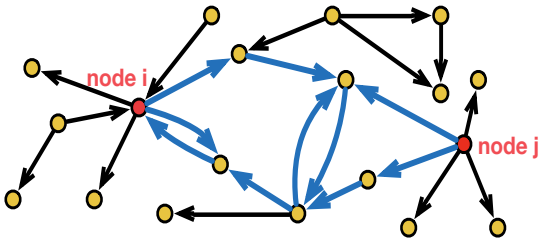


Figure 1.3: The max-flow computation between nodes i and j indicated by the blue arrows

Chapter 2.

The Bartercast reputation system is based on the transitivity of trust, unlike other approaches such as Havelaar [67]. Assuming that honest nodes tend to behave honestly over time, trust transitivity is powerful against Sybil attacks and collusions but vulnerable against traitors. To increase the effectiveness of Bartercast against traitors, the reputation values are computed according to specifically defined time-windows. Thus, peers cannot to accumulate very high reputation values and later milk their reputations exploiting the system, as presented in Chapters 3 and 4.

In order to obtain Bartercast records, Tribler was crawled from September 1, 2010 to January 31, 2011, collecting information from 29,716 nodes [43] . In our analysis, we will assume *full-gossip* in which peers forward the records they receive to all other peers, and so all peers eventually receive all the records in the system. The corresponding interaction graph, the *Bartercast graph*, is derived from all interactions between the peers in Tribler.

The Citation Network

The author-to-author Citation graph is derived from the citation network of the papers published in Physical Review E, which is a journal operated by the American Physical Society (APS). The vertices of the citation graph represent the authors of papers and the edges represent the citation relationship between two authors (or coauthors). The direction of an edge indicates the direction of the citation and its weight indicates the number of citations from one author to another. The Citation graph is one of the most widely studied graphs in network analysis [85] and its authors are tightly connected.

The dataset we analyse covers the time span from 1993 to 2011 and it became available to us upon request to APS. The authors in the dataset are not associated with unique identifiers. As a result, it is not trivial to distinguish two different authors publishing under the same names, or an author publishing with slightly different first name. We mapped a UID to a every combination of an author's first letter of first name and his surname, as it is quite uncommon for two authors to have published under the same name in the same field and so, the structure of the citation graph is not affected [85].

Facebook

The Facebook graph used in this thesis derives from the Facebook network in New Orleans. We use the dataset presented in [115] containing information about the interactions of 60,290 users, as indicated by their wall posts, from September 26, 2006 to January 22, 2009. This dataset has its limitations. First, in Facebook users have many interaction possibilities such as messaging, applications, photo uploads, and chat. From this dataset, only wall posting is available. Secondly, only users with public profiles were visible by the crawler. We consider this dataset to be representative of Facebook in New Orleans because it covers the majority of Facebook users (about 67%) there.

The vertices of Facebook graph represent the users and its edges represent interactions between two users according to the corresponding direction. The weights of edges represent the number of interactions between two users. We note that wall posts in Facebook act like a broadcast-style messager. The comments posted to a user's wall can be seen by other users visiting his profile. The visitors are able to reply to those comments or initiate a new discussion thread on a user's wall .

1.4 Research Context: Tribler

Tribler [99] is a BitTorrent-based P2P client targeted at file-sharing and video streaming applications. It is being developed by the Parallel and Distributed Systems group of Delft University of Technology for research and experimentation in P2P systems. In Tribler, a user is able to share torrents as well as to create Youtube like channels, search content, and watch and playback video. The Tribler client is available for download at tribler.org and since its first release in 2006, it counts over a million downloads. Tribler provides various advanced features including a distributed service for content discovery supported by a dissemination and database synchronization protocol named Dispersy [128] and the Bartercast distributed reputation mechanism [43].

The goal of the Tribler project is the design of a fully distributed, anonymous, and user friendly content distribution platform. Tribler evolves continuously and many researchers have contributed to it. Capotă et al. [26] have developed methods for resource allocation in multimedia communities. Zeilemaker et al. [127] introduced a privacy preserving semantic overlay. Petrocco et al. [95] have analyzed the performance of Libswift, a transport-layer protocol for P2P streaming. Lu Jia et al. [77] proposed a distributed method to estimate the strength of user interactions. D'Acunto et al. [38] analyzed strategies for peer selection and piece selection in BitTorrent-like Video-on-Demand systems.

1.5 Problem Statement

The problem of using relevant and trustworthy information imposes different challenges on the computation of reputations, the storage of information, and its collection, which are the three components of decentralized reputation systems. Below we describe these challenges of each component separately.

1.5.1 Computation of Reputation

The computation of reputations is typically based on one of two widely used methods: the max-flow algorithm or random walks. The *max-flow algorithm* [35] is at the core of many reputation systems such as Bazaar [98], Bartercast [43] and the system proposed by Feldman et al. [53]. Max-flow is resilient to misreporting by nodes who may report fake interactions to increase their reputations. However, it is computationally intensive and unable to deal with the sparsity of user interactions. As a result, the computed reputations do not represent accurately users' behaviors. In this context, the following research question arises:

How can we increase the accuracy of max-flow computation in an efficient way? Using the most central node of the interaction subgraph as a start point of the max-flow algorithm, instead of the local peer i, increases its accuracy on the computed reputations, since the majority of nodes is reachable in a short distance by the most central nodes. However, identifying the most central nodes in a large network is computationally expensive due to the involvement of the all-pair shortest path problem. Furthermore, as the network evolves over time, the most central node has to be recomputed periodically. In this context, the usage of centrality algorithms in reputation systems is prohibitive. Designing computationally efficient centrality algorithms for evolving graphs is vital for applying centrality algorithms in reputation systems.

Random walk-based algorithms are widely used to compute the reputations of nodes in a network according to the probability of visiting each node in a random walk. Many widely used reputation and recommendation systems such as EigenTrust [78], PageRank [92], and TrustRank [69], have at their core random walks. The main intuition behind those algorithms is that interactions should not contribute equally to reputations but interactions with highly reputed nodes should contribute more. Random walk-based reputations accurately represent the behavior of nodes but they are vulnerable to a great range of sybil attacks, such as spam, link farming, and collusion. Thus, for random walk-based reputations, we need to address the following research question:

How can we use trustworthy information for random walk-based computations of reputations? In random-walk based computations, nodes visited during a random walk treat all their neighbors equally, ignoring any properties they may have. This makes

random walks very vulnerable to various self-serving strategies such as free-riders, and self-promoting strategies such as sybil attacks. The defense schemes against sybil attacks can be categorized as sybil detection and sybil tolerance [116]. Sybil detection schemes label nodes as malicious or honest and exclude the nodes labeled as sybils. These schemes run the risk of false positives, honest users misclassified as malicious, and false negatives, malicious users misclassified as honest. False positives are problematic for a system [116] because they affect negatively user experience and threaten their trust towards the system. Sybil-tolerant schemes such as [25], [39] simply bound the gain of an attacker by leveraging the network structure.

1.5.2 Storage of Information

As users cannot store the complete history of user interactions due to their large numbers, and only the most relevant information can be maintained in a user's database, the following research question arises:

How can each node preserve the most relevant information in its database? The amount of historical information on the interactions maintained by each node affects the performance and the characteristics of the reputation system. Networks such as popular online markets and social networks consist of hundreds of thousands or even millions of active users and thus, using the complete history for computing the reputation of nodes is prohibitive due to its resource requirements. Particularly in decentralized systems, such as file-sharing P2P systems, the available resources at nodes are limited and thus, only scalable solutions can be applied. Furthermore, a long-term history allows previously well-behaved nodes to exploit their good reputations by acting maliciously [53, 84, 121]. To reduce storage and computation overhead, we need to find a scheme for reducing the amount of history of interactions maintained at each node.

1.5.3 Collection of Information

Information spread has to provide each user with relevant and trustworthy information.

How can each node collect relevant and trustworthy information? The collection of information directly affects both the quality of user reputations and the cost of the reputation system [42]. The resource limitations, and the lack of any centralized management challenge the collection of relevant and trustworthy information. The most widely used protocols for spreading information, epidemic protocols, are vulnerable to a great range of attacks and fail to provide relevant information. In epidemics, users blindly store and process information about other users that they will never interact with or information that contributes too little to the computation of reputations. Unlike epidemics we need to design a collection mechanism that is resilient to attacks, scalable, and provides each user

with relevant information.

1.6 Contributions and Thesis Outline

Addressing the four research questions stated in Section 1.5, the contributions of this thesis are as follows.

Efficient approximate computation of centrality (Chapter 2) Exploring the evolution of the most central nodes in growing synthetic and real-world interaction graphs, we observe that in most reputation networks the most central nodes tend to keep their central position over time. We assess their stability in our graphs, concluding that in scale-free and real-world graphs we do not need to recompute them often and so we can reduce the computational cost. Secondly, to further reduce the computational cost, we evaluate three approximation methods proposed in the literature in terms of their ability to identify the most central nodes. We conclude that in real-world and scale-free graphs approximative methods are highly accurate, while in random graphs it is harder to identify the most central nodes due to their structural properties. Finally, we assess the quality of the computed reputations when using these three approximations. This chapter is largely based on our paper:
Dimitra Gkorou, Johan Pouwelse, and Dick Epema, Betweenness centrality approximations for an internet deployed p2p reputation system, *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (HotP2P) 2011*

Leveraging node properties for robust random-walks (Chapter 3) We show that the properties of a node such as its activity and its position on the graph, indicate accurately its trustworthiness, and that random walks exploiting these properties are more resilient against attacks than simple random walks. Through extensive analysis, we identify the properties of nodes indicative of their trustworthiness and we bias random walks towards the most reliable nodes. Each node initiates its own random walks and computes its own personalized reputations for the other nodes. We consider the most common form of self-serving misbehavior, which is a lack of cooperativeness as exhibited by free-riders, who passively abuse the system by consuming its resources without contributing to it. In a self-promoting strategy, nodes try to falsely increase their reputations using a variety of techniques. From the self-promoting strategies, we consider only the sybil attack since most self-promoting strategies can be seen as special cases of it. We evaluate biased random walks in the face of free-riding and sybil attacks in growing synthetic and real-world graphs. We show that biased random outperform significantly simple random walks against attacks in all our graphs. Furthermore, we observe that the properties indicative of the trustworthiness of nodes depend on the structure and the construction process of the corresponding graph. This chapter is largely based on our paper:
Dimitra Gkorou, Tamás Vinkó, Johan Pouwelse, and Dick Epema, Leveraging node prop-

erties in random walks for robust reputations in decentralized networks, *IEEE International Conference on P2P Computing 2013*

Removing the least relevant parts of the history of interactions (Chapter 4) In order to reduce the history of interactions, we use only a subset of the complete history to approximate reputations. We model the interactions of the *complete history* of a network as a growing graph with the nodes of the network as its vertices and the interactions between pairs of nodes as its edges, and the corresponding *reduced history* as a subgraph of the complete history. The reduced history is derived from the complete history by deleting the least important edges and nodes. We define the importance of a node according to its age, its activity level, its reputation, and its position in the graph, while the importance of an edge is defined according to its age, its weight, and its position in the graph. Then we evaluate our approach using synthetic and real-world graphs for both max-flow and random walk based computations of reputations. We demonstrate that the performance of the reduced history depends on the topology of the complete history of interactions and the reputation algorithm. Finally, we conclude that the reduced history can be applied in a large range of networks. This chapter is largely based on our paper:
Dimitra Gkorou, Tamás Vinkó, Nitin Chiluka, Johan Pouwelse, and Dick Epema, Reducing the history in decentralized interaction-based reputation systems, *IFIP International Networking Conference 2012*

Collecting trustworthy and relevant parts of the history of interactions (Chapter 5) We observe that node tends to interact more often with nodes close to them due to the existence of highly connected nodes that keep the network together. As a result, a node does not need to collect the history of interactions of all the other nodes but only of those close to it. Furthermore, frequent interactions of high strength among nodes can be interpreted as trust. Based on these observations, we propose a decentralized algorithm based on random walks where each node uses its own part of the history of interactions to navigate in the network and collect parts of the histories of interactions of other nodes. We consider different types of random walks. We emulate the interactions in growing synthetic and real-world graphs and we explore the efficiency of these random walks in terms of scalability, relevance of collected information and robustness against sybil attacks. This chapter is largely based on our paper:
Dimitra Gkorou, Johan Pouwelse, and Dick Epema, Trust-based collection of information in decentralized networks which is under review.

Conclusions (Chapter 6) In the final chapter, we summarise the main findings of this thesis and we provide suggestions for further study.

Chapter 2

Betweenness Centrality Approximations

In reputation mechanisms such as Bartercast [43] and MoB [27], each node computes the reputations of others by applying a max flow-based algorithm in its interaction subgraph. It has been shown that the computed reputations are more accurate when a node starts the computation of max-flow from the most central node in its interaction subgraph, instead of itself [41]. Betweenness Centrality (BC) is a well-known and effective metric for identifying the centrality of nodes in a network, but the cost of its computation in large networks, and of its periodic recomputation in dynamically growing networks, is prohibitive. The BC of a node in a network measures the proportion of shortest paths passing through that node, and it is a global characteristic of a node rather than a local one such as its degree. It has been studied extensively in the context of, for instance, social networks [91] and networks built by interactions of proteins [76] for detecting central nodes. Furthermore, it is at the core of the most promising community identification and clustering algorithms [60]. Viewing networks as communication networks, high values of BC indicate the nodes with the highest communication load, and so, BC can be used in flow-based reputation mechanisms like Bartercast and MoB. Previously proposed approximation methods for BC computation have only been designed for static networks [21], [58].

In this chapter, we address the problem of detecting central nodes efficiently using BC in large and growing networks. As an example of max flow-based reputation systems we use Bartercast. In Bartercast, each peer can increase the accuracy of its computed reputations by using the most central node in its local Bartercast graph, that contains its view on all upload and download activities in the system, as the initial point in the computation of reputations instead of itself [41]. Nevertheless, considering that networks such as peer-to-peer networks may consist of thousands or sometimes millions of users, it is prohibitive to compute the values of BC with the existing methods in these networks, even if they are sparse.

Many previous studies have attempted to propose methods for the efficient compu-

tation of BC, but these methods are designed for and evaluated on static networks only. Brandes and Pich [21] attempt to approximate the values of BC extrapolating from a subset of selected nodes, which results in an overestimation of the BC of the nodes close to the selected nodes. Geisberger et al. [58] try to eliminate the overestimation of the BC values of the nodes close to a selected node by applying a scaling factor that adjusts the BC value of a node according to its distance from the selected node. Bader et al. [28] propose a method based on sampling from a small subset of nodes for the estimation of the BC of one specific node in the network. Another category of previously proposed approximations target only a restricted type of networks, such as modular networks [10]. Although most real-world networks grow rapidly over time, all these approaches have been designed and evaluated only for static networks. This weakness of previous studies makes the efficient detection of central nodes using BC in large and dynamic real-world networks inapplicable. A recent study [80] focuses on identifying the most central nodes using restricted random walks. A set of prior works have proposed parallel implementation of BC [11, 83]. These approaches reduce the time of BC computation, however it cannot be used in networks such as peer-to-peer networks where there is no possibility of parallel computations.

Unlike the indicated approaches, we focus on efficiently approximating the ranking of the nodes according their BC values rather than on the exact values of BC in large and dynamic networks. We evaluate the accuracy and the cost of three ways of approximating BC in random and scale-free graphs. The main contributions of this chapter are as follows:

1. We explore the evolution of the BC values, and of the nodes with the highest BC values, in networks growing over time, observing that the nodes with high BC in Bartercast remain almost invariant, thus reducing the cost of BC recomputation.

2. We compare three BC approximation methods proposed in the literature evaluating their ability to identify the top-most central nodes of a network, since for most applications, including increasing the accuracy of reputation computation Bartercast, only the detection of a small number of the top-most central nodes is required.

3. We integrate the proposed three BC approximations into Bartercast and assess their consequences for the accuracy of the reputation computations.

2.1 Background

In this section, we present the Bartercast reputation system and Betweenness Centrality along with its computational issues.

2.1.1 The Bartercast Reputation Mechanism

Using Bartercast, each peer in Tribler computes the reputations of other peers based on the history of their upload and download activities. In Bartercast, when a peer exchanges content with another peer, they both store a *Bartercast record* with the amount of data transferred and the identity of the corresponding peer. Regularly, peers contact another peer to exchange Bartercast records using a gossip-like protocol. Therefore, each peer keeps a history not only of the interactions in which it was directly involved, but of interactions among other peers as well. To limit the effect of misreporting, peers are not allowed to forward the records they receive, they can only disseminate information about their own interactions. This *one-hop* limitation has a great influence on the dissemination of records and on the performance of Bartercast [41].

From the Bartercast records it receives, every peer i dynamically creates its interaction subgraph G_i. Each peer i computes its subjective reputation of every other peer j by applying the Ford-Fulkerson algorithm [35] to its current interaction subgraph to find the maximum flows f_{ij} from itself to j and f_{ji} in the reverse direction, and then taking $\arctan(f_{ji} - f_{ij})/(\pi/2)$. The use of the max-flow algorithm in Bartercast provides resilience to misreporting by some peers who may exaggerate their uploads to increase their reputations. The flow f_{ji} is limited to the sum of the weights of the in links of peer i, no matter what in same uploads peer j reports. The original max-flow algorithm by Ford-Fulkerson tries all possible paths from the source to the destination, but in Bartercast, in order to limit the computational cost, only paths of length at most 2 are considered.

A peer i may not have a very central position in its own interaction subgraph G_i. As a consequence, the accuracy and the coverage of the reputations it computes may be low. Defining the *objective reputation* of peer j as $\arctan(u_j - d_j)/(\pi/2)$ with u_j (d_j) the total amount uploaded (downloaded) by peer j, the *accuracy* is the average error of the locally computed reputations of the peers with respect to their objective reputations.The function $\arctan$ in the computation of reputations emphasizes the differences of flows close to 0 (neutral reputation), so that newcomers with only a small contribution can achieve a good reputation value and participate in the system. Every reputation value is normalized with the factor $\pi/2$ so that it is in $(-1, 1)$. The *coverage* of the reputations computed by a peer is equal to the fraction of the peers in its interaction subgraph for which the local reputations it computes are non-zero. Replacing the local peer i by the peer with the highest BC in the graph G_i as the *start point* of max-flow, through which more of the "flow" in G_i passes, leads to more accurate reputations, and a larger coverage. However, the computational cost of BC makes its application in Bartercast prohibitive.

2.1.2 Betweenness Centrality

Although BC is a powerful metric for identifying the most central nodes in a network, its computational cost is high. BC was introduced by Anthonisse [7] and Freeman [55] in the context of real social networks as a measure of the influence of an individual over the information flow in the network. The BC of a node v, denoted by $C_B(v)$, is defined as

$$C_B(v) = \sum_{v \neq s,t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}},$$

where $\sigma_{s,t}$ is the number of shortest paths from node s to node t and $\sigma_{s,t}(v)$ is the number of such paths that contain node v. The basic assumption of this definition of BC is that the information flows in the network along the shortest paths.

The involvement of the all-pair shortest path problem (APSP) in the computational core of BC makes it expensive. A fast algorithm proposed by Brandes [20] computes BC with time complexity $O(nm)$ for unweighted graphs and $O(nm + n^2 \log n)$ for weighted graphs, where n and m denote the number of nodes and edges in a graph, respectively. This algorithm explores and counts the shortest paths at the same time using Breadth-First Search (BFS) for the unweighted case and Dijkstra's algorithm for the weighted case, and then aggregates efficiently the path counts. However, this algorithm is still very expensive for large networks.

2.2 Dataset and Evaluation Metrics

We analyze the dynamics of BC and its approximations using both synthetic and real-world graphs. Among all the different models for generating synthetic graphs, we select the two most widely used models: random and scale-free graphs. Our real-world graph is derived from Bartercast, the reputation mechanism used in Tribler.

2.2.1 Random Graphs

According to the Erdős-Rényi model [47], a random graph, denoted $G(n, p)$, is composed of n nodes and each potential edge connecting two nodes occurs independently with probability p. In a random graph, nodes exhibit homogeneous statistical properties such as their degree. Note that (an instance of) $G(n + 1, p)$ can be obtained from $G(n, p)$ by adding one node and all of its potential links with probability p. For our experiments, we have generated a growing random graph R, of which we will consider its instances R_t, $t = 1, 2, \ldots, 20$, with R_t having $n_t = 1,000 \cdot t$ nodes. To make the results for the random graph and Bartercast graph presented later in the paper comparable, the probability p of

Table 2.1: Properties of the Bartercast graphs.

Graph	Date Collected	# Nodes	# Edges	power-law exponent
B_1	July 24, 2009	1592	2159	2.2209
B_2	August 6, 2009	1870	2582	2.2106
B_3	August 14, 2009	2037	2861	2.2041
B_4	August 26, 2009	2254	3210	2.1683
B_5	September 9, 2009	2408	3463	2.1234

each edge existing in R is computed according to the number of nodes and the number of edges in the Bartercast graph, and is equal to 0.0012.

2.2.2 Scale-free Graphs

As it turns out, many real-world networks are characterized by a heavy-tailed degree distribution following a power-law, i.e., their degree distribution is expressed as $P(k) \approx ck^{-\alpha}$, where $P(k)$ is the fraction of nodes of degree k, α denotes the power-law exponent, and c is a constant. The networks whose degree distribution follow a power-law are named scale free [14]. Examples of scale-free networks are the Internet Topology [51], protein-interaction networks [5] and many social networks [91]. In scale-free graphs, the BC distribution of the nodes follows a power-law [15]. As we will show in the next subsection, Bartercast can be modeled as a scale-free graph since its degree distribution and its BC distribution as well follow a power-law.

For our analysis, we create a scale-free graph S, using the preferential attachment model proposed by Barabási and Albert [14], according to which a new node joining the network has 3 edges, which are attached to nodes already existent with probability proportional to their degrees. Similarly to the random network, we consider 20 instances of S, denoted by $S_1, \ldots, S_{20}$, with S_1 consisting of $1,000$ nodes and each following instance including $1,000$ additional nodes. To approximate the properties of the graph built by Bartercast, each new node is added to the graph with two links adjacent to it.

2.2.3 Bartercast Graphs

We examine five instances of the Bartercast network, denoted by $B_1, ..., B_5$, the first of which was crawled on 24 July 2009 and the last on 9 September 2009. Each of these instances is an extension of the previous one. Since each of these graphs is disconnected, we proceed in the analysis using their largest connected components. In our analysis of BC in Bartercast, we do not consider the weights (but we do take into account the directions) of the edges, since we are interested in the most central node in terms of increasing the coverage, and thus the accuracy, in the computation of reputations.

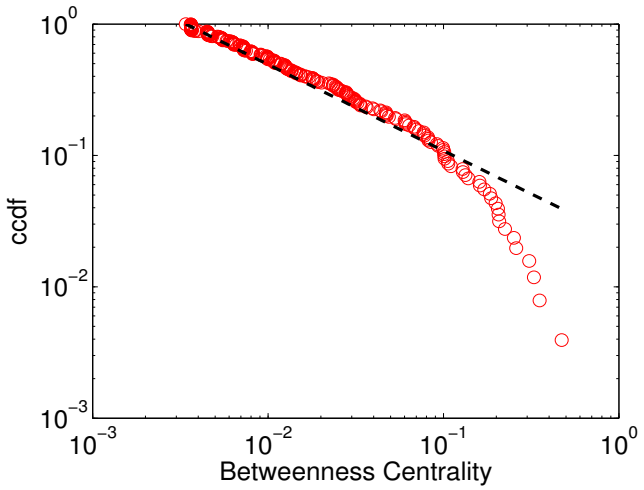
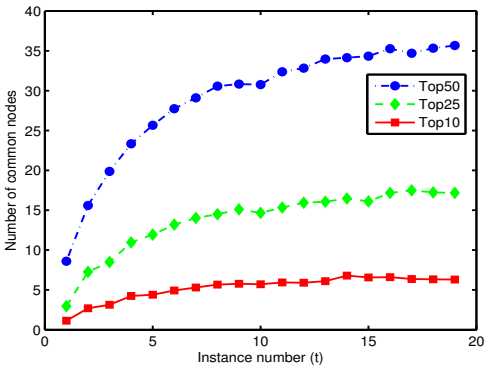


Figure 2.1: The log log plot of the complementary cumulative distribution function of the Betweenness Centrality in the Bartercast graph B_5.

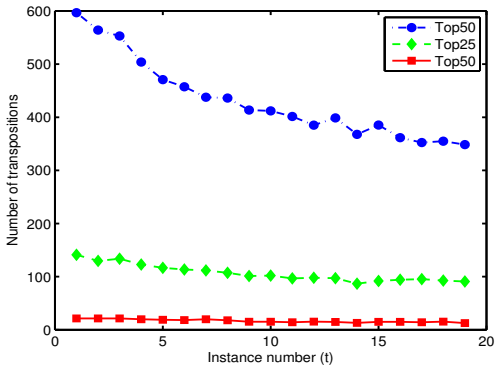
The distributions of the in-degree, the out-degree, and the total degree, that is, without considering the directions of the links, of the Bartercast graph all follow a power-law. In Table 2.1, we present the power-law exponents for $B_1, ..., B_5$ for the total degree. Our results show that the exponent decreases when the graph grows, but does not change significantly. Moreover, the BC distribution follows a power-law. Figure 2.1 exhibits the BC distribution for B_5 and the corresponding power-law distribution with exponent 2.0887. The observed cut-off on the plot of BC is caused by finite-size effects. The power-law exponents are computed as described in [34] using Kolmogorov-Smirnov statistics for the goodness of fit.

2.2.4 Evaluation Metrics

We focus on the detection of the top-l most central nodes in the graphs under consideration with l very small compared to the number of nodes, since for most applications, including the improvement of Bartercast using BC, only the detection of a small subset of the most central nodes is required. In our experimental analysis, we consider the sequences of the Unique Identifiers (UIDs) of the top-l most central nodes. We use two metrics to quantify the difference between two such sequences a and b of equal length. The *number of common nodes* of a and b is simply the number of UIDs that occur in both a and b, no matter what their positions. The *number of transpositions* between a and b is the minimal number of exchanges of neighbors needed in a to get all the common nodes in their correct positions in b (or vice-versa). In order to assess the stability of BC over time, we will apply these two metrics to the top-l most central nodes in successive instances of the graphs. In order to assess the quality of BC approximations, we will apply them



(a) The number of common nodes in the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of R_{t+1} and R_t.



(b) The number of transpositions between the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of R_{t+1} and R_t.

Figure 2.2: Comparison of the top-l ($l = 10, 25$ and 50) most central nodes between consecutive instances of the random graph R_t over time.

Table 2.2: The correlation between the degree and BC of nodes in the final instance of the random, scale-free and Bartecast graph.

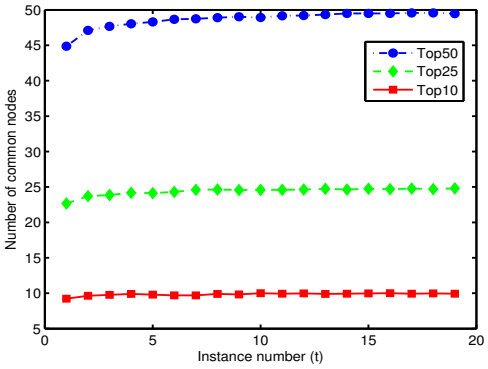
Graph	correlation
Random	0.99
Scale-free	0.82
Bartercast	0.72

to the sequences of top-l most central nodes obtained with exact BC and with the BC approximations.

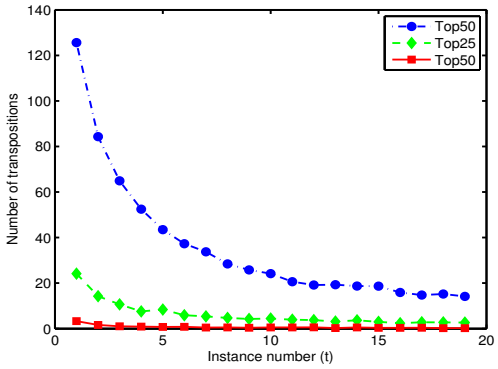
2.3 Evolution of Betweenness Centrality in Growing Networks

In dynamic networks, the BC values of all the nodes in the network potentially have to be recomputed for every change in the network. Intuitively however, since most networks have a specific structure and a specific way of construction, the central nodes do not change quickly as the network grows over time. If that is the case, the BC values do not have to be updated very often when the network grows. In this section, we explore the change in central nodes in both random and scale-free graphs over time.

In scale-free graphs, BC is highly correlated with the degree of nodes, as we observe in Table 2.2. When using preferential attachment for constructing scale-free networks, the new-coming nodes prefer to be attached to nodes with high degrees, which play the role of "hubs." Furthermore, the scale-free model as proposed by Barabási and Albert does

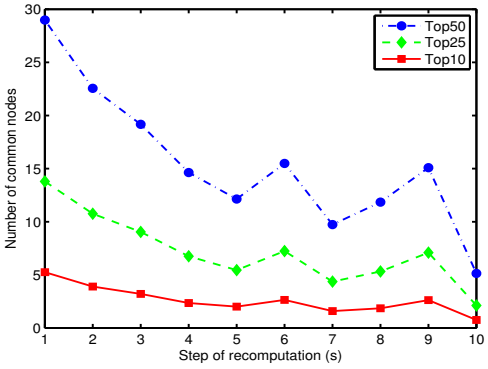


(a) The number of common nodes in the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of S_{t+1} and S_t.

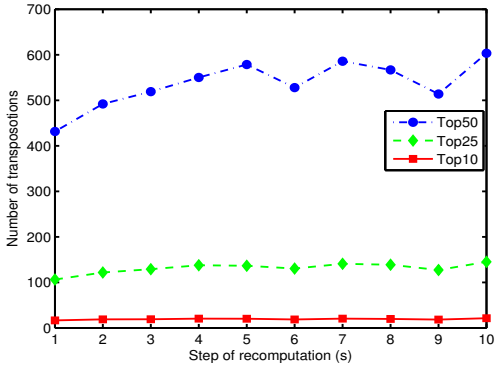


(b) The number of transpositions between the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of S_{t+1} and S_t.

Figure 2.3: Comparison of the top-l ($l = 10, 25$ and 50) most central nodes between consecutive instances of the scale-free graph S_t over time.



(a) The number of common nodes in the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of R_t and R_{t+s}.



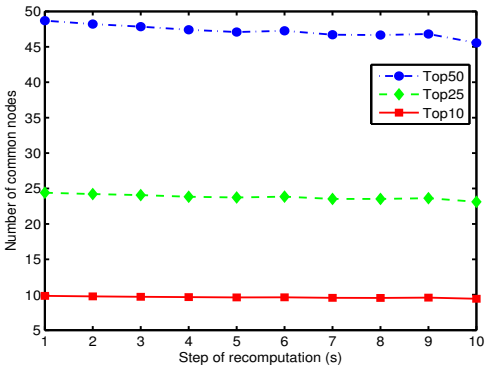
(b) The number of transpositions between the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of R_t and R_{t+s}.

Figure 2.4: The accuracy achieved by recomputing BC in the random graph every $s = 1, 2, ..., 10$ instances.

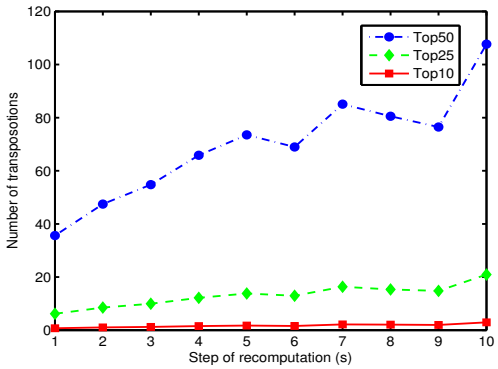
not exhibit clearly separated communities. As a result, "hubs" tend to hold their central position as the network grows. On the other hand, in random graphs, each pair of nodes is connected with some fixed probability p, and so, all nodes have similar properties of connectivity. Consequently, it is harder to predict the change in central nodes as the graph grows. The correctness of this intuition is demonstrated in Figures 2.2 and 2.3, which exhibit the difference between the sequences of the UIDs of the $10, 25$ and 50 most central nodes of the consecutive instances of R_t and S_t, respectively. The maximum possible numbers of transpositions needed for the sequences of the top-l ($l = 10, 25, 50$) most central nodes are 55, 325, and 1275, respectively, when they have all nodes in common and they are in reverse order. Nevertheless, even in random graphs, the sequences of the most central nodes exhibit stability over time to some extent, as shown in Figure 2.2. All the presented results for the random and scale-free graphs in the paper are the average of 50 independent experiments.

The stability of the most central nodes in scale-free graphs implies that BC does not have to be recomputed often. Figures 2.4 and 2.5 exhibit the accuracy achieved when BC is recomputed every $s = 1, 2, \ldots, 10$ instances in the random graph and in the scale-free graph, respectively. For scale-free graphs, even if BC is only recomputed every 6 instances, i.e., after the entrance of $6,000$ new nodes in the graph, 95% of the top-l most central nodes is correctly identified for the different values of l, and the number of transpositions required is less than 4% (Figure 2.5). However, this approach is less accurate for random graphs. Computing BC every instance (after the entrance of $1,000$ new nodes) in a random graph results in the correct identification of 52.6%, 55% and 58% of the $10, 25$ and 50 most central nodes, respectively, while the percentage of the corresponding transpositions needed is 30%, 33% and 34% (Figure 2.4).

Next we present the stability of the most central node in the growing graphs we study. We assign the stability value of 1 to an instance of a growing graph if the most central node of the previous instance is equal to that of the current instance, and 0 otherwise. In Figure 2.6, we show the average stability of the most central node across 50 experiments. In the scale-free graph, the most central node remains almost invariant and is one of the first added nodes in the graph, which is a consequence of the construction process of scale-free graphs: old nodes are well-connected and are central to the network, younger nodes are less well connected and are at the fringes of the network. On the other hand, in random graphs, the most central node is not stable but its stability increases over time. Furthermore, it can be one of the newly added nodes and its centrality is not related to the time of its stay in the system. In the Bartercast graph, the most central node holds its position over time like in scale-free graphs.



(a) The number of common nodes in the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of S_t and S_{t+s}.



(b) The number of transpositions between the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of S_t and S_{t+s}.

Figure 2.5: The accuracy achieved by recomputing BC in the scale-free graph every $s = 1, 2, ..., 10$ instances.

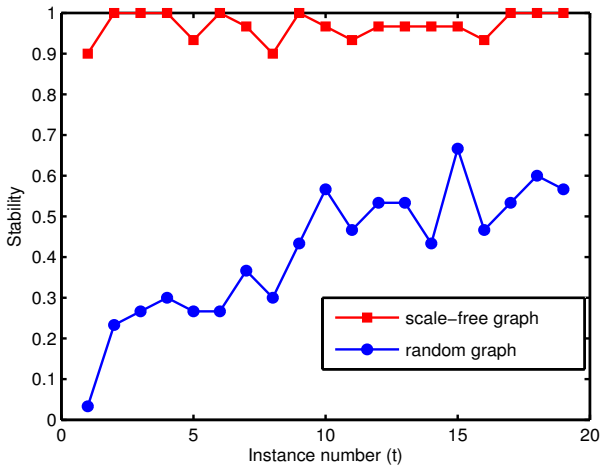


Figure 2.6: The average stability of the most central node between consecutive instances of the random graph R_t and the scale-free graph S_t.

2.4 Approximations of Betweenness Centrality

Even if we don't have to recompute BC often, its computational cost remains prohibitive for large graphs. In this section, we present the three most efficient and general approximation methods of BC proposed in the literature. Next we evaluate these methods in random, scale-free, and Bartercast graphs.

2.4.1 Definition of BC Approximations

As the first approximation method, we consider a method proposed by Brandes and Pich [21] called *Pivot-BC (P-BC)*, which is based on extrapolation from a small subset of selected nodes, the pivots, to approximate the BC of each node in a network. The contribution of a node s to the BC of a node v, $\delta_s(v)$ is:

$$\delta_s(v) = \sum_{t \in V, v \neq s,t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}},$$

where V is the set of nodes in the network. Then, with K pivots $\{s_1, \ldots, s_K\}$, the BC is approximated by

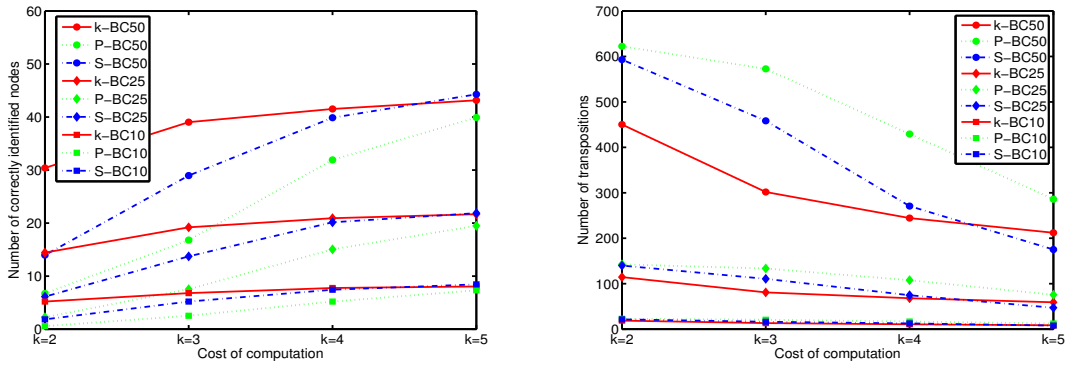
$$C_P(v) = \sum_{i=1}^{K} \frac{n}{K} \delta_{s_i}(v).$$

The selection of pivots in a random way is proven to perform better than more sophisticated strategies [21]. This extrapolation-based method approximates BC with cost $O(Km)$ for unweighted graphs and $O(Km + Kn \log n)$ for weighted graphs. However, this method is not accurate enough because it overestimates the BC of unimportant nodes close to the pivots.

To adjust this bias towards the nodes close to the pivots, Geisberger et al. [58] introduce a scaling factor that modulates the BC value of a given node according to its distance from the pivots, we denote this method by *Scale-BC (S-BC)*. According to linear scaling, the contribution of shortest paths starting from a pivot in the BC values of the other nodes depends linearly on the distance of each node from the pivot. In this case, the contribution of a node s to the BC of a node v, $\delta_s(v)$ is computed as:

$$\delta_s(v) = \sum_{t \in V, v \neq s,t} \frac{\mu(s,v)}{\mu(s,t)} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}},$$

where $\mu(u,v)$ is the distance from node u to node v. To compute the total BC value of a



(a) The number of common nodes in the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of the exact computation of BC and the approximations k-BC, P-BC and S-BC.

(b) The number of transpositions between the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of the exact computation of BC and the approximations k-BC, P-BC and S-BC.

Figure 2.7: Comparison of k-BC, P-BC and S-BC in the random graph R_{20}.

given node v, the contributions of the pivots are accumulated:

$$C_S(v) = \sum_{i=1}^{K} \frac{n}{K} \delta_{s_i}(v).$$

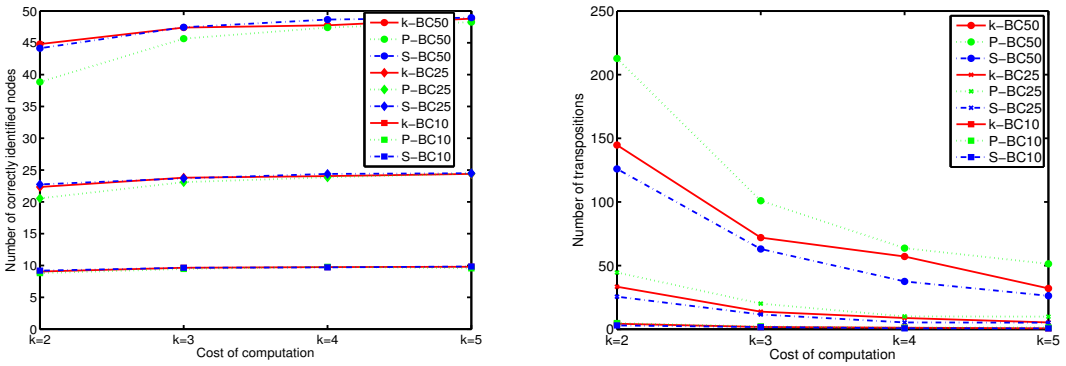
In bisection scaling, the contribution of the shortest path from a pivot counts only to the second half of the path, while this contribution is considered to be 0 for the nodes lying on the first half of the path (close to pivots). Bisection scaling is designed for networks with unique shortest paths and it performs better than the other estimators in this type of networks. However, it can be very slow in the general case with inaccurate results. Therefore, we do not consider this scaling method in our experiments. We assume the computational costs of the simple extrapolation method and of linear scaling to be equal.

As the third approximation method, we consider k-Betweenness Centrality (k-BC) or Bounded Distance Betweenness Centrality. In networks in which long paths are not used, such as wireless sensor networks, k-BC has been defined to measure centrality [19]. The basic idea of k-BC consists in reducing the number of shortest paths counted and it is defined exactly as BC but only exploring paths of lengths at most equal to k. The k-BC of a node v is defined as

$$C_k(v) = \sum_{s,t \in V : d(s,t) \leq k} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}},$$

where $d(s,t)$ is the distance from node s to node t.

For $k = n - 1$, k-BC is equivalent to standard BC. We use this metric as an approximation of standard BC. We can easily compute k-BC by the depth-limited algorithm or by imposing to classic BFS to stop at depth k. Let the *k-neighborhood* of node v of a



(a) The number of common nodes in the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of the exact computation of BC and the approximations k-BC, P-BC and S-BC.

(b) The number of transpositions between the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of the exact computation of BC and the approximations k-BC, P-BC and S-BC.

Figure 2.8: Comparison of k-BC, P-BC and S-BC in the scale-free graph S_{20}.

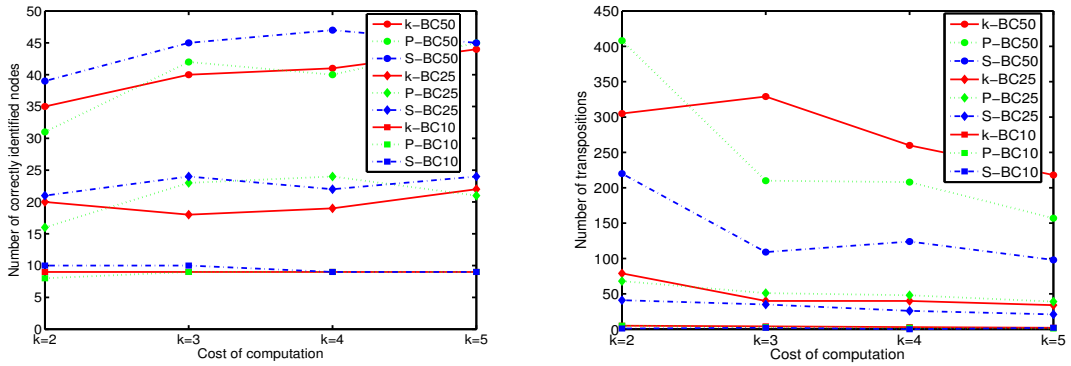
graph G be the subgraph of G containing all nodes at a distance of at most k from v, and all connections among these nodes. Then the cost of k-BC is $O(nm_k)$ for unweighted graphs and $O(nm_k + nn_k \log n_k)$ for weighted graphs, where m_k (n_k) is the average of the number of edges (nodes) present in the k-neighborhood of nodes.

2.4.2 Experimental Results

In this section, we compare k-BC with P-BC and S-BC in the random graph R_{20}, the scale free graph S_{20}, and the Bartercast graph B_5 in terms of the top-l ($l = 10$, 25 and 50) most central nodes identified in each graph instance. To make these approximation methods comparable, the number of pivots for P-BC and S-BC are selected according the cost of the computation of k-BC for the different values of k. Thus, all three approximation methods have the same cost. In the presentation of the results, the different values of k correspond to the different computational costs.

The three approximation methods exhibit similar results for the scale-free graph S_{20}, achieving a high accuracy in identifying the sequences of the top-l ($l = 10$, 25 and 50) most central nodes (see Figure 2.8). The lowest accuracy is observed for P-BC, while S-BC achieves the best accuracy. Nevertheless, the accuracy of k-BC is close to that of S-BC, and in most cases their results overlap. On the contrary, for the random graph R_{20}, all the approximation methods have a lower accuracy (see Figure 2.7). For k equal to $2, 3$ and 4, k-BC results in the best accuracy with significant difference from the other two methods, while for k equal to 5, S-BC achieves a slightly better accuracy. The worst accuracy is again obtained by P-BC.

This difference in the performance of the approximation methods is caused by the



(a) The number of common nodes in the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of the exact computation of BC in B_5 and the approximations k-BC, P-BC and S-BC.

(b) The number of transpositions between the sequences of the top-l ($l = 10, 25$ and 50) most central nodes of the exact computation of BC and the approximations k-BC, P-BC and S-BC.

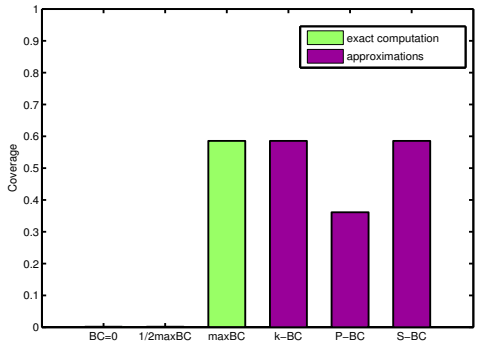
Figure 2.9: Comparison of k-BC, P-BC, and S-BC in the BarterCast graph B_5.

structural properties of random and scale-free graphs. In scale-free graphs, the small number of nodes ("hubs") having high degrees attract the majority of links participating in the majority of shortest paths of the graph. As a result, these nodes can be identified more easily since they are connected to every other node within a short distance. On the other hand, in random graphs the most central nodes have statistical properties similar to the other nodes and thus it is harder for the approximation methods to identify them. In the Bartercast graph B_5, all the approximation methods achieve a satisfying accuracy (see Figure 2.9). S-BC outperforms the other methods, particularly in the detection of the top-50 most central nodes.

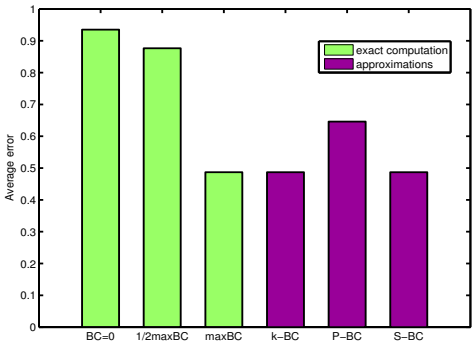
2.5 Increasing Efficiently the Accuracy of the Computation of Reputation in Bartercast

In this section, we integrate the three BC approximation methods presented in Section 2.4 into Bartercast and evaluate their effect. Under this modification, each peer willing to interact with other peers first identifies the node with the highest BC in its interaction subgraph using one of these approximations, and then applies the max-flow algorithm with that node as the start point. We evaluate the effect of this modification in terms of the average error and the coverage (see Section 2.1.1). As a matter of fact, because we assume full gossip here (see Section 2.1.1), we assume all local BarterCast graphs to be identical, equal to the single complete BarterCast graph built from all BarterCast records sent in the system.

In order to assess the need for using a node with a high BC value, in our experimental



(a) The fraction of nodes covered by the different start points in the complete Bartercast graph.



(b) The average error in the computed reputations in the complete Bartercast graph for the different start points.

Figure 2.10: Comparison of the coverage and the average error when the start point in max-flow is a node with BC equal to 0, the node with BC equal to 50% of the maximum BC, the most central node, and the most central nodes as computed by k-BC, P-BC, and S-BC, respectively.

analysis, we also choose a few other nodes as the start points for max-flow: a node with BC equal to 0 (since in the interaction subgraphs in Bartercast there are several of these, we choose one of them randomly), the node (indicated by `1/2maxBC`) with BC equal to 50% of the maximum BC, and the most central node, all according to the exact computation of BC. The number of hops used by the max-flow algorithm is equal to k in the computation for k-BC, and the number of the pivots for S-BC and P-BC is again chosen according the cost of k-BC. In the experiment on which we report here (see Figure 2.10), the value of k is equal to 3.

As shown in Figure 2.10, starting max-flow from a node with BC equal to 0 or from the node `1/2maxBC` leads to disastrous results, while using the node with the highest exact BC in the computation of the reputations gives good coverage and decent accuracy. The same performance is observed when starting max-flow from the most central node as computed by k-BC and S-BC, because they correctly identify the most central node. In contrast, P-BC fails to identify the most central node correctly and thus exhibits poorer results than the other two approximations. We observe similar results when k is equal to 2, with k-BC and S-BC still correctly identifying the most central node but P-BC not doing so. Using k equal to 4 or 5, P-BC does succeed in identifying the most central node, and then it results in the same improvement in accuracy as exact BC, although then the computational cost of the approximations increases.

2.6 Conclusion

Using the node with the highest Betweenness Centrality (BC) in the computation of reputations in Bartercast increases their coverage and accuracy, but BC is expensive to be computed. In this chapter, we have proposed and experimentally evaluated two different approximate approaches for computing BC, exploring both theoretical graph models (random and scale-free) and the graph derived from the actual operation of the Bartercast reputation system. For growing networks, our first approach relies on the observation that the nodes with high BC in real-world networks remain almost invariant over time. For large networks, our second approach consists in assessing three approximation methods, k-BC, P-BC, and S-BC, in terms of their ability to identify the top-most central nodes. We conclude that in scale-free graphs, the BC approximations are efficient and highly accurate, while in random graphs, due to their structural properties, it is harder to identify the most central nodes. In the graph derived by Bartercast, these approximations exhibit similar performance and are adequately accurate. Finally, we have integrated the approximation methods for BC into the computation of reputations in Bartercast, and we found the accuracy and the coverage of the computed reputations for two of these methods to be excellent.

Chapter 3

Leveraging Node Properties in Random Walks for Robust Reputations

For the computation of reputations, random walks constitute the core of the most widely used algorithms such as EigenTrust [78], PageRank [92], and TrustRank [69], because of their simple decentralization, their ability to take advantage of the sparsity of networks, and their computational efficiency. Even though random walks have these useful properties, they can be easily exploited by uncooperative and malicious nodes which abuse other nodes using various self-serving and self-promoting strategies. Designing random walk-based reputation systems which are resilient against malicious nodes is very important particularly for decentralized reputation systems, where nodes have only a partial view of the system. Traditionally, nodes visited during a random walk treat all their neighbors equally, ignoring any properties they may have. Nevertheless, properties of nodes, such as their age, may be indicative of their reliability, and thus, by integrating them into random walks, we can design more robust reputation systems.

Random walk-based reputation algorithms compute the reputation of a node as the probability of visiting that node in a random walk. In most implementations, random walks try to achieve resilience against uncooperative and malicious nodes based on a uniformly random selection of the next node to be visited. As a result, such *simple* random walks are vulnerable to many types of self-serving and self-promoting strategies [68, 74]. In a self-serving strategy, nodes abuse the system by first behaving properly for some time, and by then letting their reputations decrease in order to achieve a short-term gain [73]. We consider the most common form of self-serving misbehavior, which is a lack of cooperativeness as exhibited by free-riders, who passively abuse the system by consuming its resources without contributing to it. In a self-promoting strategy, nodes try to falsely increase their reputations using a variety of techniques such as web spamming and link farming [68], collusion [74], and Sybil attacks [89]. From the self-promoting strategies, we consider only the Sybil attack since most self-promoting strategies can be seen as

special cases of it. In a Sybil attack, a malicious node boosts its reputation by controlling fake identities (its sybils), which report fake interactions with each other and with the malicious node.

In this chapter, we show that the properties of a node such as its age and its activity level, accurately indicate its reliability, and that random walks exploiting these properties are more resilient to malicious nodes than simple random walks. We model reputation systems in growing synthetic random and scale-free graphs, and in real-world graphs derived from the Bartercast reputation system [43] which is used in the BitTorrent client Tribler [99], from the citation network of Physical Review E journal, and from Facebook [115]. Each node in a graph initiates its own random walks and computes its own personalized reputations for the other nodes. Due to the size of our graphs, it is prohibitive to evaluate our biased random walks after the entry of each new node or edge and so, we use properly chosen time windows.

The main contributions of this chapter are as follows:

1. We introduce the node properties indicative of their behavior and we bias random walks with those properties.

2. In the case of uncooperative nodes, we evaluate biased random walks in growing graphs based on the observation that the ranking of nodes according to their reputations is more important than the actual values of reputations.

3. In the case of sybil attacks, we evaluate the escape probability of a random walk to the sybil area depending on the number of the attack edges, since it has been already shown that the effectiveness of a sybil attack depends on this number [125].

3.1 Problem Statement

Random walk-based algorithms have been widely used for decentralized reputation systems since they have a low computational cost and resilience against noisy input [59]. However, they are vulnerable to malicious behaviors. Most implementations of random walks ignore the structural properties of nodes such as their centrality, clustering, and age, while these properties are indicative of their reliability. Our goal is to identify these properties of nodes and integrate them into random walks for building reputation systems that are more robust against exploitations. According to our approach, each node initiates its own random walks and we do not assume the existence of pre-trusted nodes. Therefore, our approach is suitable for decentralized networks.

3.1.1 Motivation

There are many exploitations of random walks in decentralized reputation systems. In this chapter, we study the two most common exploitations: the uncooperative nodes and the Sybil attacks. Nevertheless, our method can be generalized to improve the robustness against other exploitations as well.

In reputation systems such as online markets or collaboration networks (e.g., eBay, eLance and Wikipedia), where nodes have to respect constantly the protocol, many users can be exploited by traitors or uncooperative nodes. Even simple uncooperative nodes can degrade significantly the system's performance. For instance, Sopcast, a P2P live streaming network, has on average around 87% uncooperative nodes degrading its performance [66]. Reputations predicting the behavior of nodes give incentives to the nodes to behave continuously according to the protocol. Computations of reputation predicting the behavior of nodes have been proposed in the context of Wikipedia authors [4], in P2P file-sharing systems [119] and P2P live streaming [66]. When using simple random walk, the predictive ability of the system is very low and some attempts to integrate properties of nodes have already improved it for the link prediction problem [81], [9]. In distributed systems and especially in unstructured P2P networks, biased random walks have been used for searching content [62], [104] but not in the context of reputation systems.

Reputation systems based on random walks are also sensitive to Sybil attacks [45]. Specially in the context of Pagerank, this observation is very common [30], [68]. Many users perform Sybil strategies, such as link farming [68] where fake links point to both the Sybils and the malicious node. In networks where the creation of links among nodes is easy, such as WWW and Facebook, random walk performs very poorly against Sybil attacks. Therefore, biased random walk with node properties indicating trust between two nodes decreases radically the effect of Sybil attack. A first study of the effect of biased random walks on algorithms against Sybil attacks explored the use of node similarity and strength of interactions [89]. Unlike [89], our study for sybil attacks evaluates the escape probability of random walks into the sybil area, and we use random walks with restarts biased with structural and temporal parameters.

3.1.2 Definitions and Network Model

We model a reputation network as an interaction graph $G = (V, E)$ A weighted edge $e_{ij} \in E$ connects two vertices $i, j \in V$ in the direction $i \to j$ with weight w_{ij}. Depending on the context, weights may represent the amount of data transferred across edges (in a P2P network), or the number of citations among authors (in a citation network). Computing the reputation of nodes with a random walk-based algorithm implies that the past interactions between nodes are interpreted as trustiness, in a similar way that links between web pages are interpreted as votes in a search engine like Google. The transition matrix P of a

random walk in G is defined by $p_{ij} = w_{ij} / \sum_{k \in N_i} w_{ik}$, where N_i denotes the set of neighbors of node i.

We will use random walks with restarts [110], which means that each node visited by a random walk decides to direct the random walk back towards its initiator with the *teleportation probability* α. Then, the transition matrix becomes

$$P' = (1 - \alpha)P + \alpha \mathbf{1}$$

where $\mathbf{1}$ is the matrix with all its entries equal to 0 except for the elements of the column corresponding to the initiator, which are equal to 1. A random walk with restarts is *personalized* and represents better the inherent trust in a network, since each node trusts itself more than the other nodes and its trust towards the other nodes decreases with the increase of their distance. The vector π_i with the reputations computed by node i is the solution of the eigenvector equation $\pi_i = \pi_i P'$.

In an *unbiased random walk (simple RW)*, the weights w_{ij} represent simply the adjacency of the nodes in G, that is $w_{ij} = 1$ if $e_{ij} \in E$ and $w_{ij} = 0$ otherwise. In a *biased random walk (bRW)*, the weight w_{ij} is equal to some actual weight of the corresponding edge. In that case, we have to assign a weight w_{ij} to each edge e_{ij} in G in order to have a bRW visit more often the most reliable nodes. For each edge e_{ij}, we consider a vector ψ_{ij} with the values of properties of node j as perceived by node i, and we combine its elements to one weight $w_{ij} = f(\psi_{ij})$ for some function f. The function f can be defined by using the normalized product of the node properties or it can be learned in a supervised way. We refer to the former walk as *naive RW (nRW)* and to the latter walk as *supervised random walk (sRW)*.

For training our sRW, we use a slightly modified version of the method described in [9] adapted to our problem. We assume that the function f has an exponential form $f = \exp(u_i \psi_{ij})$, where u_i is the vector learned by node i with the coefficients of ψ_{ij}. We formalize the problem of determining the vector u_i as a nonlinear optimization problem:

$$\min_{u_i} \|u_i\|^2 + \sum_{d \in D_i, l \in L_i} \frac{1}{1 + \exp(-(\pi_i(l) - \pi_i(d)))}$$
$$\text{s.t. } \pi_i(j) = \sum_{k \in V} \pi_i(k) P'_{kj} \quad (\forall j \in V),$$

where D_i and L_i are the sets of the top-30 best-behaved nodes and the top-30 worst-behaved nodes from the perspective of a node i, respectively. This objective function is highly multimodal, so an optimizer can easily get trapped in a local minimum. In order to avoid this, we let every node perform the following iterative process: we make only a small number of steps (up to 5) with the optimizer, then we compute the values of w_{ij} using the current value of u_i and solve the equation $\pi_i = \pi_i P'$ with power-iteration;

using these u_i and π_i values as starting points we get back to the optimization problem again. We proceed such iterations until the solution vector u_i converges. Unlike in [9], we implemented our optimization procedure in the AMPL modeling language [54]. For solving the above defined optimization problem we used IPOPT [118], which is able to deal with constrained problems and uses the automatic differentiation feature of AMPL.

Our computation of reputations can be easily implemented in decentralized reputation systems where each node stores locally its own view of the reputation network, such as Bartercast [43], the system proposed by Piatek et al. [97], and MobID [102]. In these systems, when a node interacts with another node, they both store the weight of their interaction and the identity of the corresponding node. Nodes exchange information about their interactions using a gossip-like protocol. Based on its own interactions and the interactions gossiped about other nodes, each node builds locally its own partial view of the reputation network. Each node i performs the computation of π_i and the properties of other nodes on its own partial view.

For our analysis, we assume full-gossip in which nodes forward all their interactions, and eventually, their partial views converge to the global interaction graph G. In a real system, the partial views of nodes may not convergence to G due to their resource limitations or high churn. Nevertheless, the reputations, as computed by random walks with restarts, are only slightly affected. In random walks with restarts, an interaction between two nodes occurring in the neighbourhood of the initiator of a random walk, contributes more on the computed reputations and gossip protocols propagate fast information in the neighbourhood of a node.

3.2 Datasets

In order to evaluate bRWs, we consider synthetic and real-world graphs which are defined below. When a graph is not connected, we proceed in our analysis using its largest weakly connected component.

Both our synthetic and real-world graphs grow over time. During the construction of the synthetic graphs, in each time step, with probability p_c a new node enters the system, or with probability $1 - p_c$ already existing nodes interact and create new edges. The value of probability p_c depends on the dynamics of the system. Particularly, in highly dynamic systems the appearance of new nodes is dominant. For our synthetic graphs, we assume moderate system dynamics and so, we choose p_c equal to 0.5. Moreover, we allow the occurrence of multiple edges between a pair of nodes and we consider the number of occurrences of an edge as the weight of that edge. In real-world graphs, the addition of new nodes and edges is based on the timestamps available on the corresponding datasets and it is expressed in terms of actual time. In the synthetic graphs, no notion of actual time exists. For the construction of the synthetic graphs, time is divided into time steps

Table 3.1: The diameter, the average path length (L) and the clustering coefficient (cc) of the largest weakly connected component of our graphs.

Graph	# Nodes	# Edges	Diameter	L	cc
Bartercast	$10,364$	$44,796$	13	2.64	0.00074
Citation	$31,238$	$110,638$	15	7.66	0.20
Facebook	$63,392$	$1,545,309$	15	4.32	0.15

during which new edges and nodes are added.

A **random graph**, denoted by $R(n, p_r)$, is composed of n nodes, and each potential edge connecting two nodes occurs independently with probability p_r. We start from a single node, and in each time step, with probability p_c we add a node with each of its potential directed edges existing with probability p for some value of p, and with probability $1 - p_c$ we add pn_t directed edges adjacent to existing nodes chosen uniformly at random. In chapter 4, we show that $p_r \sim p/2p_c$ [64]. In our experiments, we use a graph $R(5000, 0.02)$.

Scale-free graphs, denoted by $S(m)$, are characterized by their degree distribution following a power law. We create a growing directed scale-free graph based on the BA model [14]. We start with a small seeding connected triangular graph, and in each time step, with probability p_c we add a node with m directed edges. The end point of each of these edges is adjacent to an already existing node i with probability $\Pi(i) = d_i / \sum_j d_j$, where d_i is the degree of node i. With probability $1 - p_c$ we add m directed edges, each of which is adjacent to an existing node i with probability $\Pi(i)$. In chapter 4, we show that S is scale-free with power-law exponent equal to $\gamma = 1 + 2/(2 - p_c)$ [64]. For our evaluation, we use a graph $S(3)$ of 5000 nodes.

The **Bartercast graph** is denoted by B. As a deployed system, the Bartercast graph has a high population turnover, and so, the derived graph consists of a dense core with very few long living and active nodes and a periphery with many loosely connected nodes of low activity (small average path length and small clustering coefficient, see Table 3.1).

The author-to-author **Citation graph**, denoted by C, is derived from the citation network of 32,584 papers. In Table 3.1, we can see that graph C exhibits small-world behavior because of its small average path length and its large clustering coefficient. Its degree distribution has a power-law tail with exponent $\gamma = 2.55$.

The **Facebook graph**, denoted by F, contains information about the interactions of users from September 26, 2006 to January 22, 2009 [115]. Graph F is a small-world graph like graph C (see Table 3.1).

3.3 Choosing the Time Window

Computing the properties of nodes such as centrality, clustering coefficient, and similarity, and the reputations of nodes in large graphs is very computationally intensive. Particularly in large growing graphs, updating those properties after every entry of a new edge or a new node is unrealistic. Fortunately, in most graphs neither the properties nor the reputations change very much with a small growth of the graph. Therefore, we choose an appropriate time window W for updating the properties and the reputations of nodes, so that we can reduce the cost of their update but we can still keep track of the dynamics of the reputations of nodes. Since we use personalized RWs, in principle, each node can use a different time window W according to its resources. For simplicity, however, we study the case that all nodes use the same value for W, but our method can be easily generalized for different durations of W across different nodes.

Usually in reputation systems, we are interested in the relative values of the reputations of the nodes rather than in their actual values. Therefore, as a metric for selecting a good value for W, instead of simply using reputations, we use the so-called *ranking stability* of nodes [59]. In order to compute this metric, we define the global reputation of a node as the average of its reputations computed by all the other nodes. Then, denoting the global reputation of node i at a certain time t in the evolution of the graph by $\pi(i)$, the ranking stability of node i at time t is defined as $(\pi(i) - \pi(j))/\sigma(\pi(i))$; here j is the node ranked immediately after node i in the ranking of nodes according to their decreasing reputation values at time t, and $\sigma(\pi(i))$ is the standard deviation of the set of values of node i's global reputation computed at different time instances up to and including time t. The rank of a node i is considered stable if its ranking stability is high.

In order to find the appropriate time window W, we choose a few different values of W and we keep track of the reputation and the ranking stability of each node over time as the graph grows. The reputation and the ranking stability of each node is recomputed at the end of every time window, that is, at the time points $t \cdot W$ for $t = 1, 2, \ldots$. Then, in order to observe the change of the ranking stability, we compute the *coefficient of variation* (CV) of the ranking stability of each node at these time points. A similar approach for computing the appropriate time window W has been used in [66] but that approach focused only on the actual reputation values and not on the ranking stability. The chosen W should result in a CV of the ranking stability that is neither too large nor too small, so that we are able to observe the dynamics of the ranking of reputations without needing to update them very often. For the Bartercast graph we evaluate W equal to one hour, one day, and one week, for the Citation graph one month, 6 months, and one year, and for the Facebook graph one day, one week, and one month. In the synthetic graphs, time is divided into time steps during which new edges and nodes are added (Section 4.3), and we use W equal to 10, 100, and 1000 time steps.

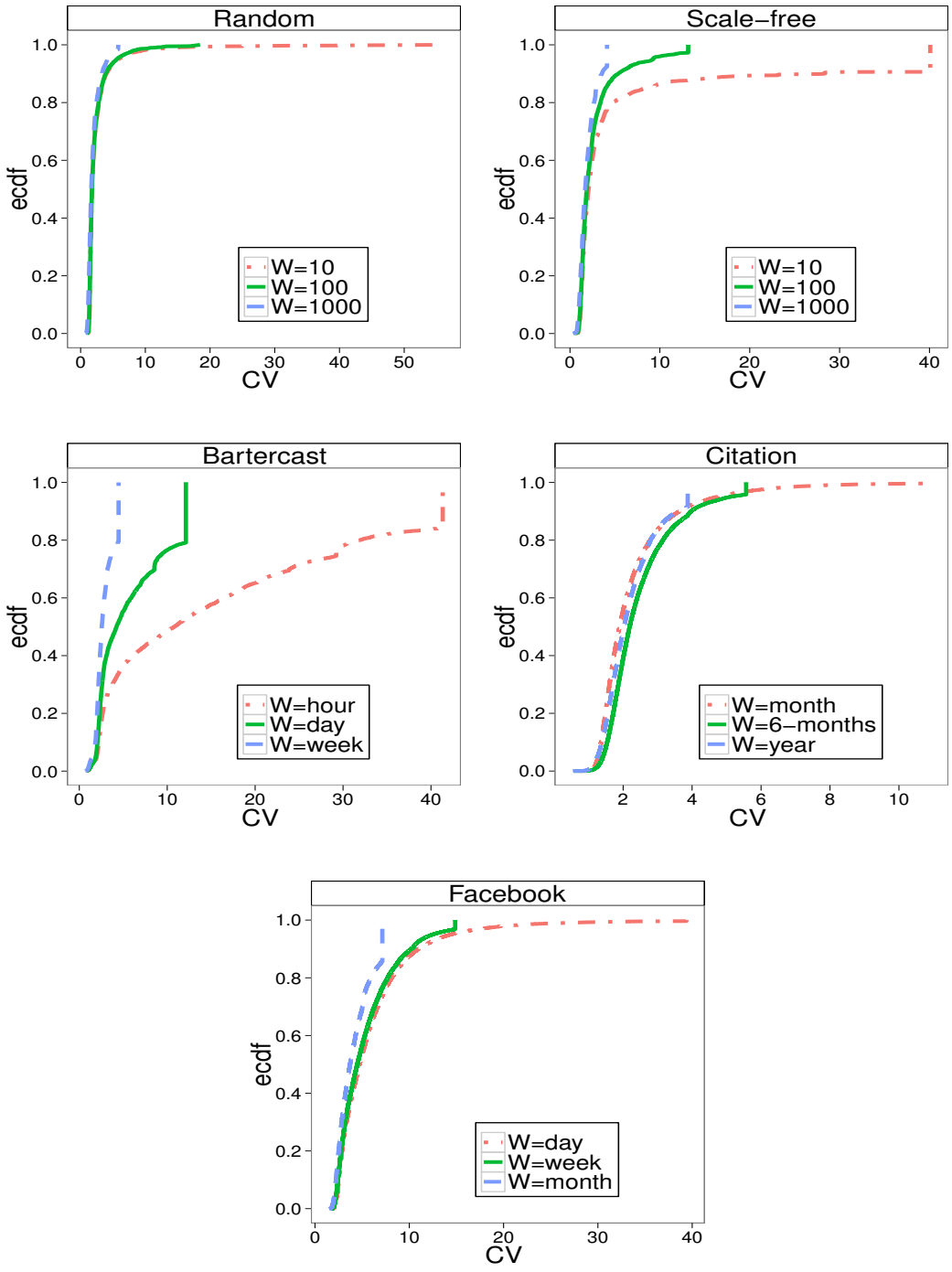


Figure 3.1: The ecdf of the coefficient of variation of the ranking stability of nodes over time for different time windows W.

In Figure 3.1, we present the empirical cdf (ecdf) of the CV of the ranking stability of the nodes for the chosen values of W. We observe that the ranking stability of the

nodes in our graphs is sensitive to W, the shorter the window W, the higher the variation of the ranking stability of nodes. Moreover, a short duration of W implies a frequent update of the reputations of nodes, fluctuating ranking stability, and noisy observations. On the other hand, a larger duration of W makes our observations smoother because of the aggregative effect of node interactions.

In real-world graphs, the variation of the ranking of nodes is smaller than in synthetic graphs, which implies that the ranking of their nodes is more stable. The Citation graph has the lowest variation of the ranking because the creation of an edge between two nodes requires more time and effort in comparison with the other graphs. In the Bartercast and Facebook graphs, the variation of the ranking is closer to those of the synthetic graphs because its nodes interact easier and so, their ranking is more dynamic. The highest variation of the rankings is observed in random graphs because node interactions follow random patterns. As a result, there are no nodes having a relatively stable behavior over time and being able to stabilize their ranking. In this chapter we choose W in such a way that the variation of the ranking is neither too small nor too large. Specifically, we choose W equal to one day for the Bartercast graph, to one year for the Citation graph, to one month for the Facebook graph, and to 100 edges for the random and scale-free graphs.

3.4 Identifying Properties of Nodes Indicative of their Behavior

In this section we define the behavior of nodes and we introduce the properties of nodes that are indicative of their future behavior. A reputation system whose calculated reputations predict the quality of future interactions reduces the effect of uncooperative nodes which do not contribute to the network resources without abusing the protocol. Such a reputation system needs to be able to predict the behavior of nodes and to rank higher the nodes with better future behavior.

3.4.1 Introducing the Properties of Nodes

We take the behavior of a node to be the difference between the resources it contributes to the network minus the network resources it consumes. We define the behavior $B(i,t)$ of a node i at time $t \cdot W$ as $B(i,t) = \sum_{j \in N_i}(s_{ji} - s_{ij})$, where the s_{ij} and s_{ji} are the strengths of the incoming and outgoing edges of node i at time $t \cdot W$. In the Bartercast graph, the strength of a link is the amount of data transferred across the link, and the behavior of a node corresponds to its cooperation level. In the Citation graph, the strength of an edge is the number of citations and in the Facebook graph, the strength of an edge is the number of interactions between two friends.

The properties of a node that may be predictive of its behavior can be divided into three categories based on the information needed for their computation: local, global, and temporal. The *local properties* can be naturally integrated in a RW since their computation does not need access to global information and they are computationally simple. The local properties of a node i that we use are:

- Its degree, which represents its activity.
- Its ego-betweenness centrality (ego-BC), which is its betweenness centrality in its ego-network, namely the network containing that node, its neighbours, and all the links among them [48].
- Its clustering coefficient, which is defined as the fraction of links among its neighbors that actually exist.

The computation of *global properties* demands high cost and global information. However, it is interesting to observe their predictive ability on the behavior of a node. The global properties of a node i that we use are:

- Its eigenvector centrality, whose basic idea is that interactions with highly reputed nodes contribute more to the reputation of a node.
- Its betweenness centrality (BC), which is defined as the sum of the fractions of shortest paths among all pairs in the graph that pass through this node and it indicates the amount of flow passing through that node.
- Its closeness centrality, which is the inverse of the sum of its distances from every other node in the network.

Finally, we use the *temporal properties* of node i to predict its behavior. Temporal properties require only local computation and can be easily integrated into RW. The temporal properties of a node i that we use are:

- Its average interaction time, which is the average time interval between successive interactions of node i.
- The time occurrence of the last interaction of a node i.
- Its age, which is expressed as $\tau(i) = t_c - t(i)$ where t_c is the current time and $t(i)$ is the time instance node i joined the system.

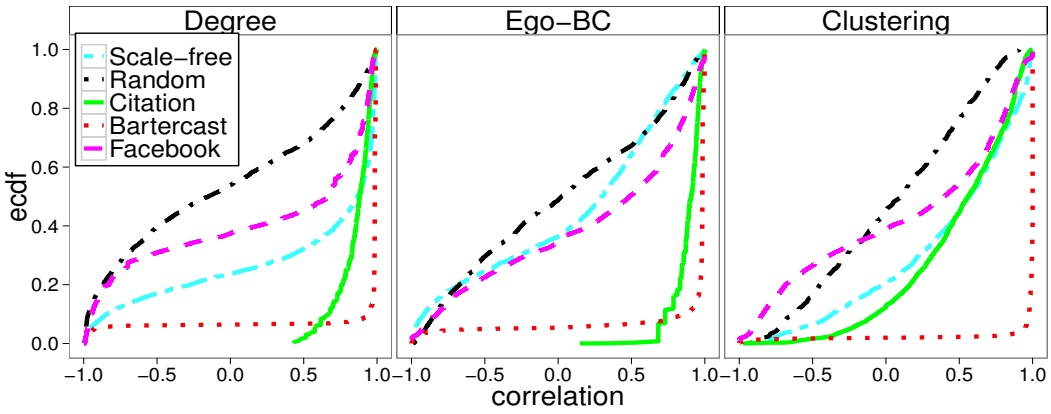
3.4.2 Evaluation

In order to assess to what extent a property of nodes is predictive of their future behavior, we compute the correlation between the node properties and the behavior of nodes over time as the graph grows. More precisely, for each node i and for each property, we compute the correlation between the sequence of values of that property of node i at time $t = 1, 2, \ldots$ and the sequence of values of its behavior at the next time step $B(i, t+1)$, for all t available from our datasets. For all the correlations, we use the Spearman correlation, which assesses the monotonic relationship between two sequences.

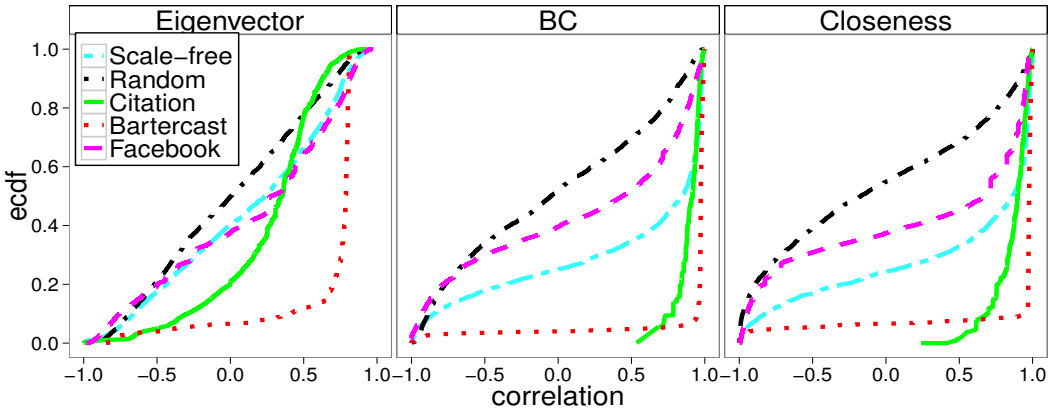
In Figure 3.2, we present the ecdf of these correlations for all the nodes in each graph. In our real-world graphs, the properties of a node are strongly correlated with its future behavior, particularly in Bartercast where the correlation is almost perfect. In these graphs, there are a few nodes attracting the majority of links. In Bartercast, these nodes are the nodes with high upload speeds that share many files, while in the Citation network, they are the authors of papers with high impact. As their degree, clustering coefficient, and centrality increase, these highly connected nodes improve their behavior as well. Nevertheless, temporal properties are also indicative of their future behavior because as has been observed in many real-world networks, the nodes gradually reduce their activity with time until they become inactive [6]. Facebook exhibits correlations similar to those in scale-free graphs.

In scale-free graphs, the future behavior of nodes is correlated mostly with their degree, BC, and age, due to the way they are constructed. In a scale-free graph, a new node connects with higher probability to nodes with high degrees, and so, a few older nodes obtain higher degrees and exhibit better behavior while the majority of nodes have much smaller degrees. In such graphs, the nodes with higher degree participate in the majority of the paths between the other nodes, and as a result they have high BC and high closeness centrality. Besides age, the other temporal properties are not correlated with the future behavior of nodes. In random graphs, all nodes have uniform connectivity and the interactions between the nodes are random. Therefore, there is almost no correlation between the properties of nodes and their future behavior.

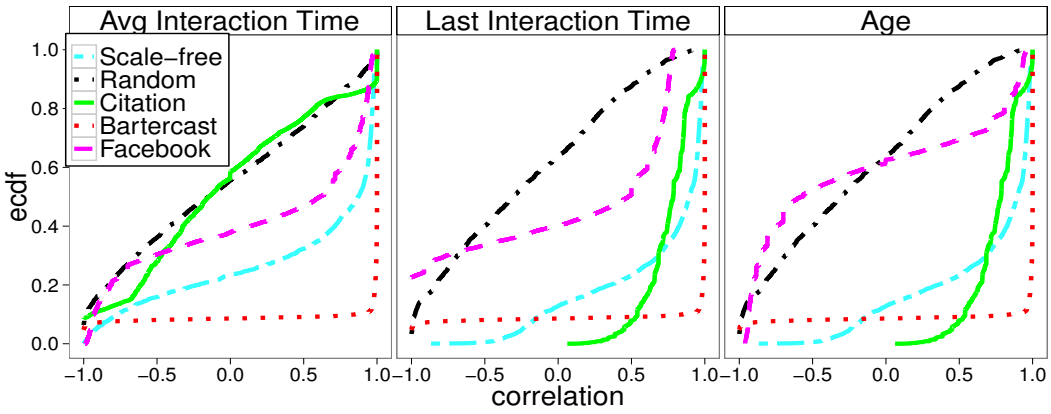
In Table 3.2, we present for all our graphs the properties of nodes having the highest correlations with their future behavior. For most graphs, the degree of nodes, even though it is the simplest local property, exhibits the highest correlation in comparison with the other local properties. Only for Bartercast, the clustering coefficient of nodes is more correlated with their future behavior because of its high churn. In Bartercast, a high clustering coefficient indicates that a node participates in the core of the network where its neighbors are active and interact with each other. A node having a low clustering coefficient is located in the periphery of the network. Nevertheless, in Bartercast also the degree of nodes predicts very well their future behavior.



(a) local properties



(b) global properties



(c) temporal properties

Figure 3.2: The ecdf of the correlations between the properties of each node and its future behavior over time.

Table 3.2: The local, global and temporal properties of nodes exhibiting the highest correlation with their future behavior.

Graph	Local	Global	Temporal
Random	none	none	none
Scale-free	degree	BC, closeness	age
Citation	degree	BC	last interaction time
Bartercast	clustering	BC, closeness	all
Facebook	degree	closeness	avg interaction time

The global properties of nodes that are based on shortest paths, namely BC and closeness, exhibit much higher correlations than eigenvector centrality which is based on random walks. In the Citation graph, the flow of information passing through an author influences his future connections. As a result, an author with high BC has a higher probability to contribute more in the network. In Facebook, a node within a short distance from other nodes has better access to their wall-post and vice versa. Therefore, this node having higher closeness centrality, has a higher probability to have a good behavior. In scale-free graphs and Bartercast, BC and closeness perform equally well. In these graphs, the nodes having high BC also have high closeness centrality because the clustering of these graphs is low. As a result, the nodes having many shortest paths passing through them are closer to the other nodes in the network. As our experiments show eigenvector centrality does not predict well future behavior of nodes.

The temporal property of nodes having the highest correlations depends on the construction process of each graph. In scale-free, the age of nodes predicts better their behavior since older nodes attract the majority of links. In Citation graph, the time of last interaction is more predictive because it indicates that an author is still active. In Facebook, the average time between two interactions performs better since it reveals the tendency of a node to participate in conversations. For Bartercast, all the temporal properties perform almost equally well. In a network with high churn like Bartercast, the nodes that stay for a long time in the system tend to interact more often with other nodes and contribute to the system. Thus, all the temporal properties of these nodes are equally good predictors of their behavior.

3.5 Biasing Random Walks in the Face of Uncooperative Nodes

After having observed the correlations between the properties of nodes and their future behavior, we bias two types of RW with those properties: the naive RW (nRW) and the

supervised RW (sRW). We assess to what extent the reputations computed by both types of bRWs predict the behavior of nodes and rank lower the nodes with bad future behavior.

3.5.1 Naive Random Walks

Naive RWs are implemented in a similar way as the simple RW but now the edge weights, and so the transition probabilities, depend on the node properties presented in Table 3.2. The only additional cost of nRWs is the computation of these properties. We consider four types of nRWs: local nRW, global nRW, temporal nRW,and mixed nRW, in which we bias each walk with the corresponding local, global, temporal property of nodes, and with the combination of all these properties, respectively. In each case, the transition probabilities are proportional to the property of the targeted node. If according to Table 3.2, more than one node properties correspond to a random walk, we chose the property with the lowest computational cost.

We evaluate whether the reputations of nodes predict their future behaviors, considering that in most reputation systems we are interested in the ranking of nodes according to their reputations. At each time $t \cdot W$, we compute the correlation between the sequence of the reputations of all the nodes in the graph and the sequence of their behaviors at the next time step $(t+1) \cdot W$, and we observe this correlation over consecutive time windows. We note that correlating the reputations at time t with the corresponding behaviors at time $t + 2$ is equivalent to choosing a W of double size. In Figure 3.3, we present the result of our evaluation for the random walks with teleportation probability $\alpha = 0.15$, a commonly used value for teleportation [92]. The presented result is the average of all the nodes. We found that the value of α does not affect much the correlation and so, we present only the values for $\alpha = 0.15$.

In all graphs in Figure 3.3, the reputations computed by the nRWs achieve much higher correlations with the future behaviors of nodes than the simple RW. Therefore, all nRWs are able to predict the nodes with the best future behavior. Nevertheless, the performance of the nRWs depends on the topology of the graph. In graphs such as scale-free, Citation and Facebook, where the creation of links follows specific patterns almost stable over time, all RWs exhibit higher correlations than in random graphs and Bartercast. Furthermore, temporal and global RWs exhibit similar correlations implying that the global and temporal properties of a node are highly dependent. For instance, in most cases, an old node with small average interaction time has high centrality.

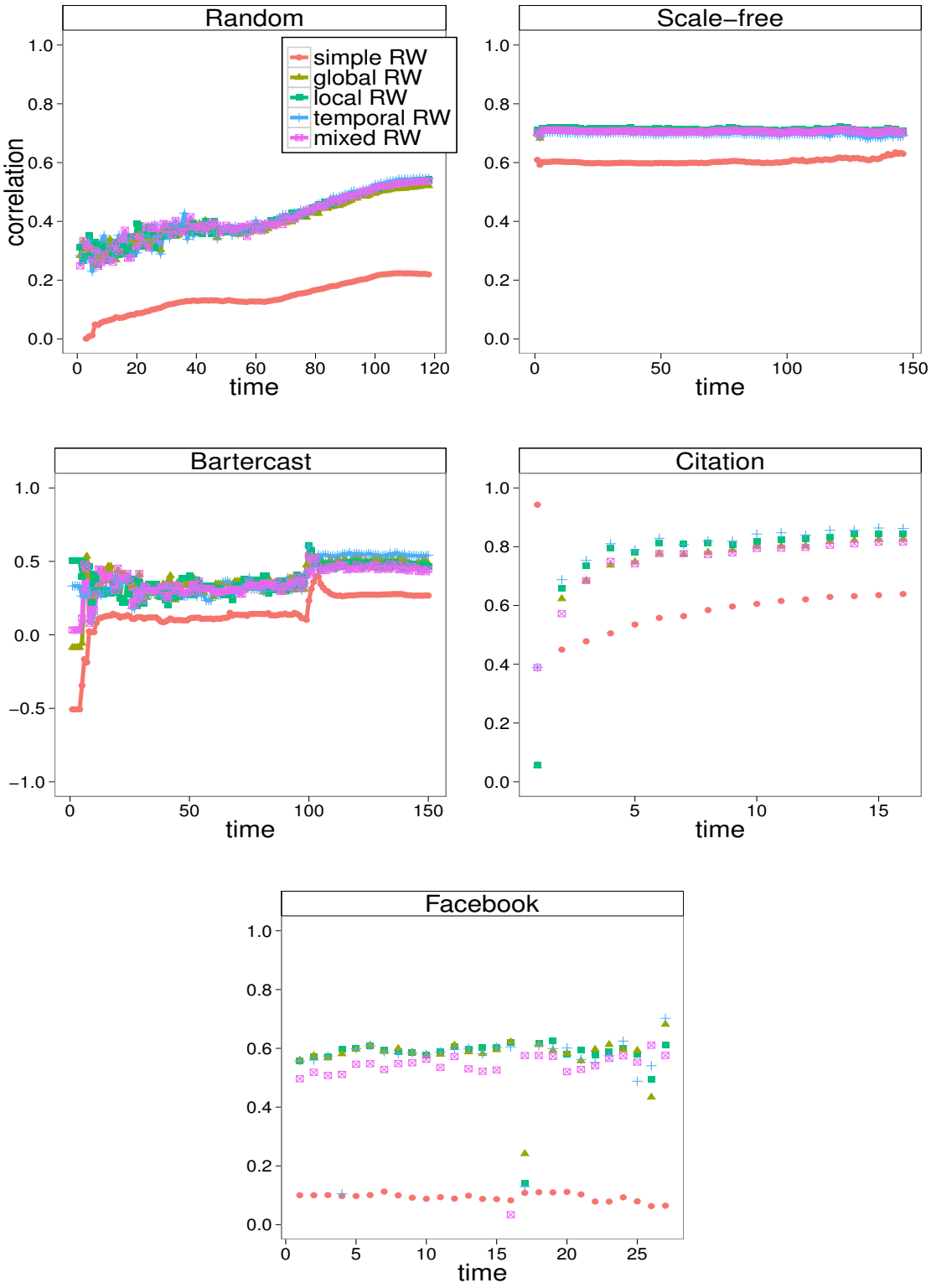


Figure 3.3: The correlation of the reputations of nodes as computed with naive Random Walks and their future behavior for consecutive time windows (note: the scale of the vertical axis of the Bartercast plot is different).

Table 3.3: The size of the intersection of the top-5 most highly reputed nodes at time $t \cdot W$ and the top-5 best behaved nodes at time $(t+1) \cdot W$ averaged over all t.

	simple RW	local RW	global RW	temporal RW	mixed RW
Random	0.64	0.95	0.91	0.42	0.94
Scale-free	3.64	4.87	4.75	2.37	4.79
Bartercast	2.15	3.22	2.94	2.67	3.16
Citation	3.36	4.64	4.23	3.77	4.58
Facebook	2.90	3.40	2.90	3.00	3.20

In many applications of reputation systems, such as recommendation of friends in Facebook or recommendation of papers in Citation graphs, we are interested only in the top ranked nodes. In Table 3.3, we show the size of the intersection of the set of the top-5 most highly reputed nodes at time $t \cdot W$ and the set of the top-5 nodes with the best behavior at time $(t+1) \cdot W$, averaged for all t. In all graphs, the nRWs rank the top-5 nodes with the best future behavior higher than simple RW does, with local nRW achieving the highest number of common nodes and temporal nRW the smallest. In all graphs other than the random graph, the number of common nodes is high. In random graphs, the nodes follow a random pattern of interactions and so, we cannot predict accurately even the top-5 nodes with the best future behavior.

3.5.2 Supervised Random Walks

Although naive RWs are able to predict the best behaved nodes with high accuracy, the weights they use combine node properties into transition probabilities in a rather arbitrary way. For further evaluation of the ability of random walks to predict the best behaved nodes, we use supervised RWs (sRWs) where the weights assigned to each edge are learned and optimized during the previous time window. We compare sRW with simple RW and naive mixed RW. For each edge e_{ij}, we assume that the vector ψ_{ij} (see Section 3.1.2) keeps all the properties of node j presented in Section 3.4.

The computation of the optimal weights for sRW starting from a node i includes the computation of the vector u_i, which is the solution of the multimodal optimization problem presented in Section 3.1.2. Due to its multimodality, this optimization problem is very computationally expensive and so, we need to further reduce the cost of computation. We observe that the vast majority of nodes in our graphs interact with other nodes that are only a few hops away. In Figure 3.4, we present the probability of interaction between two nodes in our graph as a function of their distance just before they interact. As we see, our graphs exhibit a high locality of interaction, which implies that we can reduce the cost of the computation of reputations by pruning the graph which is traversed by the random walks started at an initiator node without losing much on the performance.

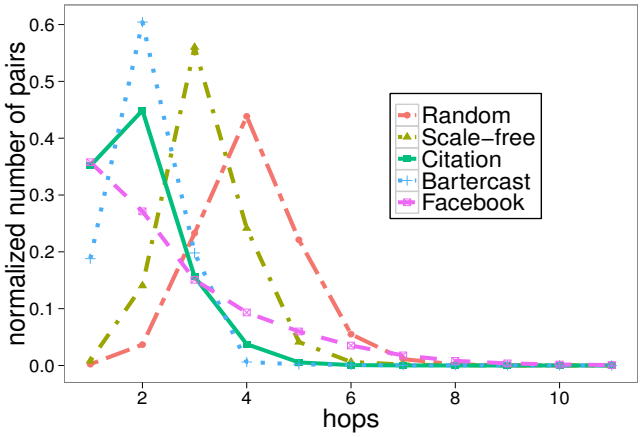


Figure 3.4: The distance in number of hops between a pair of nodes before they interact.

Table 3.4: The size of the intersection of the top-5 most highly reputed nodes at time $t \cdot W$ and the top-5 best behaved nodes at time $(t+1) \cdot W$ averaged over all t.

	simple RW	naive RW	supervised RW
Random	0.09	0.24	0.26
Scale-free	2.5	3.10	3.14
Bartercast	1.75	3.00	3.00
Citation	3.18	4.21	4.22
Facebook	1.24	1.53	2.14

We observe that in all graphs but random graphs, more than 90% of pairs of interacting nodes have a distance of at most 3 hops just before they interact. Therefore, here, we use random walks with length of 3 hops.

In Figure 3.5, we present the correlation between the reputations of nodes as computed by RW, nRW, and sRW, and their future behavior. For sRW, we start the random walks from the two most well connected nodes in each graph, due to the high computational complexity of sRW. Our sRW outperforms nRW and RW in all graphs. However, the computational cost of sRW is much higher than nRW.

In Table 3.4, we present the size of the intersection of the set of the top-5 most highly reputed nodes and the set of the top-5 best behaved nodes in the next time window averaged over all consecutive time windows. In most graphs, the sRW identifies the best behaved nodes only slightly better than nRW does. Therefore, if we are only interested in the top ranked nodes, nRW constitutes a good compromise between accuracy and computational cost.

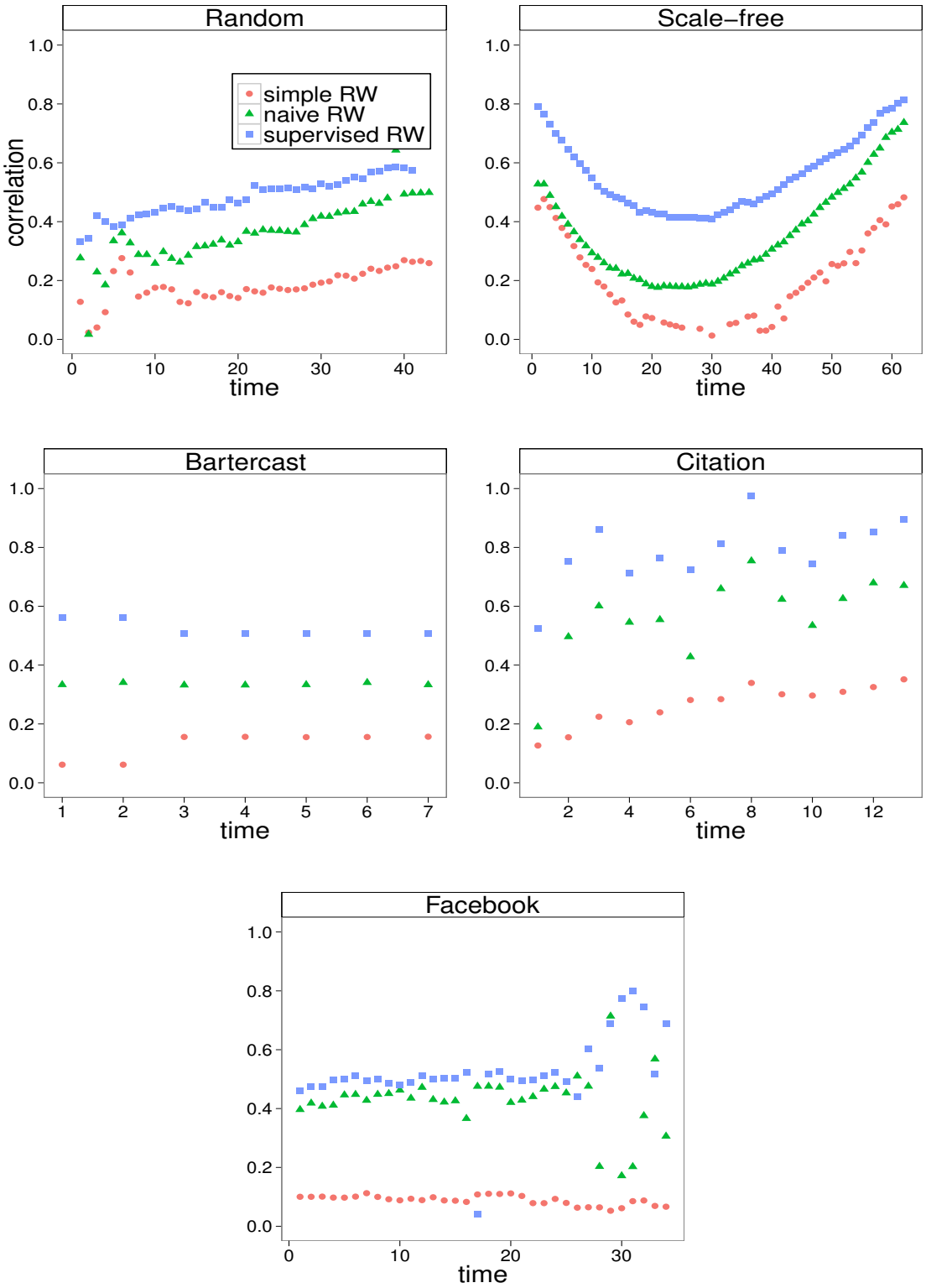


Figure 3.5: The correlation between the reputations of nodes as computed by simple, naive and supervised Random Walks and their future behavior for consecutive time windows.

3.6 Biasing Random Walks in the Face of Sybil Attacks

In this section, we bias RWs with node properties in order to increase their resilience against sybil attacks. Our aim is to make RWs stay away from malicious nodes and sybils so that the reputations assigned to such nodes are low. We bias only nRWs because for sRWs we cannot have a meaningful training set [108], since we have not observed any sybil attack in our datasets. Nevertheless, our experimental evaluation shows that even nRWs drastically reduce the effect of sybil attacks.

Most of the schemes proposed against sybils attacks in the literature [125], [126] are based on the observation that the sybil nodes can create only a limited number of edges to honest nodes because interacting with honest nodes requires a high social engineering cost [117]. As a result, the honest nodes form a region that is well separated from the sybil region containing the sybil nodes. The sybil nodes connect with each other and with the malicious nodes in an arbitrary way. The two regions are connected by the *attack edges* that link nodes in the sybil region to *victim* nodes in the honest region. The probability that an RW escapes to the sybil region depends on the number of attack edges and the visit ratios of the RW to the victims, but not on the topological characteristics of the sybil region [125]. In our experiments, we take as the honest region our initial graph $G = (V, E)$. Since the topology of the sybil graph is not important, we create a sybil graph $G_s = (V_s, E_s)$ using the BA model [14]. Then, we chose some sybil nodes from G_s and some prespecified victim nodes from G, and connect them through the corresponding attack edges E_a. The resulting graph is $G' = (V', E')$ where $V' = \{V \cup V_s\}$ and $E' = E \cup E_s \cup E_a$. To chose the victim nodes in G we use two approaches. Either the victims are chosen uniformly at random, or the malicious nodes try to increase their impact and attack highly reputed nodes by choosing the victims with probabilities proportional to their reputations. The latter selection of victims is also known as centrality attack [32].

The properties of nodes used to bias RWs must not depend on the topological properties of the sybil region. Therefore, we do not use global properties of nodes, but we use the following *local* and the *temporal properties*:

- The similarity of two nodes i and j with neighborhoods N_i and N_j, respectively, defined by the Jaccard similarity ($|N_i \cap N_j|/|N_i \cup N_j|$), which assumes that two nodes are similar if they have many common neighbors.

- The weight of an edge e_{ij} connecting two nodes i and j, indicating the strength of the corresponding interaction, as mentioned in Section 3.1.2.

- The inverse log-weighted similarity between two nodes i and j, defined as the number of their common neighbors weighted by the inverse logarithm of their degrees ($\sum_{k \in |N_i \cap N_j|} (1/\log[d(k)])$, where $d(k)$ is the degree of node k). It assumes that two nodes are similar if they have low-degree common neighbors [3].

- The time t_{ij} that an edge e_{ij} is created.

The nodes in the sybil region can claim any values for these properties without affecting the probability of a RW escaping from an honest node to the sybil region. Since in order to escape to the sybil region, the RW has to traverse an attack edge, the properties of the nodes adjacent to the attack edges determine the probability that an RW escapes into the sybil region. We assume that it is more costly for an attacker to create an attack edge with a large weight than an attack edge of a low weight and so, attacks edges of low weights are more common. Therefore in our experiment, we assign probabilistically a weight to each attack edge so that, attack edges with small weights are more common. For the time the attack edges have been created, we assume that they are uniformly distributed over time. We bias the nRW with each of the properties defined above, and we correspondingly have four types of nRWs: Jaccard nRW, weighted nRW, inverse nRW, and temporal nRW.

In Figure 3.6, we show the escape probability of the different RWs versus the ratio of the number of attack edges and the number of honest nodes when victims are chosen uniformly at random. Due to the large size of most of our graphs, the results are the average escape probability with 500 nodes performing the corresponding RW with teleportation parameter $\alpha = 0.15$. The effect of parameter α on the escape probability is not assessed in this chapter. However, there is a first study on this effect in [89].

The real-world graphs where the honest nodes form a well connected region, have the smallest escape probability for all types of RWs, while the synthetic graphs have the largest. The type of nRW giving the smallest escape probability depends on the topology and the characteristics of the graph. In the random, scale-free, and Citation graphs, the weights take values in a small range and so, the weighted nRWs perform similarly to simple RW. In these graphs, inverse nRW results in the smallest escape probability, especially in the Citation graph, which has a large clustering coefficient indicating that nodes share many neighbors. On the other hand, in Bartercast, where the weights can vary from a few KB to several MB, the weight of an edge indicates accurately the trust between the interacting nodes and so, weighted nRW results in an escape probability that is almost zero, even though the number of attack edges is relatively high. On the contrary, due to its small clustering coefficient which is smaller than the corresponding random graph, the Jaccard and inverse nRWs result in escape probabilities similar to that of simple RW. Nevertheless, the Jaccard and inverse nRWs result in small escape probabilities for all the graphs but Bartercast. The temporal nRW performs better in Facebook, resulting in an escape probability that is almost zero because two nodes with many fresh interactions between them trust each other.

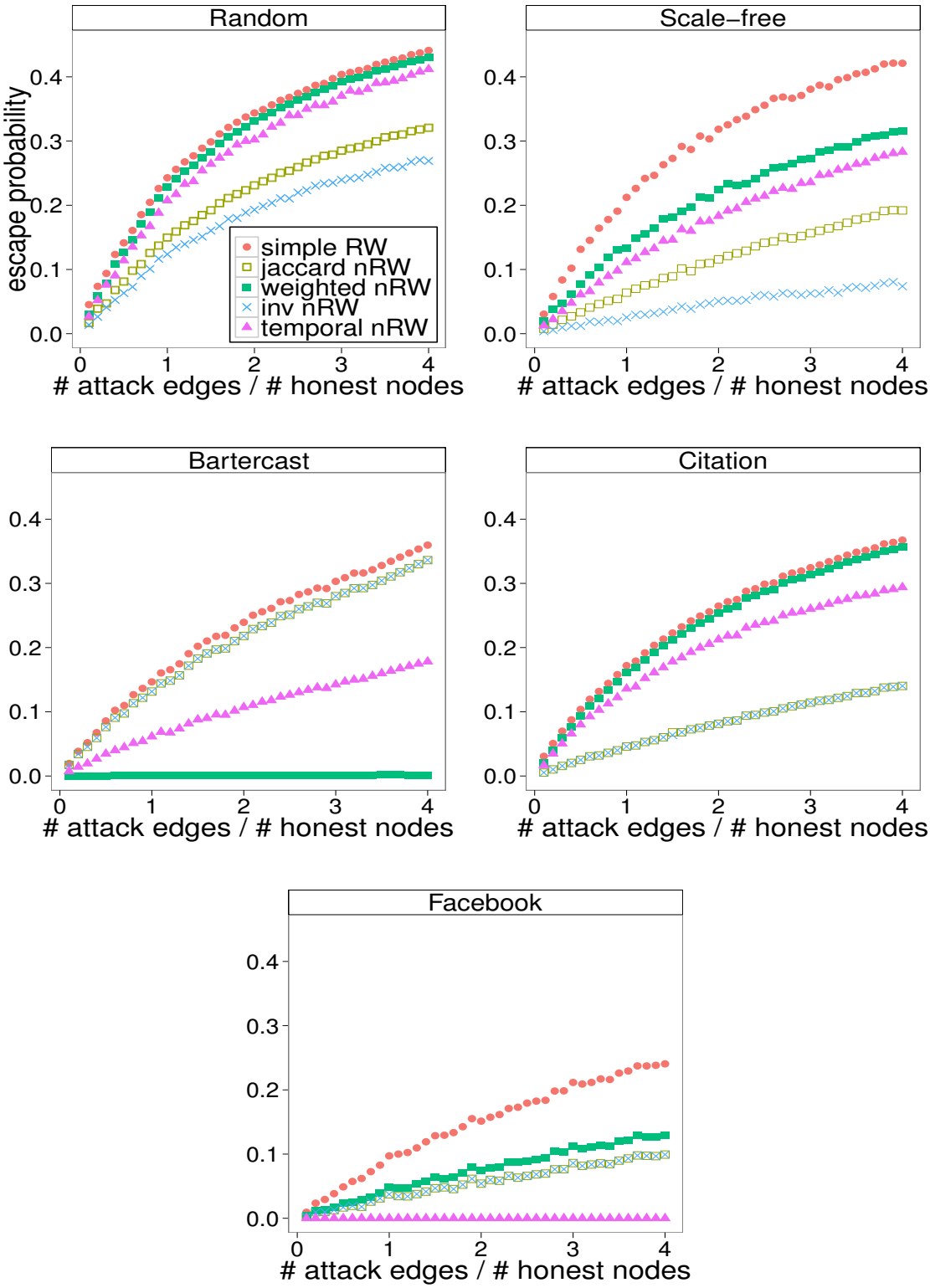


Figure 3.6: The escape probability to the sybil region of the simple and the biased Random Walks versus the ratio of the number of attack edges and the number of honest nodes when the victims are chosen uniformly at random.

In Figure 3.7, we show the escape probabilities of different RWs when the victims are chosen with probabilities proportional to their reputations. Counter-intuitively, when malicious nodes use this targeted attack instead of randomly choosing victims, the escape probability is smaller for all types of RWs. The most highly reputed nodes attract the majority of the attack edges while they also have many edges from honest nodes connected to them. As a result, an RW visiting them has a lower probability to traverse an attack edge even though highly reputed nodes are visited with a higher probability by RWs. Nevertheless, in a random graph the difference between the impact of the two types of sybil attack on the escape probability is very small due to the homogeneity of its nodes. Furthermore, in all graphs, inverse nRW gives a smaller escape probability than Jaccard nRW because low-degree nodes usually have low reputations and are not targeted by the malicious nodes.

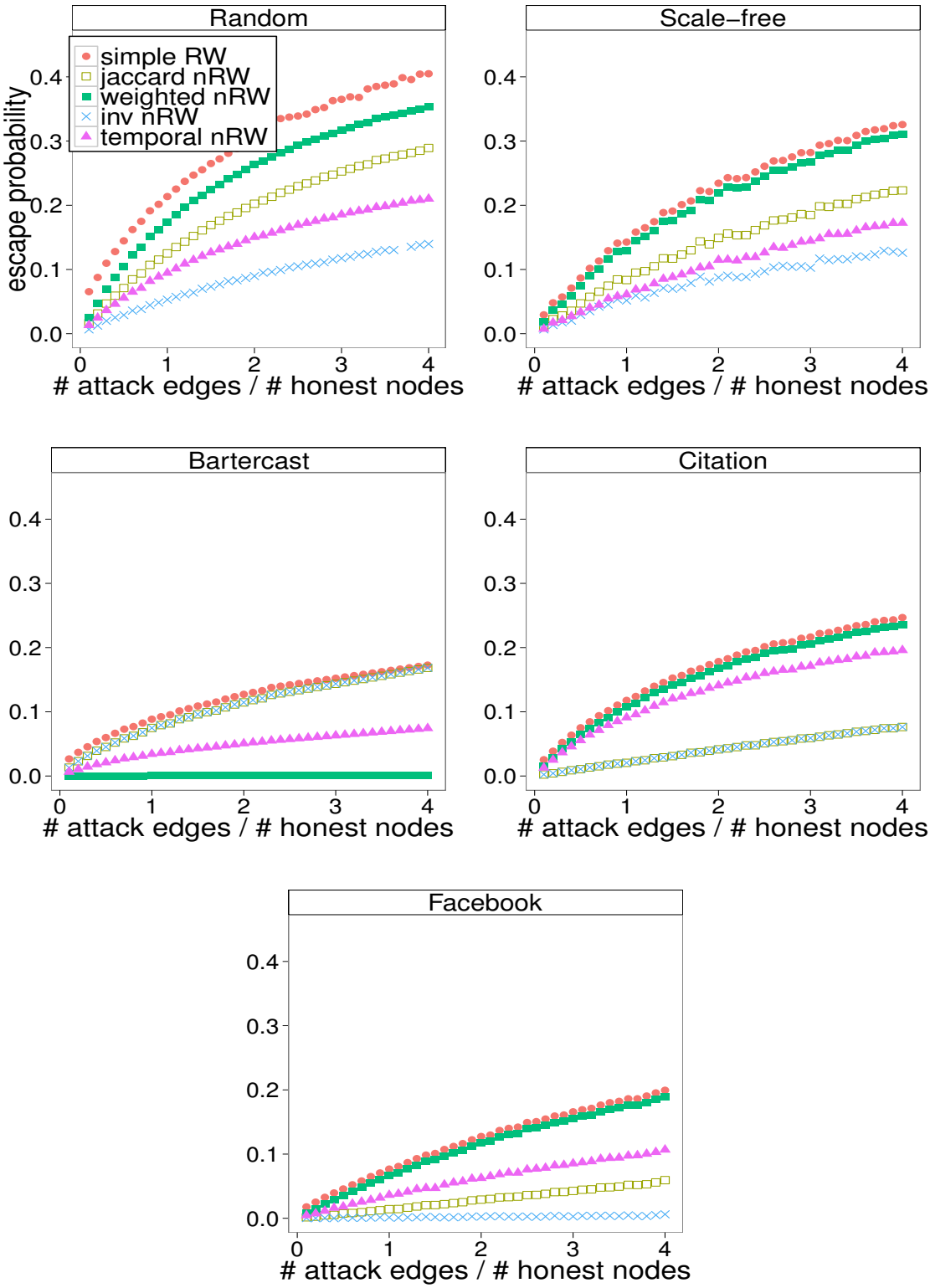


Figure 3.7: The escape probability to the sybil region of the simple and the biased Random Walks versus the ratio of the number of attack edges and the number of honest nodes when the victims are chosen with probabilities proportional to their reputations.

3.7 Conclusion

Our evaluations indicate that using node properties improves a lot the resilience of RWs against uncooperative nodes and Sybil attacks. Concluding, our results imply the following.

First, the time window used for observing a graph depends on the characteristics of the graph. In graphs such as Citation, where the creation of an edge requires large effort and time, the time window can be large, for example a year, and still follow accurately the dynamics of nodes. On the contrary, in graphs such as Bartercast graph, where the creation of edges is easier, we need a smaller time window, for example a day.

Secondly, the prediction of the behavior of nodes depends on the characteristics of the graph. Predicting the behavior of nodes is very accurate in graphs with both specific construction patterns and nodes with heterogenous properties, such as the scale-free, Citation and Facebook graphs. In graphs of nodes with uniform properties and highly dynamic behavior, biased random walks predict less accurately the behavior of nodes but still much better than simple RW.

Furthermore, the appropriate node properties to bias random walks against Sybils depend on the characteristics of the graph. In graphs with large clustering coefficient, such as our scale-free, Facebook and Citation, RW biased with node similarities, especially inverse log-weighted similarity, are very effective. In graphs with edges with heterogenous strengths, such as Bartercast and Facebook, biasing RW with the strength of an edge is very effective while using temporal properties is effective in graphs with strong temporal patterns.

In most of our graphs, random walks biased with very simple node properties with low computational cost such as, the degree, the weights, and the age, perform very well against uncooperative nodes or sybil strategies. Biasing random walks does not necessarily add a lot of extra computational cost and as a result, biased random walks can be easily used in decentralized systems where nodes have limited resources.

In this chapter, we have shown that node properties enhance a lot the robustness of RW against exploitative nodes. Nevertheless in a distributed environment, nodes do not necessarily have access to the properties of other nodes, nor the information to compute them. Nodes can exchange their properties using a gossip-like protocol, but this is not reliable due to potential misreporting by some nodes. A reliable alternative is the use of a system like Bartercast [43], where the nodes store locally their own perception of the graph and then they can compute the properties of the nodes in their locally stored graph. Moreover, directing most of the RWs through nodes with particular properties results in overloading those nodes. This overload might cause even the failure of some highly reputed nodes and thus, it must be studied before adopting biased RW.

Chapter 4

Forgetting the Least Important parts of the History of Interactions

Networks such as popular online markets and social networks consist of hundreds of thousands or even millions of active users and thus, using the complete history of interactions for computing the reputations of nodes is prohibitive due to its resource requirements. Particularly in decentralized systems, such as file-sharing P2P systems, the available resources at nodes are limited and thus, only scalable solutions can be applied. Furthermore, a long-term history allows previously well-behaved nodes to exploit their good reputations by acting maliciously [53, 84, 121].

In this chapter, we propose a scheme for reducing the amount of history maintained in decentralized interaction-based reputation systems. We experimentally explore its effect on the computed reputations using synthetic and real-world graphs. In order to reduce the history of interactions, we use only a subset of the complete history to approximate reputations. We model the interactions of the *complete history* of a network as a growing graph with the nodes of the network as its vertices and the interactions between pairs of nodes as its edges, and the corresponding *reduced history* as a subgraph of the complete history. The reduced history is derived from the complete history by deleting the least important edges and nodes. We define the importance of a node according to its age, its activity level, its reputation, and its position in the graph, while the importance of an edge is defined according to its age, its weight, and its position in the graph. Then we evaluate our approach using synthetic random and scale-free graphs, and two real-world graphs, one derived from the Bartercast reputation system and the other from the author-to-author Citation network. The main difference between the Bartercast and the Citation graphs, besides their structural properties, is that the former is derived from a deployed distributed system with personalized reputations while the latter is derived from a centralized system with global reputations. On these networks, we apply both max flow-based and random walk-based computation of reputations.

The contributions of this chapter are as follows:

1. We propose an approach for reducing the history of interactions according the following to two observations: (i) for the vast majority of reputation systems, the ranking of reputations is more important than the actual reputation values themselves; and (ii) in most cases the identification of the highest ranked nodes is enough.

2. We demonstrate that the performance of our approach for reducing the history of depends on the topology of the complete history.

3. Furthermore, we show that the performance of our approach for reducing the history of interactions depends on the reputation algorithm and we conclude that it can be applied in a large range of networks.

4.1 Problem Statement

Our main motivation for reducing the history of interactions in a network is the computational cost and the storage requirements of decentralized reputation algorithms. Reputation systems, such as those of eBay or Google, cover hundreds of thousands of active nodes while reputation algorithms (e.g., Eigentrust [78], PageRank [92] and max-flow based ones [35]) have a high computational complexity. In decentralized systems, like BarterCast, where each node stores and analyzes data locally using, e.g., the max-flow algorithm (with complexity $O(nm^2)$ where n is the number of nodes and m and the number of edges), even much smaller graphs of 10^6 nodes make the computation of reputations prohibitive. Taking into account that the contributions of nodes in the computation of reputations are not equal in quality and quantity [40], thus we aim to delete the least important contributions and compute reputations using only a subset of the complete history. In this way, we can reduce the computational cost significantly without decreasing the accuracy very much.

In addition to the computational cost, the dynamic behavior of many reputation systems makes the use of the complete history ineffective. In systems with a high population turnover such as P2P networks, only a few nodes remain for a long period in the system while the majority of nodes enters the system performing some interactions and then leaves it. This behavior has also been observed in Bartercast (see Table 4.1). Also a node behaving properly for a long time can build a good reputation and become a traitor [84] by exploiting other nodes. Preserving only short-term history forces all nodes in the system to behave consistently according to the protocol. For these reasons, several widely used reputation mechanisms, such as those of eBay and eLance, allow the use of historical information of a 1 or 6-month window. Although using a time window is useful for

such feedback-based reputation systems, it is not effective in interaction-based reputation systems since important information of highly reputed nodes is deleted.

We model the interactions of a network using the interaction graph $G = (V, E)$, where the vertices V represent the nodes and the edges E the interactions among the nodes. The weight of an edge represents its strength; for instance, in Bartercast, the weight of an edge between nodes represents the amount of data transferred in the corresponding direction, and in a citation graph, it represents the number of references to an author by another. The graph is dynamically growing over time and allows not only new nodes to join but also existing nodes to create new edges. The graph G represents the *complete history* (CH) of interactions in the network. Given the growing graph G, our target is to create a subgraph of G, denoted by G', which preserves the highest ranked nodes in G and keeps the ranking of the reputations similar to that in G. The subgraph G' has to be dynamically maintained as the complete history grows while its size has to be almost fixed. The graph G' will be used for the computation of reputations, and represents the *reduced history* (RH) of interactions in the network.

4.2 Creating the Reduced History

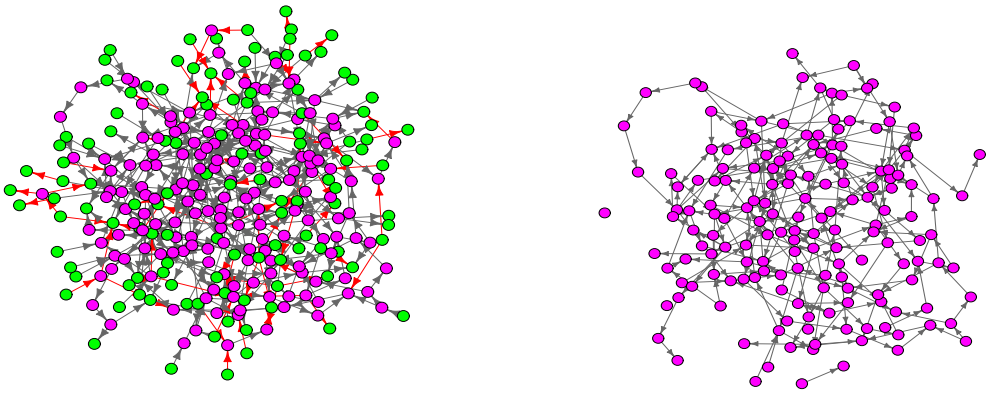
The basic idea of creating the reduced history G' consists of removing the least important elements, either nodes or edges, from G. We use a node removal process in conjunction with edge removal. The ratio of removed nodes versus removed edges depends on the dynamics of the network. Nevertheless, edge removal implies node removal and vice versa. More precisely, edge removal can lead to disconnecting a node from the graph and node removal results in deleting the adjacent edges of the removed node. In Figure 4.1, we illustrate the complete history of interactions and the corresponding reduced history.

4.2.1 The Parameters for Node Removal

The parameters for removal of a node consist of its age, its activity level, its reputation, and its position in the graph.

The *age of node* i is expressed as $\tau_i = t - t_i$ where t is the current time and t_i is the time instance node i joined the system. In most networks, the age of a node i affects its behavior in a non-linear way (e.g. [6, 70]). Thus, instead of its age, we consider its aging factor $f(\tau_i)$, where f is a decreasing function with $f(0) = 1$ (e.g., $f(\tau) = e^{-b\tau}$, where τ represents the age of a node and b is a constant). Keeping fresh information allows the reputations system to capture the dynamic behavior of nodes.

The *activity level* d_i *of a node* i represents its degree. Nodes with a high activity level participated in many interactions, and so, they provide much information.



(a) The complete history of interactions (b) The reduced history of interactions

Figure 4.1: The least important nodes and edges in the complete history are denoted by green and red respectively.

The *reputation of node* i is denoted by r_i. Our aim is to preserve the information of nodes with high reputations, since these nodes are the most reliable in the network. Moreover, allowing nodes with high reputations to contribute to the computation of reputations longer is a kind of rewarding the most trusted nodes.

For node i the *importance of its position in the graph* is expressed by its betweenness centrality (BC), denoted by $C_B(i)$, which measures the sum of the fractions of the numbers of shortest paths among all pairs of vertices that pass through node i [55]. Removing nodes from the graph can result in destroying its structure by creating many disconnected components and thus, we need to maintain the nodes that keep the graph connected.

The first three factors represent the behavior of node i while the fourth factor is added for preserving the structure of the graph during the deletion process. Therefore, in our method, the *priority score* $P_n(i)$ of deleting node i is defined as

$$P_n(i) = \alpha P_A(d_i, r_i, \tau_i) + (1 - \alpha) P_B(C_B(i)), \tag{4.1}$$

where $P_A(d_i, r_i, \tau_i)$ expresses the priority score of deleting node i based on its activity level, aging factor and reputation, and $P_B(C_B(i))$ represents the priority score of deleting node i according to its position in the graph. The parameter α takes values in $[0, 1]$ and can be chosen according to the graph properties. We define the priority score P_A as

$$P_A(d_i, r_i, \tau_i) = \frac{n - d_i r_i f(\tau_i)}{n^2 - \sum_j d_j r_j f(\tau_j)}, \tag{4.2}$$

where n equals the number of nodes in the graph, and the denominator acts as a normalization so that the sum of the priority scores sum to 1. Clearly, a node with a higher age, a lower activity level, or a lower reputation will be removed. Although the maximum value of $d_i r_i f(\tau_i)$ is equal to $n-1$ (corresponding to $d_i = n-1, r_i = 1$ and $f(\tau_i) = 1$), for simplicity, we approximate it to n. Similarly, P_B is expressed as

$$P_B(C_B(i)) = (n^2 - C_B(i))/(n^3 - \sum_j C_B(j)).$$

Again, even though the maximum value of $C_B(i)$ is equal to $(n-1)(n-2)$, we approximate it by n^2. When considering a single parameter for node removal, Eq.4.2 can be adapted in a straightforward way (similarly as P_B for parameter $C_B(i)$).

4.2.2 The Parameters for Edge Removal

The removal of an edge is determined by its age, its weight, and its position in the graph.

The *age of edge* e_{ij} connecting nodes i and j, is defined similarly to the age of a node, and is denoted by $\tau_{ij} = t - t_{ij}$, where t is the current time and t_{ij} is the time of its creation. The aging factor of edge e_{ij} is a decaying function $f(\tau_{ij})$ and can be, e.g., an exponential function.

The *weight of edge* e_{ij}, denoted by w_{ij}, is one of the parameters for edge removal, since interactions with a high cost are more important for the computation of reputations, edges with high weights have to be preserved in the graph.

The *importance of the position of edge* e_{ij} in the graph is expressed by its edge betweenness centrality (BC), denoted by $C_E(e_{ij})$, which is defined as the sum of the ratios of shortest paths between all pairs of nodes containing this edge [55]. The aging factor and the weight of an edge represent its contribution to the computation of reputations, while its C_E helps in preserving the structure of the graph.

Similarly to node removal, we express the priority score of removing an edge e_{ij} as

$$P_e(e_{ij}) = \alpha P_S(w_{ij}, \tau_{ij}) + (1-\alpha) P_F(C_E(e_{ij})), \tag{4.3}$$

where α is the parameter used in the definition of P_n to control the topology of the derived graph. The scores P_S and P_F are defined similarly to P_A and P_B, respectively. Therefore, edges with lower age, lower weight, and lower betweenness centrality will be removed.

The basic computational components of reducing the history consist in the computation of BC (we do not distinguish between node and edge BC because the algorithm is the same). Computing the degree, the aging factor of nodes, the weight, and the aging factor of edges has a linear cost on the number of nodes and edges respectively and can be computed incrementally. However, the computational cost of BC is high (for unweighted networks it is $O(mn)$ where n is the number of nodes in the network and m the number of

edges). The cost can be significantly reduced by using approximations [58] and exploiting the structure of the network. In particular in scale-free networks, the BC values do not have to be updated very often with the network growth [63] and in networks without community structure, the BC of a node shows a strong correlation with its degree. Note that the reputations of nodes are computed by the core reputation mechanism.

4.3 Datasets

In order to assess our method for creating the complete history, we consider both synthetic graphs and graphs derived from real networks. In our synthetic complete history graphs we consider two processes that occur simultaneously: first, new nodes enter the system, and secondly, the already existing nodes interact, thus creating new links. Thus, we define the probability p_c which represents the probability of adding a new node at each time step to the graph, and the probability $1 - p_c$ which represents the probability of adding new links between existing nodes. In highly dynamic systems, the appearance of new nodes is dominant, and so the value of p_c is high. In our models for synthetic graphs, we allow the occurrence of multiple edges between a pair of nodes and we consider the number of multiple edges as the weight of that edge.

For our experiments, we create the complete history G and the corresponding reduced history G' in parallel. In the complete history, we store all the new information. For the construction of the reduced history we keep its size (almost) constant to a maximum number of nodes n_{max}, which represents the computational or memory limitation of the system. We control the size of the reduced history by removing nodes or edges from the graph as new information is stored as described in the previous section. Below, we describe in detail our models for random graphs and scale-free graphs, the properties of the Bartercast and Citation graph, and the construction of the corresponding reduced histories.

A **random graph**, denoted by $R(n, p_r)$, is composed of n nodes, and each potential edge connecting two nodes occurs independently with probability p_r. Based on this model, we generate a growing directed random graph $R(n_t, p_r)$ representing the complete history of interactions.

To create $R(n_t, p_r)$ with n_t nodes at time t, starting from a single node, we perform the following two operations at each time step:

- With probability p_c we add a new node with each of its potential directed edges existing with probability p, for some value of p.

- With probability $1 - p_c$ we add pn_t new directed edges adjacent to chosen existing nodes uniformly at random.

One can prove that $p_r \sim p/2p_c$ in the following way.

We start with one initial node with no edges. Then, we start building $R(n_t, p_r)$, and at time $t > 0$, the expected number of nodes is $n_t = 1 + p_c t$. Since the probability of connection is p, the expected number of edges at time t is:

$$E(t) = \sum_t p n_t = \sum_t p(1 + p_c t) = p(t + p_c t(t-1)/2).$$

Thus, the probability of connection in the random graph is equal to

$$E(t)/(n_t(n_t - 1)) \approx (pp_c t^2/2)/(p_c^2 t^2) = p/2p_c$$

for large t, which proves that our procedure creates a random graph $R(n_t, p/2p_c)$.

In accordance with R, we create the reduced history graph R'. The reduced history R' is equal to R up to the maximum number of nodes n_{max}. After having reached n_{max} nodes, R' is maintained by performing the following operations at each time step:

- When a new node is added to R, we also add this node to R' along with its edges, and then we remove one node together with its edges with the highest priority score (Eq. (4.1)).
- When new edges are added to R, we add the same edges to R'. Then we remove from R' the same number of edges with the highest priority score (Eq. (4.3)).

Note that some edges in R may be adjacent to nodes that have been removed from R'; in this case, these edges are not added to R'.

Scale-free graphs are characterized by their degree distribution following a power law. We create a growing directed scale-free graph based on the preferential attachment model [14]. Similarly to the procedure for random graphs, we generate two directed graphs S and S' corresponding to the complete history and the reduced-history.

We create $S(n_t)$ by starting with a small seeding graph with m_0 nodes connected by $m_0 - 1$ edges and then performing the following steps:

- With probability p_c we add a new node with m directed edges, with $m \leqslant m_0$. Each edge is adjacent to an already existing node i with probability $\Pi(i) = d_i / \sum_j d_j$, where d_i is the degree of node i.
- With probability $1 - p_c$ we add m new directed edges. Each of these edges are adjacent to an existent node i with probability $\Pi(i)$.

In line with S, we build the reduced history S' using the same procedure as for random graphs.

One can prove that S is scale-free with power-law exponent equal to $\gamma = 1+2/(2-p_c)$ in the following way. The proof of S being a scale-free graph is based on the mean-field theory proposed by Barabási and Albert [14]. With $p_c = 1$ we have the classic Barabási-Albert model, where only a new node is added and the exponent of power-law is $\gamma = 3$. We start with one initial node and then, to construct our scale-free graph, we follow the constructive process described in Section 4.3. With probability p_c we add a new node with m edges, and so the degree of node i, denoted by d_i, changes with rate: $\partial d_i/\partial t = m d_i/\sum_j d_j$. With probability $1 - p_c$ we add m new directed edges and the degree of node i changes with rate: $\frac{\partial d_i}{\partial t} = 2m d_i/\sum_j d_j$. Therefore, in total:

$$\frac{\partial d_i}{\partial t} = p_c m \frac{d_i}{\sum_j d_j} + (1-p_c)2m\frac{d_i}{\sum_j d_j} = (2-p_c)m\frac{d_i}{\sum_j d_j}. \tag{4.4}$$

Moreover, $\sum_j d_j = 2E(t) = 2mt$, where $E(t)$ is the number of edges in the graph at time t, so we can solve Eq. (4.4) for d_i and find:

$$d_i = m\Big(\frac{t}{t_i}\Big)^{(2-p_c)/2}, \tag{4.5}$$

where t_i represents the time that node i joined the network. Using Eq. (4.5), the probability $P[d_i(t) < d]$, that a node i has a connectivity d_i smaller than d, can be written as $P[d_i(t) < d] = P\Big(t_i > (m/d)^{2/(2-p_c)}t\Big)$.

We assume that each operation of either adding a new node or a set of edges takes one unit of time, and so the probability density of t_i is $P_i(t_i) = 1/(m_0 + t_i)$. Thus,

$$P\Big(t_i > \Big(\frac{m}{d}\Big)^{2/(2-p_c)}t\Big) \;=\; 1 - P\Big(t_i \le \Big(\frac{m}{d}\Big)^{2/(2-p_c)}t\Big) = 1 - \Big(\frac{m}{d}\Big)^{2/(2-p_c)}\frac{t}{m_0+t}.$$

The degree distribution is the probability density for $P(d)$, thus we obtain:

$$P(d) = \frac{\partial P[d_i(t) < d]}{\partial d} = \frac{2m^{2/(2-p_c)}}{(2-p_c)}\frac{1}{d^{2/(2-p_c)+1}}\frac{t}{(m_0+t)},$$

and as a consequence, for large t, $P(d) \sim d^{-\gamma}$ with $\gamma = (2/(2-p_c)+1)$.

The **Bartercast graph**, denoted by B, includes information from 29,716 nodes from September 1, 2010 to January 31, 2011. Bartercast presents high population turnover and thus, the derived graph consists in a dense core with very few long living and active nodes and a periphery with many loosely connected nodes of low activity (small average path length and small clustering coefficient, see Table 4.1). The addition of new nodes/edges in B is based on the actual timestamps of the crawled database of Bartercast. Similarly to the procedure for random and scale-free graphs, we maintain the reduced history B' by

Table 4.1: The average path length (L) and the clustering coefficient (cc) of the largest connected component of the Bartercast and Citation graph, and of the corresponding random graphs with similar average path length.

Graph	# Nodes	# Edges	L	cc	L_{rand}	cc_{rand}
Bartercast	$10,634$	$31,624$	2.64	0.00074	2.63	0.0032
Citation	$15,360$	$365,319$	3.29	0.1098	3.31	0.0012

removing nodes and edges using Eqs. (4.1) and (4.3) as new nodes and edges are added according to the timestamps.

The author-to-author **Citation graph**, denoted by C, is derived from the citation network of 21,858 papers from January 2001 to November 2011. Unlike Bartercast, the graph C is derived from a centralized system with global reputations. In Table 4.1, we can see that graph C exhibits small-world behavior with small average path length and large clustering coefficient. Its degree distribution has a power-law tail with exponent $\gamma = 2.55$. As described for the Bartercast graph, we create the complete history C and the corresponding reduced history C' based on the actual timestamps in the database of the Citation graph.

4.4 Computation of Reputations and Evaluation Metrics

We consider two methods for computing reputations: the max-flow algorithm and the random walk-based computation of reputation. However, our approach can be generalized to other methods for computing reputations as well.

The **max-flow algorithm** [35] computes the maximum flow passing between two nodes and is the core of many reputation systems (such as Bazaar [98], Bartercast [43], and the system proposed by Feldman et al. [53]) because it provides resilience to misreporting by nodes who may exaggerate their contributions to increase their reputations. In our study, we use the definition of reputation of Bartercast mechanism [63] since we use a graph derived from it for the evaluation of our approach.

Random walks constitute the core of many reputation and recommendation mechanisms (such as EigenTrust [78], PageRank [92], TrustRank [69] and many others). The basic idea of random walk-based computations of reputations is that interactions with highly reputed nodes contribute more to the reputation of a node. In our analysis, we use PageRank computed using the power iteration: $r_{t+1} = dAr_t + [(1-d)/N]\mathbf{1}$, where A represents the normalized adjacency matrix of the network, r_t the ranking vector at time step t, d the damping factor (we set it equal to its typical value 0.85 [92]), N the number of nodes, and $\mathbf{1}$ the vector of length N containing only ones. In some networks like

Bartercast, an incoming edge of a node has a negative meaning for the reputation of that node (because a weighted edge represents the amount of transferred data and so, adds to the reputation of the donator of the data). Therefore, in these networks, first we reverse the direction of links before we apply PageRank (reverse PageRank [13]).

The evaluation of our method is based on the observations that for the vast majority of reputation systems, the ranking of nodes according to their reputations is more important than the actual reputation values themselves, and that in many systems the identification of the highest ranked nodes is more important than of the rest of the nodes. Therefore, we define the *ranking error* as the difference between the rankings of the nodes according to their reputations in the reduced history and the complete history. More precisely, we consider the sequences of the Unique Identifiers (UIDs) of the nodes in the reduced and the corresponding complete history of our graphs, and we compute the minimum number of inversions of consecutive elements needed in the sequence of the reduced history to get all the common nodes in their correct order in the complete history. This minimum number of inversions is then normalized over the worst case, which would occur if the ranking would be completely reversed. Furthermore, to explore the ability of the reduced history to identify the highest ranked nodes, we define a second metric called the *ranking overlap* which is defined as the fraction of nodes the sequences of the top-5%, 10% and 20% ranked nodes in the reduced history and the corresponding sequences in the complete history have in common. More precisely, we compute the ranking overlap as $|\mathcal{U} \cap \mathcal{V}|/|\mathcal{U}|$, where $\mathcal{U}$ is the set of the top-5%, 10% and 20% ranked nodes in the reduced history and $\mathcal{V}$ is the set of the top ranked nodes in the complete history of size $|\mathcal{V}| = |\mathcal{U}|$.

4.5 Evaluation

In this section, we present our experimental evaluation. First, we explore the effect of each of the parameters for node and edge removal separately and in combination. Next, we study the effect of the size of the reduced history relative to the size of the complete history. Finally, we evaluate the effect of the growth of the complete history while the size of the reduced history is constant. In our experiments, we use the synthetic and real-world graphs introduced in Section 4.3. Our synthetic graphs consist of $5,000$ nodes with α and p_c neutral (both equal to 0.5), unless other initializations are mentioned. We choose the other parameters for the random graph ($p_r = 0.02$) and the scale-free graph ($m = 3$ and $\gamma = 2.2$) so that they roughly correspond to the Bartercast graph. For the synthetic graphs, our results presented in each plot are the average of 25 independent experiments, while for the Bartercast and Citation graphs, we conduct only one experiment since we have only one instance of these real-world graphs.

We first explore the effect of the parameters for node and edge removal defined in Section 4.2 on the ranking error. To explore the effect of the parameter α, we remove

50% of the nodes and edges of the complete history according to Eqs. (4.1) and (4.3) for different values of α. In particular, for random graphs using max-flow (or Pagerank), the ranking error starts at 0.33 (or 0.21) for α equal to 0, and it slightly decreases by 0.02 (or 0.01) until α is equal to 0.8. As α increases further, the ranking error increases by 0.07 (or 0.06). A similar stable behavior for the ranking error is observed for the scale-free and real-world graphs. Since α doesn't affect the performance of the reduced history much we take it as neutral, equal to 0.5, for all the following experiments.

4.5.1 Experiments and Results

Next, we explore the effect of the parameters for node and edge removal separately, and their combination as defined by Eqs. (4.1) and (4.3). For the parameters for node or edge removal, we remove fractions nodes or edges of the complete history using only one parameter at a time. The effect of these parameters on the ranking error is plotted in Figure 4.2 for the random and scale-free networks. We observe that creating the reduced history using only node removal results in similar performance as edge removal for the corresponding parameters. This is to be expected as there is a correlation between these parameters: in general, an edge with high BC is adjacent to nodes with high BC, an old edge is attached to old nodes, and an edge with a large weight is adjacent to a node with high reputation. Furthermore, the combination of all parameters in Eqs. (4.1) and (4.3) results in the smallest ranking error. The largest ranking error occurs when we remove nodes based on their age. The reputation of a node depends on the period it participates in the system and thus, when only new nodes with low reputations participate in the reduced history, the ranking error is high. All the other parameters cause quite similar ranking errors because they exhibit correlations in graphs without strong community structure, such as the random and scale-free graphs. In the real-world graphs, the parameters for node and edge removal and their combination exhibit similar relative performance as in the scale-free graphs. Since the combination of the parameters for node and edge removal achieves the lowest ranking error, we use it to create the reduced history for all the following experiments.

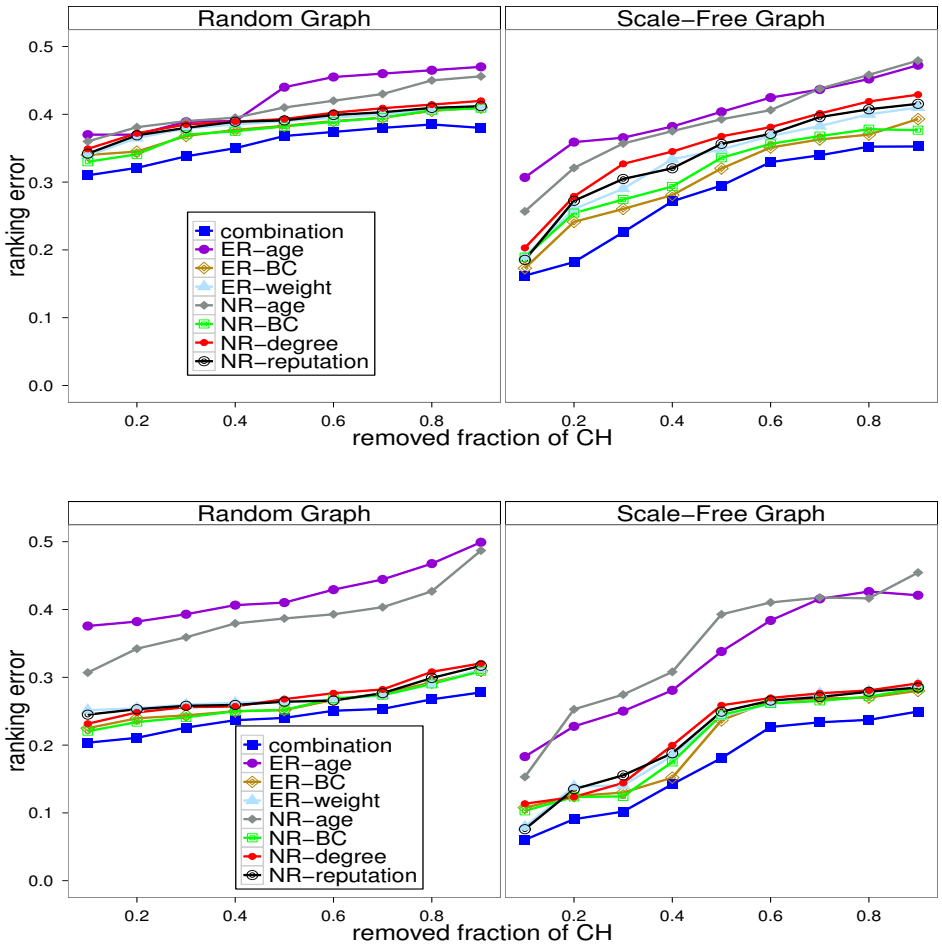


Figure 4.2: The effect of the parameters for node and edge removal when removing a fraction of the nodes and edges of CH for random and scale-free graphs when the reputation algorithm is max-flow (top) and Pagerank (bottom). The indication ER in the legend denotes parameters for edge removal and NR parameters for node removal.

We next evaluate the effect of the size of the reduced history relative to the size of the complete history on the ranking error and the ranking overlap. For this purpose, we construct reduced histories of different sizes for a complete history of fixed size as described in Section 4.3. Figure 4.3 (left) plots the ranking error for different relative sizes of the reduced history. We observe that when using max-flow, the scale-free, Bartercast and Citation graphs exhibit much smaller ranking error than the random graphs. For all the graphs using Pagerank, the reduced history exhibits smaller ranking error than using max-flow. Figure 4.4 plots the ranking overlap for different relative sizes of the reduced history. The scale-free and Bartercast graphs exhibit much higher ranking overlap than the random and Citation graphs when using the max-flow based algorithm. Particularly, in these networks the ranking overlap decreases quite slowly with the decrease of the size

of the reduced history, until the size of the reduced history is about 0.4 of the complete history. The reason is that these networks have a large amount of redundant information for approximating the highest ranked nodes when using the max-flow algorithm. When the size of the reduced history is smaller than 0.3 of the complete history, the ranking overlap degrades quickly. With Pagerank, the reduced history exhibits very low ranking overlap for all the graphs.

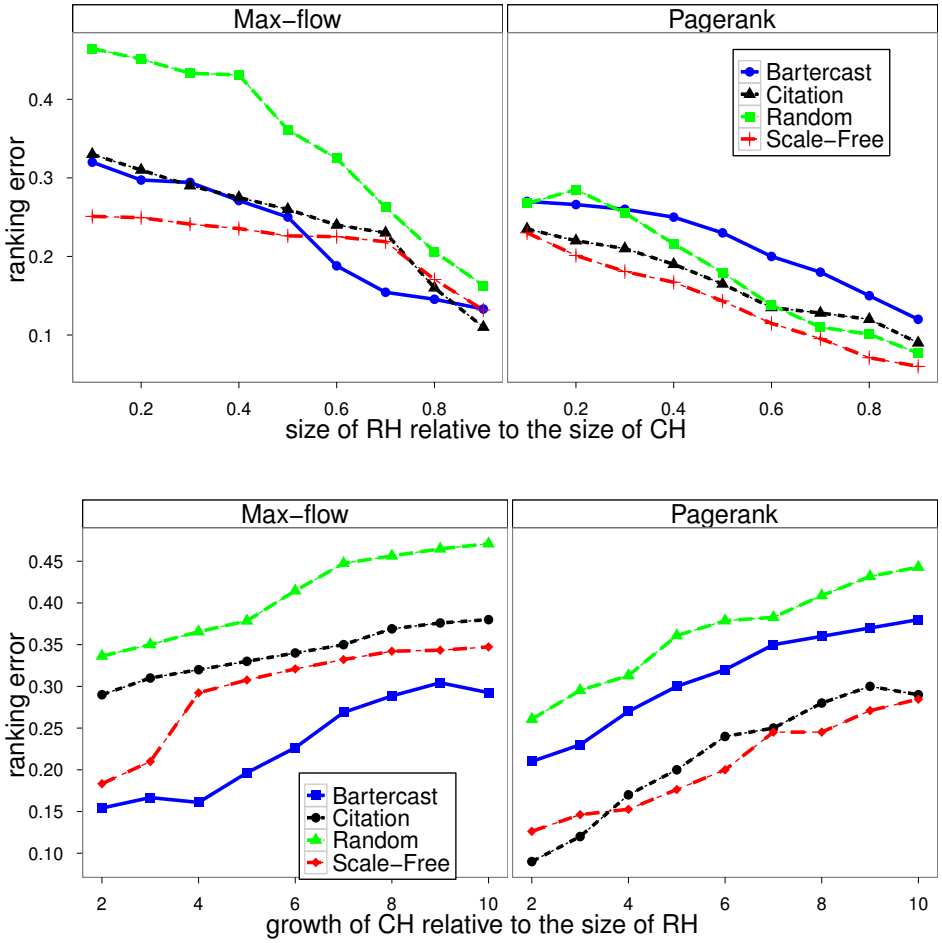


Figure 4.3: The effect of the size of RH relative to the size of CH (top) and the effect of the growth of CH relative to the size of RH (bottom).

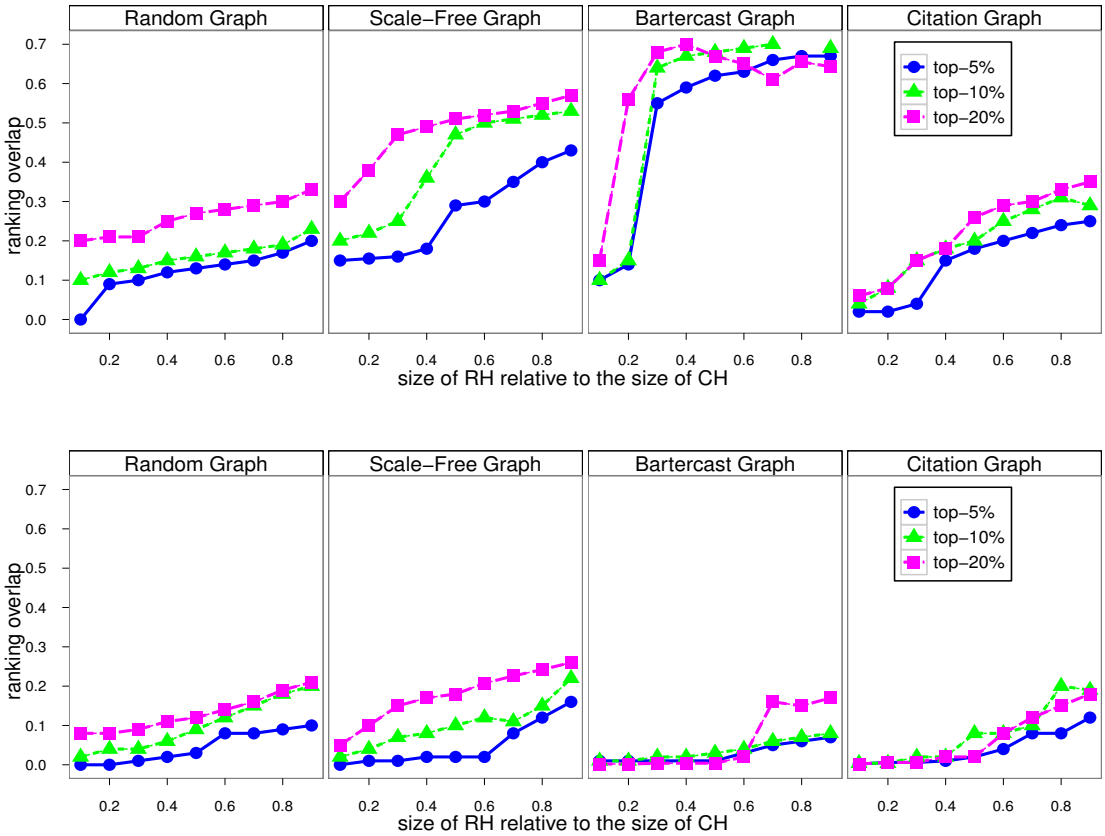


Figure 4.4: The effect of the size of RH relative to the size of CH for max-flow (top) and Pagerank (bottom).

Finally, we evaluate the effect of the growth of the complete history while the reduced history is of constant size on the ranking error and the ranking overlap. For the synthetic graphs, we let the complete history grow from 500 to $5,000$ nodes while we keep the size of the reduced history constant at 500 nodes. For the real-world graphs, using the available temporal information, we have the Bartercast graph grow from $1,063$ to $10,634$ nodes with the reduced history constant at $1,063$, and the Citation graph from $1,536$ to $15,360$ nodes with the reduced history at $1,536$. Figure 4.3 (right) plots the ranking error and Figure 4.5 plots the ranking overlap for different relative growths of the complete history. We observe again that Pagerank achieves a smaller ranking error while the max-flow based algorithm achieves a better ranking overlap, specially for the scale-free and real-world graphs.

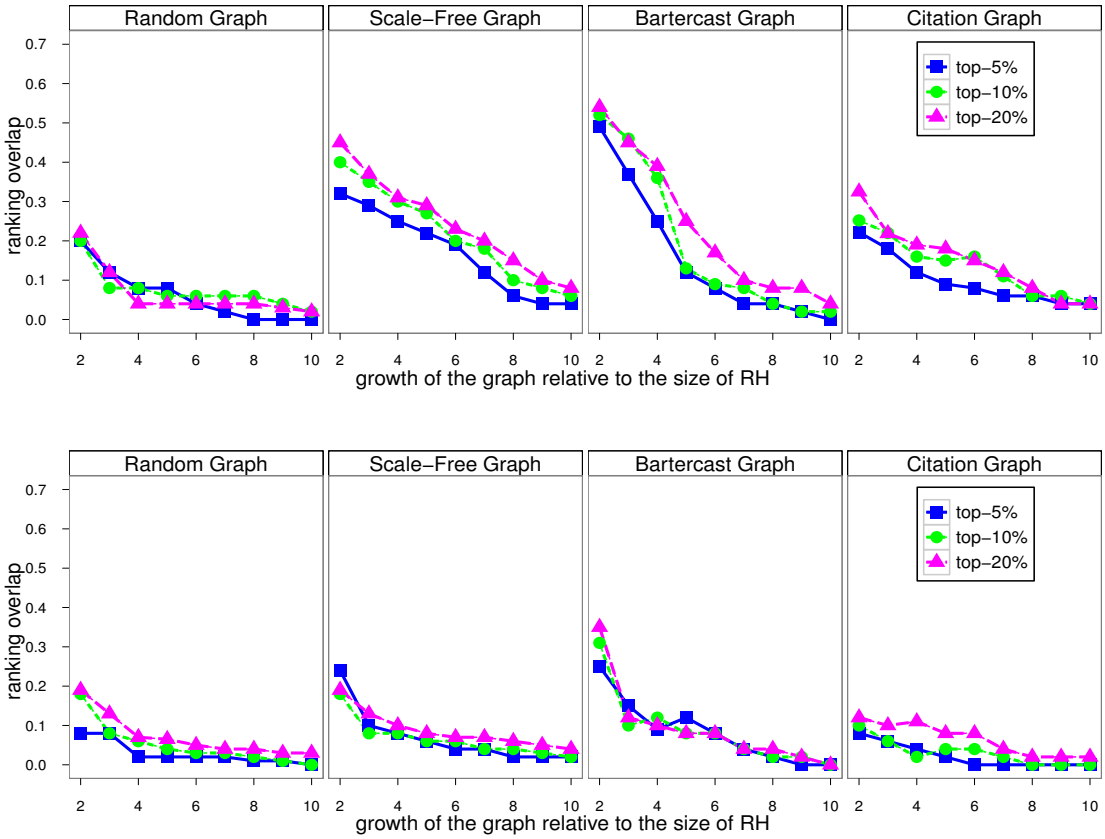


Figure 4.5: The effect of the growth of CH relative to the size of RH for max-flow (top) and Pagerank (bottom).

4.5.2 Discussion

The observations arising from our experiments indicate that the reduced history can give a good approximation of the ranking of nodes according to their reputations when the complete history exhibits a particular structure. In this subsection, we explain and discuss our main observations.

First, we observe that constructing the reduced history using the combination of all the parameters for node and edge removal results in the lowest ranking error. Considering only parameters such as degree and reputation gives priority for removal to the newest nodes and so, new nodes will not participate in the reduced history. On the other hand, considering only the age as parameter for removal results in high ranking error because then, only new nodes participate in the reduced history and information of old important nodes has been removed. Therefore, for good performance of the reduced history, it is required to use a combination of these parameters as defined by Eqs. (4.1) and (4.3).

Secondly, the performance of the reduced history depends on the topology of the

graph, and is better in the scale-free, Bartercast and Citation graphs than in the random graphs. The scale-free and our real-world graphs have only a few well connected nodes accumulating the majority of links, while the vast majority of nodes has a very low connectivity. In the reduced history, the highly connected nodes are preserved keeping their good ranking position, while most of the loosely connected nodes have been removed. In contrast, in random graphs all nodes have stochastically similar connectivity properties. Since most real networks exhibit heterogeneity in the connectivity properties of their nodes [6], we can conclude that the reduced history can be applied in a large range of networks.

Finally, the performance of the reduced history depends on the reputation algorithm used. In particular, it causes a lower ranking error when using Pagerank, while it achieves a higher ranking overlap when using max-flow. Pagerank computes the reputation of a node by aggregating the interactions of all nodes participating in a graph. The aggregative computation of centrality by Pagerank achieves lower ranking error even if the reduced history has a relatively small size. Unlike Pagerank, the max-flow based algorithm computes the reputation of a node taking into account only the interactions between that node and the most central node. Since both the most central and the highest ranked nodes are considered as important, they are preserved in the reduced history. Therefore, we achieve a high ranking overlap when using the max-flow based algorithm.

In conclusion, our observations demonstrate the effectiveness of the reduced history in approximating the ranking of nodes with Pagerank and in identifying the highest ranked nodes with the max-flow based algorithm. This implies that the reduced history can approximate with reasonably accuracy the complete history in real world graphs, while it has much smaller resource requirements. As we stated in Section 4.1, this result is valuable especially for decentralized systems, such as Tribler, because of the limited resources available at each node.

4.6 Related work

The observations of our experiments are consistent with the findings of prior published research for the robustness of centrality measures under sampling or missing data. In particular, our finding that node and edge removal cause similar ranking errors has been discussed in the context of the robustness of centrality measures under missing data [18]. In the context of network sampling, it has been observed that ranking nodes with random walks is highly robust [36]. Moreover, the result that the use of BC for node and edge removal does not affect the ranking error much, has been also observed under the context of edge removal for security reasons [129]. However, our approach is different from sampling techniques, since sampling techniques focus on creating a static subgraph with similar properties as the original graph. In our case, we need to maintain the reduced his-

tory dynamically with the growth of the original graph, and we are interested in producing a reduced history that preserves the reputations of nodes and not necessarily the general properties of the original graph.

4.7 Conclusion

Using the complete history of interactions in a reputation system is not efficient due to its high computational cost and high memory requirements, and to the high population turnover. We have proposed the use of the reduced history instead of the complete history defining the main parameters for choosing the nodes participating in it. Next, we have evaluated our approach experimentally exploring both theoretical graph models and real-world graphs using two reputation algorithms, a max-flow based algorithm and Pagerank. We conclude that for scale-free and real-world graphs, the reduced history is reasonably accurate while for random graphs, due to their structural properties, the reduced history causes high error. Furthermore, we have demonstrated that using the max-flow based algorithm results in better identification of the highest ranked nodes while using Pagerank results in better ranking error.

Chapter 5

Trust-based Collection of Information

In decentralized reputation systems users have only a limited view of the system. As a result, each user has to collect information about the past interactions of other users in order to compute their reputations. The collection of information directly affects both the quality and the cost of a reputation system. Without any central coordination, the collection of reports about user interactions is challenging. In such systems, the ease of creating accounts enables malicious users to create numerous fake identities, their Sybils, and spread false reports about their interactions. Furthermore, the large number of interactions between users causes an information overload. By blindly storing and processing information, users easily become victims of Sybil attacks as well as waste their resources for information that contributes too little to the reputations. To avoid the impact of malicious nodes and the misuse of their resources, users have to collect *trusted* and *relevant* information.

In this chapter, we propose a method to collect information in decentralized reputation systems. While traditional methods against Sybil attacks, such as SybilGuard [126], and SumUp [111], require the existence of a social network and interpret the social connections between users as trust, we assume no social network. As a result, our approach is suitable for systems without a social network such as P2P networks or markets on mobile devices [27]. Our only assumption is the existence of interactions between users, which are interpreted as indicators of trust and similarity between the corresponding users. Our method, EscapeLimit, constrains the ratio of the collected interaction reports over all the interactions, *recall*, by collecting only relevant and trusted information. EscapeLimit uses random walks with restarts to successively visit nodes to collect interaction reports. In this way, it exploits the transitive flow of positive interactions and guarantees a link between the creator of a report and the user who uses it in a reputation calculation. The restart probability determines the recall and the vulnerability to Sybil attacks.

In EscapeLimit, each node knows its own direct interactions, and using its locally stored interaction reports, it creates its *interaction subgraph*. Using its subgraph, each

node calculates the reputations of other users. Nodes periodically contact each other through random walks to collect new reports and expand their interaction subgraphs. Every node maintains a list of neighbours, consisting of currently online direct interaction counterparts. Upon request, it introduces those counterparts to other nodes. EscapeLimit reduces the *escape probability*, which is defined as the probability that a random walk initiated by an honest node ends up in a Sybil node.

We evaluate EscapeLimit by emulating user interaction patterns derived from networks with different properties. In our experimentation, we allow nodes to collect information while they perform interactions. Particularly, we use a synthetic power-law network and two real-world networks, one network deriving from the Internet-deployed Bartercast reputation system [43] used in the BitTorrent-based client Tribler [99], and one network deriving from Facebook [115]. In our evaluation, the computation of reputations is trivial, namely the ratio of the contribution of a node to the network over its consumption. In this way, we can determine the quality of the collection of information independently of the computation of reputations. The main contributions of this chapter are as follows:

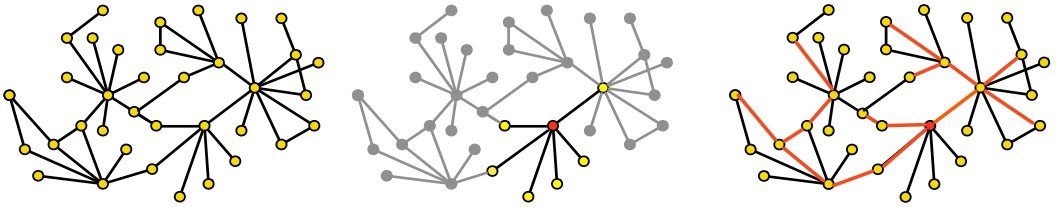
1. We evaluate EscapeLimit in terms of its resilience to Sybil attacks, its scalability, and its ability to collect relevant information, and we show its efficiency.

2. To further enhance EscapeLimit, we bias random walks with trust-driven properties such as the strengths of user interactions and the activity levels of users, and we explore their effect on the collection of information.

3. We demonstrate that the performance of EscapeLimit depends on the topology of the interaction graph and that EscapeLimit biased with the strength of user interactions is efficient in almost any type of graph.

5.1 Problem Statement

Initially, a node knows only its own *direct interactions* and creates its interaction subgraph as described in Section 1.3. We present an illustration of the interaction subgraph of a node in Figure 5.1a and 5.1b. As a result, it is able to compute only the reputations of nodes with which it has previously interacted. Repeated interactions are not frequent and so, nodes need to expand their interaction subgraphs in order to compute the reputations of other nodes in the network. In eBay 89% of transactions between a pair of buyer-seller [44] and in P2P systems about 92% of data transfers between a pair of peers [96] have been conducted only once. Even in networks such as Facebook, where pairs of friends repeat their interactions often, yet users interact with unknown contacts. For this



(a) The interaction graph

(b) The interaction subgraph of the red node, the part of the interaction graph drawn with grey indicates the unknown area from the perspective of this node

(c) The red node learns the complete interaction graph after having walked to several nodes, the red line indicates the walk steps

Figure 5.1: The red node collects information using its interaction subgraph. (Figures 5.1a and 5.1b have also appeared in Section 1.3)

purpose, they should periodically contact each other acquiring information about the interaction subgraph of other nodes. Ideally, after a node has contacted several other nodes, its interaction subgraph converges to the interaction graph. The information spreading method is crucial for the quality of the computed reputations, since poor information spreading results in inaccurate reputations [42]. Our goal is the design of an applicable information spreading method. Next, we define three general requirements for an applicable spreading method in decentralized reputation systems.

5.1.1 Resilience to Attacks

From the various self-promoting attacks in a decentralized reputation system, we consider only sybil attacks, as most other types (e.g. link farming, collusion, spam), can been seen as special cases of it. Furthermore, simple misreporting can be easily eliminated by assuming that after each positive interaction, the corresponding interaction report is cryptographically signed. The sybil attack is predominant in reputation systems. It has been reported that in Facebook more than 1.5M fake accounts have been identified during February 2010 [25], in RenRen more than 660K fake accounts [123], and in Tuenti about 180K fake accounts [25]. Acquiring polluted information results in inaccurate reputations.

5.1.2 Scalability

In distributed systems, scalability has been defined along three dimensions: fault-tolerance, system load, and administration [90]. The *fault-tolerance* of a system is affected by its size. The probability that some nodes are unavailable for communication increases with its growth, particularly in systems with high population turnover such as

P2P networks. Fault-tolerance requires high autonomy of nodes so that they can function despite failures of other nodes. *System load* in terms of computational cost and communication overhead at each node, increases with the size of the network. Due to the limited resources available at nodes, the method of information spreading should not add a lot of computational and communication overhead. Moreover, it should distribute the communication load evenly across all the nodes. Particularly, in many real world networks where a few nodes have the majority of connections, balancing communication load is challenging. Finally, the *administrative* control on the information about other nodes, e.g. connectivity, maintained at each node becomes impractical with the growth of the network. Administration of information at each node is easier when each node maintains information only from its neighbourhood.

5.1.3 Relevance of Information

All the information about node interactions does not contribute equally to the computation of reputations. Interactions of high strength occurring closer to a node contribute more in the computation of reputations than interactions of low strength in the periphery of the network. Furthermore, a node is more likely to interact with highly active nodes closer to it. Computing the reputation of these nodes accurately is more useful. Therefore, each node must acquire *relevant* information so that the reputations computed from its interaction subgraph are close to the reputations of the interaction graph.

5.1.4 Trade-off among the Requirements

These requirements cannot be completely satisfied by one method since they are conflicting to some extent. For instance, in a perfect attack resilient solution, each node should only contact nodes with which it has previous successful interactions. However, this results in very poor collection of relevant information since that node fails to compute the reputations of potential new encounters. Furthermore, a node obtains fast relevant information if it contacts the highly active nodes often, since those nodes keep the network connected and perform the largest number of interactions. However, this results in overloading these nodes. Nevertheless, by not considering all these requirements, a collection mechanism is not applicable to online distributed systems and so, it has to make a trade-off among them.

5.2 Design Considerations

Having presented the three requirements of information spreading in distributed reputation systems, we proceed with the design considerations of EscapeLimit regarding the

Table 5.1: The different design considerations grouped by their ability to satisfy the requirements for a collection mechanism applicable to distributed reputation systems

Method	Resilience to Attacks	Scalability	Relevance
communication protocol			
epidemics	✗	✓	✗
similarity of taste	✗	✓	✓
DHT	✗	✓	✗
random walks with restarts	✓	✓	✓
incorporation of trust			
social network	✗	✓	✗
interactions subgraph	✓	✓	✓
direction of information			
push	✗	✓	✗
pull	✓	✓	✓
type of information			
direct intractions	✓	✓	✓
direct-indirect interaction	✗	✓	✗

communication protocol, the incorporation of trust, the direction of information spreading and the type of information spread. The design of EscapeLimit is directed by the three requirements which must be considered from the beginning of the design. Otherwise, they can be irrelevant.

5.2.1 Collection of Reports

Four different approaches have been widely used in the core of communication protocols in decentralized systems: gossip or epidemics, methods based on similarity of taste, Distributed Hash Tables (DHT), and random walks. According to a typical epidemic protocol, each node disseminates its fresher information to a randomly chosen node [79]. Epidemics are resilient to failures. However, the recipients of information cannot asses its validity and so, epidemics are vulnerable to sybil attacks. No notion of trust can be incorporated in epidemics due to the randomness on the node selection. Furthermore, the disseminated information is not relevant in most cases resulting in waste of bandwidth and node resources. To reduce the communication cost, epidemic-based methods such as Credence for LimeWire p2p client [119], and SimilDis [42], disseminate information among nodes with similar tastes. These methods manage to reduce radically the communication cost but still they are not secure.

According to DHT, the reputation of a node is stored and maintained by other nodes determined by a hash value. This approach has been used in systems like DHTrust [122]

and EigenTrust [78]. DHTs are vulnerable to a great range of attacks. Even though many techniques have been proposed for secure DHTs, securing DHTs is still a challenging problem [114]. Furthermore, they have to deal with problems such as load balancing and population turnover.

Random walks are computationally tractable, and naturally decentralized using only information locally available at each node. While proposed solutions fail to meet the three requirements of a distributed reputation system, random walks model the different levels of trust and similarity among nodes [65], [89]. Random walks are flexible and with the appropriate biases, they are able to quickly detect relevant and trustworthy information in a network. As a result, they have been widely used in distributed systems for search [61], topology maintenance, and computations of reputations such as Eigentrust [78], and SybilRank [25].

In EscapeLimit, we will use random walks with restarts [110], where a random walk may be directed back towards its initiator with a fixed *restart probability*. A random walk with restarts represents better the inherent trust in a network, since each node trusts itself more than the other nodes and its trust towards the other nodes decays with the increase of their distance [65, 89]. We present an implementation of random walks suitable for internet deployed networks in Section 5.3. Each node in the network performs its own random walks and requests parts of the interaction subgraphs of the contacted nodes.

5.2.2 Incorporating Trust

In many proposed systems such as SybilRank [25] and SybilInfer [39], random walks enhance their resilience against sybil attacks by using a social network to incorporate trust. In this way, it is assumed that a connection with a node in the social network reflects trustiness towards that node. Each node performs its random walks across the social links in the network. However, in many systems such as P2P networks no social network is available. Furthermore, in social networks many social connections between nodes are superficial or of very low strength and thus, they do not indicate trust [120]. For instance, many users in Facebook have much more friendship connections than 150 which is roughly the number of people they can regularly interact [46].

Unlike previously proposed methods, in order to incorporate trust we use the interaction subgraph available at each node. In other words, instead of using social connections between nodes as trust indicators, we use their interactions. Regular and successful interactions between nodes are strong indicators of trust relationships and similarity among users [65], [89]. As a result, EscapeLimit not only is useful to systems without any social network available but it is more effective against sybil attacks, as well. Furthermore, we bias random walks with information deriving from user interactions, such as the strength of interactions.

5.2.3 Direction of Information

The propagation of the interactions subgraphs when using random walks or epidemics can be implemented with two different strategies, *push* (information dissemination) or *pull* (information collection). In the push strategy, a node *sends* (pushes) its received messages towards other nodes in the network while in a pull strategy, it probes another node for messages that it has not received yet, and then it *fetches* (pulls) the corresponding messages. It has been shown that push protocols are resilient to failures and ensure fast propagation of a message to a large portion of the network [52]. However, the propagated information is not always relevant. Moreover, the redundancy of the propagated messages makes push-based mechanism prohibitive for large-scale networks due to its high bandwidth cost. On the other hand, pull protocols result in significantly smaller overhead and ensure the delivery of message in sparsely connected areas of the network. From a security perspective using the push strategy, a node blindly accepting messages from other nodes results in polluting its interaction subgraph with false information, while using the pull strategy, a nodes requests messages from selected nodes following the flow of trust on the interaction graph. As a result, in EscapeLimit we implement pull-based random walks and so, each node *collects* information.

5.2.4 Type of Information Spread

During the collection of information, a node contacting another node requests a part of its interaction subgraph containing either only its direct interactions or both its direct and indirect interactions. The collection of the history of both direct and indirect interactions is equivalent to epidemics and thus, it faces the same problems. Collecting only direct information from other nodes enhances security, scalability, and the relevance of the collected information since it allows control on the received information and decreases the redundancy of information. In EscapeLimit, each node collects the history of direct interactions of the probed node. Figure 5.1 presents an illustration of our method. In Table 5.1, we summarise the design considerations and we group them along their ability to satisfy the three requirements.

In order to show that random walks outperform epidemics against sybil attacks, in Figure 5.2 we present a fast comparison between push-based gossip and push-based random walks with restart probability equal to 0.25 when nodes collect the direct interactions of other nodes. For this comparison, we created two power-law graphs using Barabási-Albert model [14] with each one having 500 nodes, representing the honest and sybil nodes respectively. We connect each honest node with one randomly chosen sybil node. Then, the honest nodes start collecting information from other nodes using either gossip or random walks. We observe that even simple random walks are much more robust against sybil attacks than gossip.

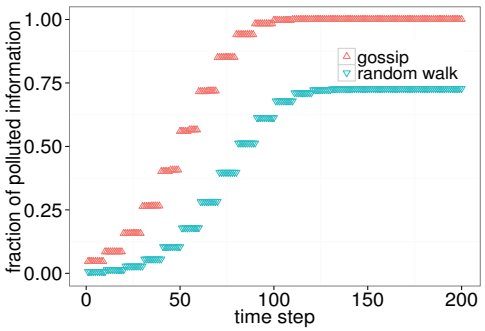


Figure 5.2: The average fraction of polluted information at the honest nodes when using gossip and random walks.

5.3 Collecting Information Using Random Walks

In this section we present the network model, three parameters for biasing random walks, and an implementation of the random walker suitable for internet deployed networks.

5.3.1 Network Model and Definitions

The interaction graph of a network is a weighted directed graph of interactions $G = (V, E)$ whose vertices V correspond to the nodes in the network, and whose edges E correspond to the interactions among nodes. Its adjacency matrix is denoted by $A = \{\alpha_{ij}\}$. A weighted edge $e_{ij} \in E$ connecting $i, j \in V$ in the direction $i \to j$ has a weight w_{ij} which represents the strength of an interaction, for instance the amount of data transferred across edges in a P2P network or the number of interactions in Facebook. We denote r, the vector containing the reputations of nodes. Depending on the application, the vector r can be computed by various computations. We use a very simple computation of reputation so that we can investigate the dissemination of nodes. The reputation of a node j as the ratio of the resources it contributes to the network over the resources it consumed and so, $r(j) = \sum_{k \in N_j} w_{jk} / \sum_{k \in N_j} w_{kj}$ where N_j denotes the neighbours of node j in G. Each node i in the network locally stores its interaction subgraph $G_i = (V_i, E_i)$ with $V_i \subseteq V$ and $E_i \subseteq E$.

A random walk on G is defined by its transition matrix $P = \{p_{ij}\}$, and its stationary distribution π is given by the equation $\pi = \pi P$. Its *mixing time* indicates the time (in walk steps) needed for any initial distribution π_0 to approach the stationary distribution π. To measure the mixing time of our graphs we compute the total variation distance between the two distributions $1/2||\pi - \pi_0 P^t||_1$ over consecutive walk steps t. A fast mixing time implies that the initial distribution converges to the stationary distribution in $O(\log V)$ steps.

5.3.2 Types of Random Walks

We use random walks with restarts with restart probability α. Then, the transition matrix becomes $P' = (1-\alpha)P + \alpha\mathbf{1}$, where $\mathbf{1}$ is the matrix with all its entries equal to 0 except for the elements of the column corresponding to the initiator, which are equal to 1.

In *simple RW* (RW) with restarts, the next step of the walk is chosen uniformly at random among the neighbors of the currently visited node. The transition probability at each step is determined by the adjacency matrix $p_{ij} = \alpha_{ij} / \sum_j \alpha_i$. We use three additional biased random walks.

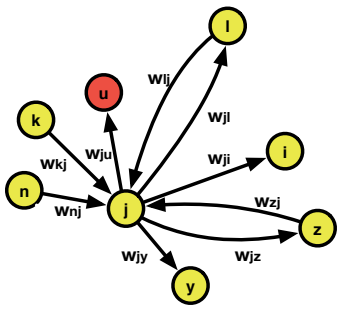
First, we consider a random walk biased towards the strength of interactions assuming that the strength of an interaction reflect both trust and similarity between the adjacent nodes. We call this walk *weighted RW* (wRW) where the bias towards node j from node i is denoted by w_{ij} and $p_{ij} = w_{ij} / \sum_j w_{ij}$.

Then, we define random walks biased towards the nodes with the lowest activity level, namely the smallest degree. In RW high degree nodes are visited with higher probability since more paths lead to them. On the contrary, low activity nodes are rarely visited and by biasing a RW towards them helps in faster spread of their information. Furthermore, it balances the communication overhead among the nodes in the network. This random walk results in uniform visiting probability of nodes and it corresponds to *Metropolis-Hastings Random Walk* (MHRW) [72] for uniform selection of nodes. According MHRW, the probability of visiting node j from node i when $i \neq j$ is defined as $p_{ij} = (1/d_i)\min(1, (d_i/d_j))$ with d_i representing the degree of node i and $p_{ii} = 1 - \sum_j p_{ij}$.

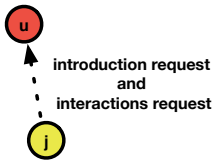
Finally, we consider random walks biased towards the nodes with the highest activity level as defined by their degree. Intuitively, highly active nodes are trustworthy, have fresh information and interact with the other nodes with higher probability. This type of random walk corresponds to *Maximal Entropy Random Walk* (MERW) [24] has been introduced and studied in [23]. The probability of visiting node j from node i is equal to $p_{ij} = (\alpha_{ij} u[j])/(\lambda u[i])$, where u is the principal eigenvector of A, $u[i]$ is the i-th entry of u, and λ is the corresponding eigenvalue. This RW requires global information but recently Sinatra et al. [107] showed that MERW can be accurately approximated by a RW biased towards the degree of nodes, in networks without degree correlations. MERW results in fast diffusion of information since it uses the highly connected node more often.

5.3.3 Implementation of Random Walks

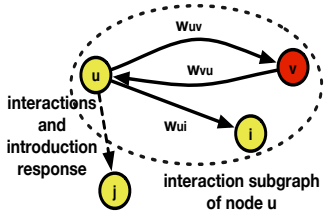
Internet deployed networks such as P2P networks and distributed social networks, are characterized by high dynamics and nodes behind firewalls or Network Address Translators (NAT) boxes. In highly dynamic networks nodes frequently enter and leave the system. As a result, random walks may easily get lost as nodes go offline. Furthermore, nodes behind firewall or NAT are not directly connectable. Even though according to



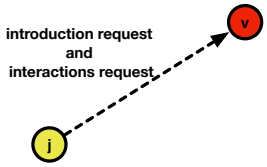
(a) Node j chooses u from its interactions subgraph according to the type of the random walk



(b) Node j sends an introduction and an interaction request to u



(c) Node u sends its direct interactions to j and chooses to introduce node v



(d) Node j completes a walk step and sends an introduction and an interaction request to v

Figure 5.3: Node j performs a walk step.

recent studies, approximately 64% of the nodes in the internet are behind a firewall or NAT [71], classic epidemic protocols assume that all nodes can communicate directly and thus they are ineffective. To face high dynamics, in this implementation of random walks each node controls its own random walks and so, the walks cannot be lost. For NAT traversing, we design random walks using a node introduction mechanism which allows NAT traversal.

We design the random walker with two threads, an active one and a passive one, running concurrently at each node. The active thread is executed by the initiator of the walk and the passive thread by all the nodes contacted during a random walk. We assume that each node is initialized with two global parameters: the restart probability α and the type of the random walk, so that it is able to compute the transition probabilities. In Figure 5.3, we present an illustration of the walker. Next, we describe the implementation.

Conducting a random walk (active thread): The initiator of a walk selects a neighbor according to the type of the random walk from its interaction subgraph (Figure 5.3a) and it sends it two messages: an introduction request and an interaction request (Figure

5.3b). According to the *introduction request*, the initiator of the walk requests from the selected neighbor an introduction to one of its neighbours, namely its contact details. The introduced node will be contacted as the next step of the walk. Next, by sending an *interaction request*, the initiator requests from its selected neighbor the history of its direct interactions. The transferred data packages could be encrypted so that the content cannot be deciphered by third parties. After having received the corresponding responses from its selected neighbor, the initiator has completed a *walk step* and it proceeds to the next step of the walk by sending an introduction and an interaction request to the introduced node. If that node turns out to be off-line, it returns to the previously visited node and asks for a new introduction. As a result, the random walker is robust to the dynamics of distributed networks and random failures. Each node continuously performs walk steps. At each step, the walk might restart with restart probability α.

Reply to requests (passive thead): Each node waits for introduction and interaction requests from other nodes. Upon receiving those requests, a node responses by selecting one of its neighbors from its interaction subgraph according to the type of random walk. Before introducing a node, it checks whether it is online.

This design ensures that our proposed walker is suitable for networks deployed in the internet, where nodes may be behind firewalls or NATs, since UDP hole puncturing mechanisms can be easily implemented during the introduction request. Upon an introduction request, a node selects a neighbour and sends it a puncturing request. Then, the selected neighbour sends a puncturing message to the initiator of the walk. Afterwards, the initiator of the walk is able to send to it an introduction request that traverses the NAT of the introduced node.

For simplicity we allow revisits and so, we do not have to keep in memory the previous steps of the random walk. This could result in sending redundant messages, which could be avoided by using Bloomfilters [29], a probabilistic data-structure testing whether an element of information is already present in a set. In this case, before requesting the history of interactions of a selected node, the initiator of a walk sends it a Bloomfilter with its own part of the history of interactions.

Under extreme churn, the interaction graph of a network might be disconnected and so, nodes might not be able to find walk counterparts among their neighbours. For instance, in Yahoo! Instance Messenger only 5% to 15% of users are online at any time instance during a day and most users stay in the system for limited time [31]. As a result, the corresponding interaction graph is disconnected. In this case, we can integrate in our method the solution proposed in [31] according to which trust is extended to two-hop relationships among nodes. Consequently, each node looks for walk counterparts in its two-hop neighborhood. This solution improves connectivity properties at the expense of resilience to attacks.

Table 5.2: The diameter, the average path length (L) and the clustering coefficient (cc) of real-world graphs.

Graph	# Nodes	# Edges	Diameter	L	cc
Power-law	$1,000$	$5,725$	5	2.92	0.067
Bartercast	$1,000$	$4,723$	8	2.64	0.0065
Facebook	$1,000$	$11,596$	9	3.38	0.13

5.4 Experiment Methodology

We evaluate EscapeLimit with the different types of random walks in terms of its resilience to attacks, its scalability, and its ability to provide relevant information. Particularly, we integrate EscapeLimit into Tribler, a p2p BitTorrent-based client and we run 1000 clients on a computer cluster. Each client emulates the interaction patterns deriving from traces of synthetic and real-world datasets with different connectivity properties and construction patterns. Simultaneously, each client collects information about the interactions of other nodes using EscapeLimit. In this section, we describe our datasets, the experiment setup, and the model to create the sybil attacks.

5.4.1 Datasets

In order to perform our emulations, we use datasets from synthetic power-law graphs and graphs derived from Bartercast and Facebook networks. In the real world graphs, the creation of edges is defined by timestamps available in the corresponding datasets which are expressed in actual time. In the synthetic graphs, we divide time into time steps during which new edges are added, since no notion of actual time exists.

We create a growing directed **power-law graph** based on the Barabási-Albert model [14]. We start with a small connected seeding graph, and at each time step we add a new node with 3 edges whose end points are adjacent to already existing nodes with probabilities proportional to their degrees. After having created a network with 1000 nodes, we continue adding 3 directed edges at each time step, adjacent to existing nodes again with probabilities proportional to their degrees. We allow the occurrence of multiple edges between a pair of nodes and we consider the number of occurrences of an edge as the weight of that edge.

The **Bartercast graph** contains information about 29,716 nodes and their interactions [65]. In order to interpret interactions as trust in Bartercast, we reverse the direction of links since when a user downloads from another user, the corresponding trust flows from the former towards the latter.

The **Facebook graph** contains 63,732 users and their interactions [115].

Figure 5.4: The power-law (left), Bartercast (middle), and Facebook (right) graphs of 1,000 nodes.

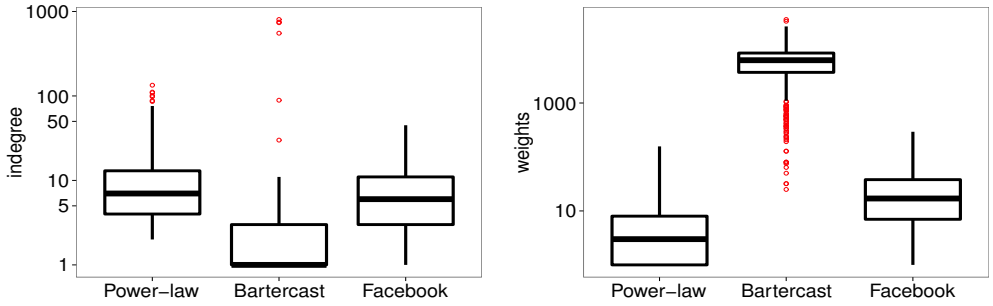


Figure 5.5: The distributions of nodes degrees and edge weights in the power-law, Bartercast, and Facebook graphs

The main difference between the Bartercast and the Facebook graphs, besides their structural properties, is that the former is derived from a deployed distributed system while the latter is derived from a centralized social network. Due to resource limitations of the computer cluster, we conduct our emulations using a strongly connected component of 1000 nodes. We constructed those strongly connected components from the initial graphs by starting from the node with the highest degree a Breadth First Search (BFS) modified so that it traverses only bidirectional edges. The characteristics of the corresponding subgraphs are presented in Table 5.2. All the graphs are small-world characterized by small average path length and small diameter. The Facebook graph forms a tightly connected community. In Figure 5.4, we illustrate the selected strongly connected components (small average path length and high clustering coefficient). On the contrary, the Bartercast graph consists of a few highly connected nodes and many loosely

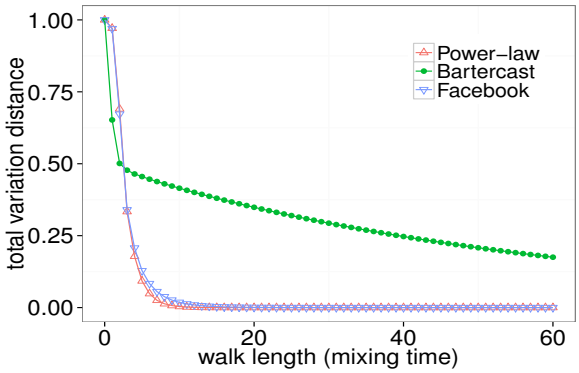


Figure 5.6: The mixing time of the power-law, Bartercast, and Facebook graphs

connected nodes (small average path length and small clustering coefficient), since it has a high population turnover. In Figure 5.5, we show their indegree distribution and the distribution of their weights. We observe that the power-law and Bartercast graphs have a few outlier nodes with high indegree. In this chapter, we use Tukey boxplots where the bottom and top of the box depict the first and third quartiles of the distribution, the band inside is the median. The outliers are identified using the interquartile range (IQR), defined as the difference between the third and first quartiles. Outliers fall below 1.5 IQR from the first quartile, and above 1.5 IQR from the third quartile. The whiskers indicate the range of the distribution without the outliers.

To better understand the community structure of the graphs and interpret the evaluation results of the random walks, we estimate their mixing time. The mixing time of our strongly connected graphs is defined since all the nodes will be visited by a random walk. In Figure 5.6 we present the total variation distance versus the mixing time (walk length) averaged over 1000 initial distributions of a random walk. Facebook and power-law graphs are fast-mixing graphs with tightly connected nodes. On the other hand, Bartercast is slow-mixing because a few highly connected nodes keep the nodes connected by forming clusters around them, as we see in Figure 5.4. Note that the clustering coefficient of a graph indicates its clustering on a local level (the fraction of closed triangles among its nodes), while its mixing time indicates its clustering into large communities.

5.4.2 The Restart Probability

During a random walk, the value of the restart probability α determines its expected length l and as a result, its resilience against sybil attacks and its ability to collect relevant information fast. Even though a large value of α allows the discovery of new nodes at a large depth in the network, it draws the walk away from trusted and similar nodes. The appropriate value of α depends on the characteristics of the graph. In our graphs, we

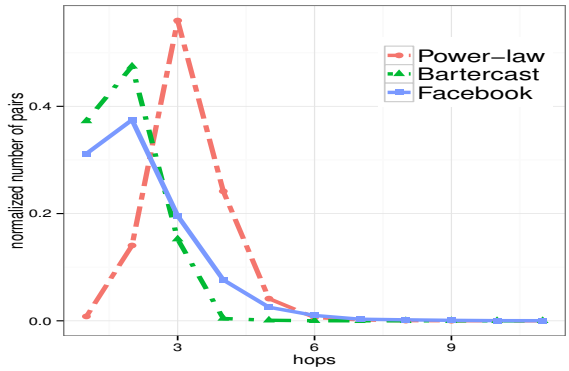


Figure 5.7: The distance (in hops) between two nodes just before they interact

observe that the vast majority of nodes interact with other nodes that are only a few hops away. In Figure 5.7, we present the probability of interaction between two nodes as a function of their distance just before they interact. Our graphs exhibit a high locality of interaction, which implies that a node does not need to perform long walks in order to acquire relevant information. Particularly in real-world graphs, we observe that more than 90% of pairs of interacting nodes have a distance of at most 3 hops just before they interact. Therefore, for these graphs we use random walks with an expected length l of 3 hops. In synthetic graphs, we use random walks of an expected length of 4 hops, since for power-law graphs the majority of pairs of interacting nodes have a distance of at most 4 hops just before they interact.The restart parameter is computed as a function of the desired expected length as $\alpha = 1/(l+1)$ [12]. So, for real-world graphs we use $\alpha = 0.25$ and for synthetic graphs, $\alpha = 0.2$.

5.4.3 Experiment Setup

We integrated the proposed random walker in Tribler and we can evaluate it on the DAS-4 supercomputer [1] available at Delft University of Technology. Particularly, we run 1000 clients distributed evenly on 20 nodes of DAS-4. In Tribler, the dissemination of information is based on epidemics [43]. We modified only its dissemination component integrating the proposed random walker.

Each Tribler client has its own local database keeping its own locally stored history of interactions and it interacts with other clients following the interactions in the previously described datasets. At the same time it performs its own random walks towards other nodes. We divide our datasets into two parts: a small part used for initialization and the main part used for emulation. In Facebook, this initialization part consists of the interaction occurred during the first week in the corresponding dataset, in Bartercast during the first day, and in power-law graphs during the first 1000 steps.

After the end of the initialization process, nodes emulate the interactions, compute the reputations of nodes and walk towards each other collecting information. The time of emulated interactions has been mapped to the duration of our emulation. A node emulates an interaction with another node, by creating a record with the details of this interaction and it sends it to the other interacting node. Then, both nodes store this record in their database and they include it in the history of interactions they distribute. All nodes perform walk steps with the same period about every 30 secs so that all nodes perform a similar number of random walks during the experiment. Each experiment lasts 2 hours.

5.4.4 Sybil Attack Model

We evaluate EscapeLimit against sybil attacks assuming that the honest nodes interact frequently with each other forming a well-connected community. On the contrary, interactions between honest and sybil nodes are limited due to the high social engineering cost required [117]. As a consequence, the honest nodes form a region that is well separated from the sybil region. The sybil nodes can be connected with each other in an arbitrary way. The two regions are connected by *the attack edges* that link nodes in the sybil region to victim nodes in the honest region. This sybil attack model has been used by most of the schemes proposed in the literature [125], [126] and it has been verified in real social networks [25]. Some other studies on social networks have shown that honest and sybil nodes are well connected [123]. However, those studies examined only the graph deriving from the social connections among nodes and not their interactions.

For our evaluation, we allow the graphs having different community structures without any assumptions. Most of the random walk-based methods proposed in literature against sybil attacks [25], [39] assume fast mixing graphs, since they have tight trust relationships between the nodes. On the contrary, in a slow-mixing graph multiple communities exist and so, sybil nodes might be incorrectly recognised as honest. We use both fast-mixing graphs such as Facebook where honest nodes form one well-connected community, and slow-mixing graphs such as Bartercast, where more than one community is present.

In our experiments, we take as the honest region our initial datasets and we create a power-law graph of 100 nodes as a sybil region, since it has been shown in a recent study [25] that the sybil region has a skewed degree distribution. Then, we randomly choose some sybil nodes and some victim nodes and connect them through the corresponding attack edges. To each attack edge, we assign probabilistically a weight in the range of the weights of edges among honest nodes so that, attack edges with small weights are more common, since it is more costly for an attacker to create an attack edge with a large weight than an attack edge of a low weight. For the timestamp of the attack edges, we assume that they are uniformly distributed over time. The nodes in the sybil region can claim any values for the properties of their edges.

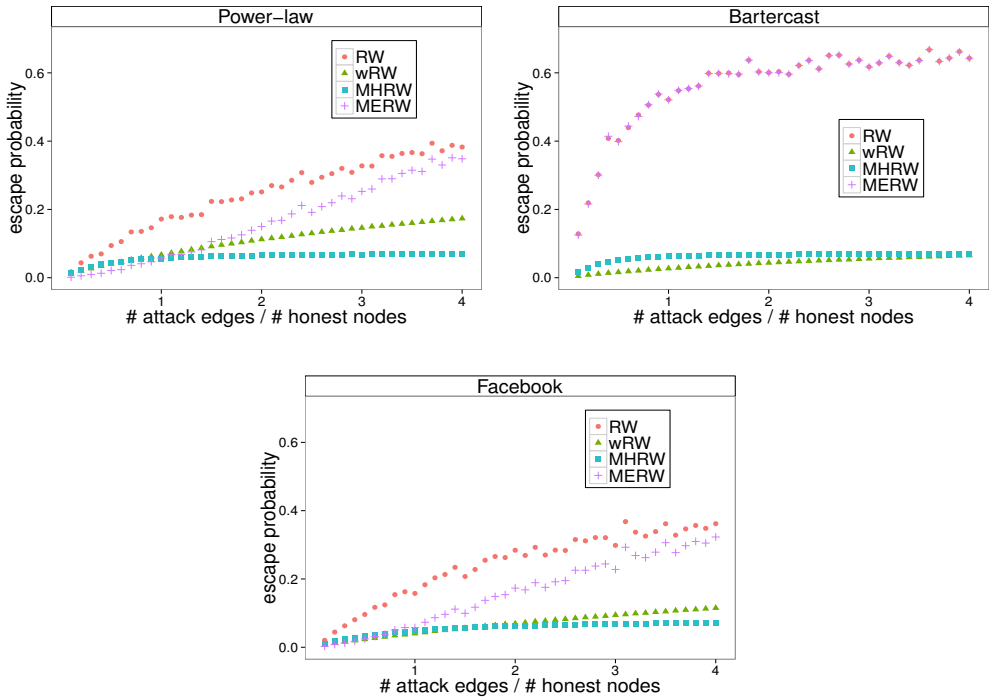


Figure 5.8: Resilience against sybil attacks: the escape probability of the different random walks in the power-law, Bartercast, and Facebook graphs.

5.5 Evaluation

In this section, we present the results of the evaluation of EscapeLimit with the different biased random walks in terms of its resilience to attacks, its scalability and its ability to acquire relevant information fast. We use a set of metrics associated with our requirements. All the presented results are the average of 10 experiments.

5.5.1 Resilience against Sybil Attacks

A random walker escaping into the sybil area will be trapped there till it restarts. Therefore, we evaluate the fraction of walks escaping into the sybil region when starting at any node in the honest region, which is called the *escape probability* [125]. Lower values of escape probability indicate higher resilience against sybil attacks. This probability depends on the number of attack edges, since in order to escape to the sybil region, the random walk has to traverse an attack edge.

In Figure 5.8 we present the escape probability of the different random walks depending on the average number of attack edges per honest node for the different datasets. Independently of the characteristics of the graphs, all biased RWs exhibit a smaller escape

Table 5.3: Scalability: the correlation between the indegree and the distribution of the communication load of the walks

	power-law	Bartercast	Facebook
RW	0.939	0.88	0.77
bRW	0.94	0.75	0.78
MHRW	−0.82	−0.6	−0.51
MERW	0.95	0.89	0.817

probability into the sybil region than simple RW, indicating that the strength of interactions, and the activity level of nodes are accurate indicators of trust. The fast-mixing graphs, power-law and Facebook, have smaller escape probability when using RWs and MERW than Bartercast. Being fast-mixing, those graphs have a tightly connected honest region and a random walk does not escape into the sybil region with high probability. In power-law graphs, wRW exhibits the highest escape probability in comparison with the other graphs since in power-law graphs the range of the weights is smaller.

Nevertheless, independently of the characteristics of the graph, wRW and MHRW exhibit the lowest escape probability, which increases very slowly with the increase of the number of attack edges per honest node. Highly weighted attack edges are more rare due to high engineering cost required for their creation and as a result, wRW traverses with low probability the attack edges. Hence, it exhibits low escape probability for all the examined graphs. Furthermore, MHRW tends to visit low degree nodes at the periphery of the network and so, it rarely escapes into the sybil region. On the contrary, MERW visits more often high degree nodes, and as a result it has a similar escape probability to RW.

5.5.2 Scalability

In Section 5.1, we define scalability in terms of three dimensions: fault-tolerance, system load, and administration. By construction, our proposed random walker is able to handle node failures. Furthermore, it has low computational cost since at each step only the transition probabilities are computed and each node maintains connectivity information only about its neighbors. Therefore, it is scalable in terms of the fault-tolerance and the administration requirements and so, we have to evaluate its system load in terms of its communication overhead. In EscapeLimit, each node sends one introduction request to another node in the system. Thus, the communication overhead at each node during a random walk depends on the visit ratio of that node at each step, namely the fraction of introduction requests it receives from other nodes at each walk step. To evaluate this overhead at each node, we define the *distribution of communication load* in terms of the

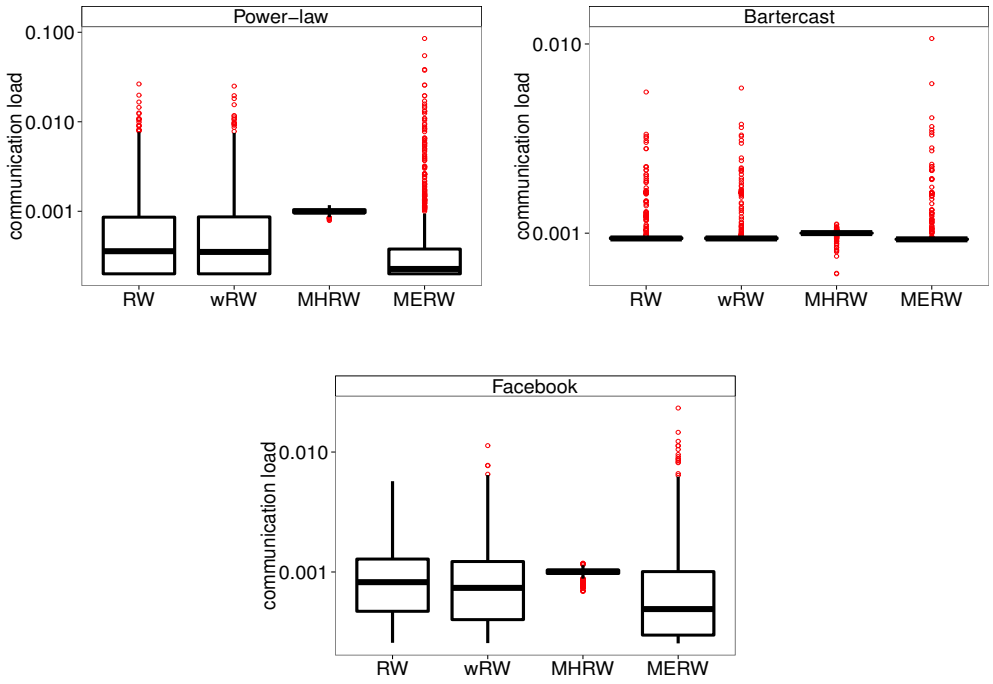


Figure 5.9: Scalability: the distribution of the communication load of nodes for the different random walks in the power-law, Bartercast, and Facebook graphs.

average visit ratios of the nodes per step. This ratio is proportional to indegree of a node, as is presented in Table 5.3. Due to the highly skewed indegree distributions of real-world networks, a few highly connected nodes receive the majority of introduction requests. As a results, those highly connected nodes may be overloaded.

In Figure 5.9, we present the distribution of communication load of nodes in the power-law, Bartercast and Facebook graphs when we integrate different biases in the random-walk based collection method. Since we ran the experiment for 1000 nodes, the optimal communication load value for a node is 0.001, meaning that it has been visited exactly once during a walk step. In this experiment, we do not include sybil nodes, since sybils will have no impact on the communication load of honest nodes.

For all the examined graphs, MHRW distributes the load evenly to almost all the nodes independently of the indegree distribution of the corresponding graph. In a random walk, the high degree nodes are visited more often, but this property is counterbalanced by the bias of MHRW towards the low-degree nodes and so, MHRW achieves an almost uniform load distribution. Conversely, MERW intensifies the selection of the highly connected nodes and as a result, in all graphs it exhibits the most skewed distribution of communication load. Particularly in the power-law graph, a few highly connected nodes have a communication load value close to 0.100 implying that those nodes receive 100 introduc-

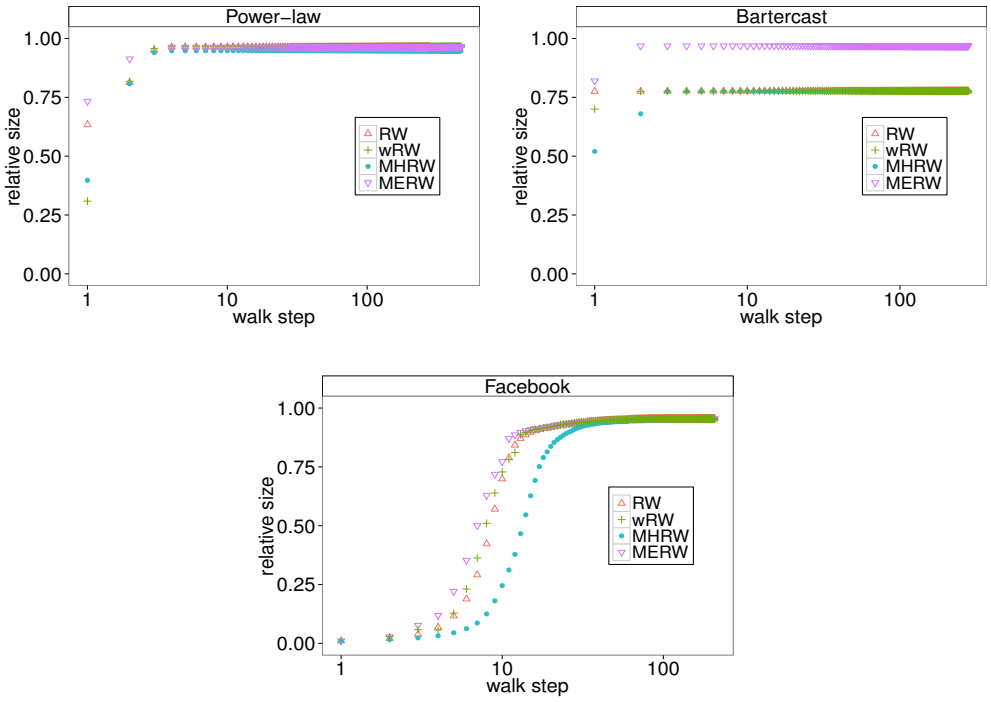


Figure 5.10: Relevance: the average relative size between the interaction subgraphs of nodes and the interaction graphs of the power-law, Bartercast, and Facebook networks over consecutive steps of the different random walks.

tion requests during a walk step. Those nodes cannot reply to all those request and as a result, MERW is not scalable in that graph. Furthermore, wRW has a load distribution similar to RW.

5.5.3 Relevance of Information

In order to capture different characteristics of the relevance of the acquired information at each step of the walk, we use two metrics. The first metric is the *relative size* of the interaction subgraph G_i at node i with respect to the size of G and it is defined as $RE(G_i, G) = |E_i|/|E|$ [100].

According to the second metric, the *ranking similarity* (RS), the interaction subgraph G_i at node i is similar to G if it produces similar reputation rankings of the most highly reputed nodes. Ranking similarity is a modification of Spearman coefficient and the vertex ranking metric proposed in [93], so that it can be applied to lists of different lengths and takes into account the reputation of each node. If we denote by r (and r_i) the reputation

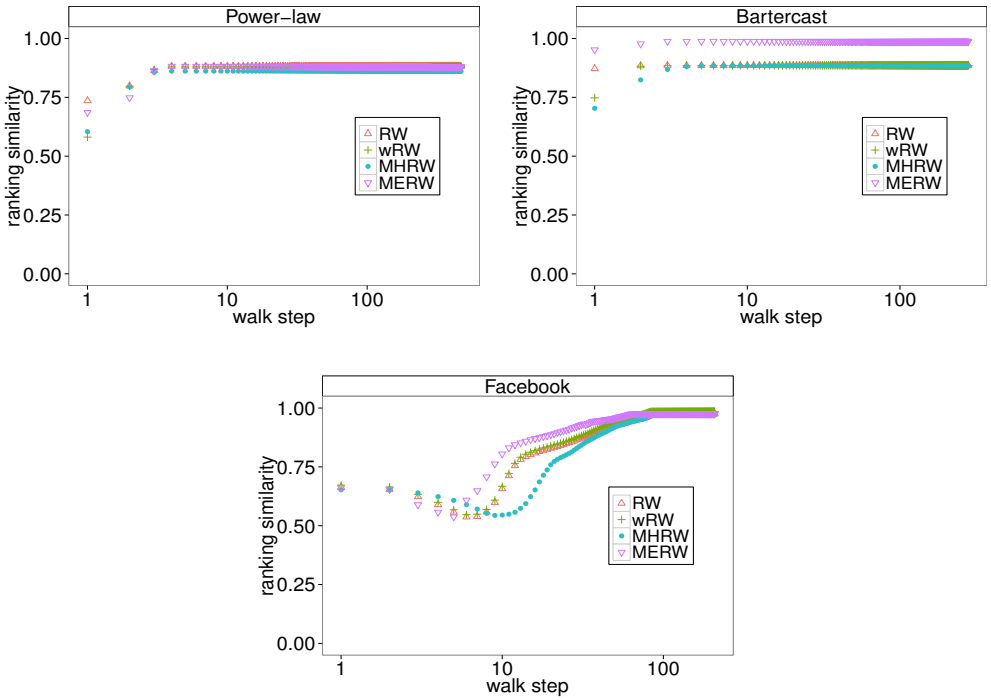


Figure 5.11: Relevance: the average ranking similarity between the interaction subgraphs of nodes and the interaction graphs of the power-law, Bartercast, and Facebook networks over consecutive step of the different random walks.

vector produced at G (and G_i), then the ranking similarity is defined as:

$$RS(G, G_i) = 1 - \frac{\sum_{u \in V_i} r(u)(\sigma(r(u)) - \sigma(r_i(u)))^2}{D},$$

where $\sigma(r(u))$ (and $\sigma(r_i(u))$) is the rank of the reputation of node u in $r(u)$ (and $r_i(u)$) when only vertices in V_i are considered and r (and r_i) is ordered in decreasing order. The normalization factor D is equal to $\sum_{u \in V_i} r(u)(\sigma(r(u)) - \sigma(r_w(u)))^2$ where r_w is the sequence containing the nodes in V_i in reverse order from r. The ranking similarity between the two graphs is equal to 1 if their reputation vectors produce exactly the same ranking. On the contrary, a ranking similarity equal to 0 indicates that the ranking derived from r_i is the reverse of the ranking deriving by r.

In Figure 5.10, we present the relevant size between the interaction graph and the interaction subgraphs over consecutive steps of the different random walks for all the datasets. For all the different graphs, MERW achieves the largest relative size faster. According to MERW, each nodes visits with higher probability the highest degree nodes. These nodes have the majority of information since they perform the majority of interactions. On the contrary, MHRW visits mostly low degree nodes which perform very few interactions and

Table 5.4: The ability of random walks to satisfy the requirements of a collection mechanism applicable to distributed reputation systems

Method	Resilience to Attacks	Scalability	Relevance of information
RW	fair	good	good
wRW	very good	good	good
MHRW	very good	very good	poor
MERW	good	poor	very good

so, results in very small relative size.

In fast-mixing graphs after some steps all the random walks achieve almost perfect relative size. On the contrary, in Bartercast only MERW achieves perfect relative size. As we can see from its indegree distribution in Figure 5.5, Bartercast has 3 hubs and most nodes are not connected with each other but only with the hubs. As a result, all the RWs but MERW do not manage to visit all the hubs and visit other nodes in the periphery since their expected length is small. In graphs with skewed degree distribution, such as power-law graph and Bartercast, all the different random walks achieve most of the information after a small number of walks steps since those hubs are visited with higher probability. Piatek et al [96] based their dissemination scheme on a similar observation. However, in graphs with a more symmetric degree distribution, such as Facebook, the collection of information is much slower. Particularly, MHRW on Facebook needs more than 40 steps to achieve most of the information.

In Figure 5.11, we present the ranking similarity between the interaction graph and the interaction subgraphs over consecutive steps of the different random walks for all the datasets. In accordance with the results for relative size, MERW achieves faster a high ranking similarity while MHRW is the slowest. In power-law and Bartercast graphs, ranking similarity follows the patterns of relative size due to the skewed degree distribution. In these graphs, the hubs not only have a high degree but a high reputation as well. In Facebook, there is an instability in the ranking similarity in the first steps of random walks. The Facebook graph has a large clustering coefficient and it does not have outliers like the other two graphs. As a result, it achieves a high ranking similarity slower. From Figure 5.10, we observe that during those steps nodes collect about 75% of the information. Afterwards, the ranking similarity increases.

5.5.4 Discussion

Our evaluation indicates that properly biased random walks satisfy the requirements of an applicable collection method since they achieve resilience to sybil attacks, good load balancing and provides relevant information. The bias of a random walk determines the

extent to which each requirement is satisfied. In Table 5.4, we summarize the experimental results for all the different random walks.

In fast-mixing networks, simple RW achieves good resilience against attacks while in slow-mixing networks it escapes with high probability into the sybil region even if the number of attack edges per honest node is small. Furthermore, it distributes the communication load across the nodes of a network with a preference towards nodes with high degree. RW is suitable only for fast-mixing networks. Adding the appropriate biases by using richer information about the interactions of nodes further improves its quality.

Through our experimental results, we have shown that wRW achieves robustness against sybil attacks and collects relevant information independently of the characteristics of the graph. Therefore, the strength of interactions represent accurately both the trust and the similarity of nodes. Furthermore, the communication load at each node when using wRW is close to RW. This type of walk is suitable for all networks and particularly for networks where the strength of edges has a skewed distribution.

Next, we studied the bias towards the nodes with low activity, namely the nodes with low degree. In MHRW, the bias is towards the low degree nodes. This walk achieves high resilience to attacks since sharing interactions with low degree nodes is a stronger indication of trust than interactions with high degree nodes. Furthermore, MHRW has excellent load balancing properties independently of the degree distribution of the network. However, a node visiting the low degree nodes cannot obtain fast relevant information. This type of walk can be used when for a network the main concerns are security and load balancing.

Besides the strength of interactions, the activity level of a node as represented by its degree is another indicator of trust and similarity among nodes. In MERW, the resilience against sybil attacks is similar to that of RW indicating that sharing interactions with high degree nodes is not a strong indication of trust. Furthermore, the bias towards the high degree nodes results in overloading those nodes. Nevertheless, MERW achieves fast relevant information since it visits more often the hubs even if the network is slow-mixing. Through the hubs, MERW manages to visit the different communities in a slow-mixing network. This type of walk is suitable for slow-mixing graphs when the fast acquisition of relevant information is important.

5.6 Conclusion

In this chapter, we propose a method to collect information in distributed reputation systems. EscapeLimit collects only relevant and trusted information by constraining relative size. information in distributed reputation systems based on random walks. EscapeLimit collects only relevant and trusted information as well as reduces the escape probability of an honest node to the Sybil area. EscapeLimit uses the observation that user interac-

tions require real effort and so, they reflect trust and similarity among users. Our design is suitable for internet deployed networks with high dynamics. We guide random walks in EscapeLimit with three different trust-driven user properties and through experimental evaluation we show their effectiveness in terms of resilience to attacks, scalability and the ability to provide each user with relevant information. Our evaluation suggest that the strength of user interactions guides random walks efficiently in almost any type of network.

Chapter 6

Conclusion

In this thesis, we studied decentralized reputation systems in order to establish trust among strangers in online communities. Their large number of participants and their vulnerability to attacks challenge the identification of relevant and trustworthy information. We represent node interactions as a growing graph with nodes representing the users and edges the interactions between two users, called the interaction graph. This interaction graph could capture the individual characteristics of each user as well as the collective user behavior. By analyzing interaction graphs derived from real-world and synthetic networks, we gained key insights about the interaction patterns of nodes and we designed more efficient algorithms for each of the three components of decentralized reputation systems: the computation of reputations, and the storage of the history of user interactions, and its collection. Below we present our conclusions and suggestions for future work.

6.1 Conclusions

Our conclusion answer the research questions presented in the introduction of this thesis and are presented below:

1. Max flow-based computation of reputations is more accurate when using the node with the highest Betweenness Centrality (BC) in the computation of reputations. However, BC is expensive to be computed. We experimentally evaluated two different approximate approaches for computing BC, exploring both theoretical graph models (random and scale-free) and the graph derived from the actual operation of the Bartercast reputation system. For growing networks, our first approach relies on the observation that the nodes with high BC in real-world networks remain almost invariant over time. For large networks, our second approach consists in assessing three approximation methods in terms of their ability to identify the top-most central nodes. We conclude that in scale-free graphs, the BC approximations are efficient and highly accurate, while in random graphs, due to their structural properties, it

is harder to identify the most central nodes. In the graph derived from Bartercast, these approximations exhibit similar performance and are adequately accurate. Finally, we have integrated the approximation methods for BC into the computation of reputations in Bartercast, and we found the accuracy and the coverage of the computed reputations for two of these methods to be excellent.

2. Random walk-based computations of reputations are more robust against uncooperative nodes and Sybil attacks when they are biased with the properties of nodes. We showed that the robustness of a biased random walk agains uncooperative nodes depends on the characteristics of the graph. In graphs with both specific construction patterns and nodes with heterogenous properties, such as the scale-free, Citation and Facebook graphs, predicting the behavior of nodes is very accurate. In graphs of nodes with uniform properties and highly dynamic behavior, biased random walks predict less accurately the behavior of nodes but still much better than simple random walks. Furthermore, the appropriate node properties to bias random walks against Sybils depend on the characteristics of the graph. In graphs with large clustering coefficient, such as our scale-free, Facebook and Citation graphs, random walks biased with node similarities are very effective. In graphs with edges with heterogenous strengths, such as Bartercast and Facebook, biasing random walks with the strength of an edge is very effective while using temporal properties is effective in graphs with strong temporal patterns.

3. For storing information in a decentralized reputation system, using the complete history of interactions in a reputation system is costly. Having observed that not all the stored information contributes equally to the computation of reputations, we have proposed the use of the reduced history of interactions instead of the complete history and we defined the main parameters for choosing the information participating in it. Next, we have evaluated our approach experimentally exploring both theoretical graph models and real-world graphs using two reputation algorithms, a max-flow based algorithm and Pagerank. We conclude that for scale-free and real-world graphs, the reduced history is reasonably accurate, while for random graphs, due to their structural properties, the reduced history causes lower accuracy. Furthermore, we have demonstrated that using the max-flow based algorithm results in better identification of the highest ranked nodes while using Pagerank results in better ranking similarity.

4. We proposed a method for the collection of information for distributed reputation systems using random walks. Our method is based on the observation that user interactions require real effort and so, they reflect trust and relevance among users. Our design is suitable for internet deployed networks. We guided random walks

with three different trust-driven user properties and through experimental evaluation we show their effectiveness in terms of their resilience to attacks, their scalability and their ability to provide each user with relevant information. Our evaluation suggests that the strength of user interactions guides random walks efficiently in almost any type of network.

6.2 Suggestions for Future Work

Below we present directions for future research based on the insights derived from this thesis:

1. As betweenness centrality is too expensive to be computed in large growing graphs, we used approximations for identifying the most central nodes in Chapter 2. The approximations are efficient only for graphs with skewed degree distributions. Incremental algorithms for betweenness centrality may give new opportunities for centrality computations in online systems. Their design is not trivial because of unrealistic storage costs for intermediate results. However, incremental betweenness centrality algorithms might be efficient for graphs with particular topologies.

2. In Chapter 3, we proposed biased random walks assuming that all nodes use a globally fixed time window and teleportation parameter. As future work, it will be useful to design an adaptive algorithm in which each node defines its own values for the time window and teleportation parameter based on its perception of the graph. Those parameters will dynamically adapt to changes of the graph.

3. Reducing the history of interactions in decentralized reputation systems is very challenging as indicated by the high values of error in our results in Chapter 4. Our proposed algorithm leverages global properties of nodes and edges in order to reduce the history of interactions. Personalized graph-based algorithms may further improve the accuracy of the reduced history. Evaluating those algorithms using graph-based similarities will result in new insights into the usage of reduced history.

4. In Chapter 5, we proposed a random walk-based method for collecting the history of interactions. Our approach is suitable for online networks but we did not evaluate its performance under extreme churn. Assessing the impact of extreme churn in a random walk-based collection of information is very useful for networks with high dynamics such as P2P systems. Furthermore, we biased the random walks with three different trust-driven user properties resulting in different performance in terms of resilience to attacks, scalability and ability to provide each user with relevant information. The design of an adaptive algorithm that is able to switch

between these three biases depending on the demands of each user, could be very useful, as user behavior changes over time.

5. While in this thesis we assumed that users only participate in one online community, nowadays users participate in many online communities at the same time. A line of research may explore the propagation of reputations across multiple online communities. For example, researchers could study which communities have similar user reputations. In this context, we could answer if the reputation of a user in Bartercast reveals his behavior in Facebook or if the interaction subgraph of a user in Facebook is useful for computing reputations in Youtube.

Bibliography

[1] Das-4. http://www.cs.vu.nl/das4/, 2014.

[2] Diaspora. `https://diasporafoundation.org`, 2014.

[3] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 2003.

[4] B. Thomas Adler and Luca de Alfaro. A content-driven reputation system for the wikipedia. In *WWW*, 2007.

[5] R. Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 2005.

[6] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *PNAS*, 2000.

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

[8] Robert Axelrod and Richard Dawkins. *The Evolution of Cooperation: Revised Edition*. Basic Books, 2006.

[9] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. In *WSDM*, 2011.

[10] D. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *Algorithms and Models for the Web-Graph*, 2007.

[11] D. A. Bader and K. Madduri. Parallel algorithms for evaluating centrality indices in real-world networks. In *International Conference on Parallel Processing*, 2006.

[12] Coralio Ballester and Marc Vorsatz. Random walk based segregation measures. *Review of Economics and Statistics*, 2011.

[13] Ziv Bar-Yossef and Li-Tal Mashiach. Local approximation of pagerank and reverse pagerank. In *ACM SIGIR*, 2008.

[14] A. L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 1999.

[15] M. Barthélemy. Betweenness centrality in large complex networks. *Eur. Phys. Jour. B*, 2004.

[16] Yochai Benkler. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press, 2006.

[17] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[18] S. Borgatti, K. Carley, and D. Krackhardt. On the robustness of centrality measures under conditions of imperfect data. *Social Networks*, 2006.

[19] S.P. Borgatti and M.G. Everett. A graph-theoretic framework for classifying centrality measures. *J Social Networks*, 2006.

[20] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 2001.

[21] U. Brandes and C. Pich. Centrality estimation in large networks. In *Intl. Journal of Bifurcation and Chaos, Special Issue on Complex Networksí Structure and Dynamics*, 2007.

[22] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: Early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, 2009.

[23] Z. Burda, J. Duda, J. M. Luck, and B. Waclaw. Localization of the Maximal Entropy Random Walk. *Physical Review Letters*, 2009.

[24] Z. Burda, J. Duda, J. M. Luck, and B. Waclaw. The various facets of random walk entropy. *Acta Phys. Polon. B*, 2010.

[25] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI*, 2012.

[26] Mihai Capotã, Nazareno Andrade, Tamás Vinkó, Flavio R Santos, Johan Pouwelse, and Dick Epema. Inter-swarm resource allocation in bittorrent communities. In *11-th IEEE International Conference on Peer-to-Peer Computing (P2P11)*, 2011.

[27] Rajiv Chakravorty, Sulabh Agarwal, and Suman Banerjee. Mob: A mobile bazaar for wide-area wireless services. In *ACM MobiCom*, 2005.

[28] S. Y. Chan, I. X. Y. Leung, and P. Lio. Fast centrality approximation in modular networks. In *Proceedings of the 1st ACM International Workshop on Complex Networks in Information and Knowledge Management (CNIKM)*, 2009.

[29] Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. The bloomier filter: An efficient data structure for static support lookup tables. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, 2004.

[30] Alice Cheng and Eric Friedman. Manipulability of pagerank under sybil strategies. In *NetEcon*, 2006.

[31] Nitin Chiluka. *Securing Social Media: A Network Structure Approach*. PhD thesis, Delft University of Technology, 2013.

[32] Nitin Chiluka, Nazareno Andrade, Dimitra Gkorou, and Johan A. Pouwelse. Personalizing eigentrust in the face of communities and centrality attack. In *AINA*, 2012.

[33] Nicholas A. Christakis and James 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*. Back Bay Books, 2011.

[34] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Reviews*, 2007.

[35] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[36] E. Costenbader and T. Valente. The stability of centrality measures when networks are sampled. *Social Networks*, 2003.

[37] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook : a privacy preserving misc social network leveraging on real-life trust. *IEEE Communications Magazine, Consumer Communications and Networking Series*, 2009.

[38] Lucia D'Acunto, Nitin Chiluka, TamáS Vinkó, and Henk Sips. Bittorrent-like p2p approaches for vod: A comparative study. *Computer Networks*, 2013.

[39] George Danezis and Prateek Mittal. SybilInfer: Detecting Sybil Nodes using Social Networks. In *NDSS*, 2009.

[40] L. De Alfaro, A. Kulshreshtha, I. Pye, and B. T. Adler. Reputation systems for open collaboration. *Commun. ACM*, 2011.

[41] R. Delaviz, N. Andrade, and J. A. Pouwelse. Improving accuracy and coverage in an internet-deployed reputation mechanism. In *Peer-to-Peer Computing*, 2010.

[42] R. Delaviz, J.A. Pouwelse, and D.H.J. Epema. Targeted and scalable information dissemination in a distributed reputation mechanism. In *Proceedings of the seventh ACM Workshop on Scalable Trusted Computing (ACM STC)*, 2012.

[43] Rahim Delaviz, Niels Zeilmaker, Johan Pouwelse, and Dick Epema. A network science perspective of a distributed reputation mechanism. In *IFIP Networking*, 2013.

[44] Chrysanthos Dellarocas. The digitization of word of mouth: Promise and challenges of misc feedback mechanisms. *Manage. Sci.*, 2003.

[45] John R. Douceur. The Sybil attack. In *IPTPS*, 2002.

[46] R. I. M. Dunbar. Neocortex size as a constraint on group size in primates. *Journal of Human Evolution*, 1992.

[47] P. Erdős and A. Rényi. On random graphs i. *Publicationes Mathematicae (Debrecen)*, 6, 1959.

[48] Martin Everett and Stephen P Borgatti. Ego network betweenness. *Social networks*, 2005.

[49] Facebook. Facebookb form s-1 @. `http://www.sec.gov/Archives/edgar/data/1326801/000119312512325997/d371464d10q.htm#tx371464_14`, 2012.

[50] Facebook: Key facts. Facebook statistics. `http://newsroom.fb.com/Key-facts`, 2014.

[51] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, 1999.

[52] Pascal Felber, Anne-Marie Kermarrec, Lorenzo Leonini, Étienne Rivière, and Spyros Voulagris. PULP: an Adaptive Gossip-Based Dissemination Protocol for Multi-Source Message Streams. *Peer-to-Peer Networking and Applications, Springer*, 2011.

[53] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *ACM EC*, 2004.

[54] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL: a modeling language for mathematical programming*. Duxbury Press, 2002.

[55] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 1977.

[56] Francis Fukuyama. *Trust: The Social Virtues and the Creation of Prosperity*. The Free Press, 1996.

[57] Diego Gambetta. *Trust: Making and Breaking Cooperative Relations*. Blackwell Pub, 1990.

[58] R. Geisberger, P. Sanders, and D. Schultes. Better approximation of betweenness centrality. In *ALENEX*, 2008.

[59] G. Ghoshal and A.-L. Barabási. Ranking stability and super-stable nodes in complex networks. *Nature Communications*, 2011.

[60] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 2002.

[61] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks: algorithms and evaluation. *Performance Evaluation*, 2006.

[62] Christos Gkantsidis and Milena Mihail. Hybrid search schemes for unstructured peer-to-peer networks. In *IEEE INFOCOM*, 2005.

[63] Dimitra Gkorou, Johan Pouwelse, and Dick Epema. Betweenness centrality approximations for an internet deployed p2p reputation system. In *IEEE IPDPSW (HotP2P)*, 2011.

[64] Dimitra Gkorou, Tamás Vinkó, Nitin Chiluka, Johan Pouwelse, and Dick Epema. Reducing the history in decentralized interaction-based reputation systems. In *IFIP Networking*, 2012.

[65] Dimitra Gkorou, Tamás Vinkó, Johan Pouwelse, and Dick Epema. Leveraging node properties in random walks for robust reputations in decentralized networks. In *EEE P2P Computing*.

[66] Glauber D. Gonçalves, Anna Guimarães, Alex Borges Vieira, Ítalo S. Cunha, and Jussara M. Almeida. Using centrality metrics to predict peer cooperation in live streaming applications. In *IFIP Networking*, 2012.

[67] Dominik Grolimund, Luzius Meisser, Stefan Schmid, and Roger Wattenhofer. Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems. In *1st Workshop on the Economics of Networked Systems (NetEcon)*, 2006.

[68] Z. Gyongyi and H. Garcia-Molina. Web spam taxonomy. In *AIRWeb*, 2005.

[69] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *VLDB*, 2004.

[70] K. Hajra and P. Sen. Aging in citation networks. *Physica A: Statistical Mechanics and its Applications*, 2005.

[71] Gertjan Halkes and Johan Pouwelse. Udp nat and firewall puncturing in the wild. In *IFIP Conference on Networking*, Berlin, Heidelberg, 2011.

[72] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 1970.

[73] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys*, 2009.

[74] John Hopcroft and Daniel Sheldon. Manipulation-resistant reputations using hitting time. In *Conference on Algorithms and models for the web-graph*, 2007.

[75] Gareth James, Daniela Wittenand Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning. Springer. p. 6.* Springer, 2013.

[76] H. Jeong, S. P. Mason, A. L. Barabási, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 2001.

[77] Adele Lu Jia, Boudewijn Schoon, Johan Pouwelse, and Dick Epema. Estimating user interaction strength in online networks. Technical Report PDS-2013-007, TU Delft, 2013.

[78] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW*, 2003.

[79] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. FOCS, 2003.

[80] Nicolas Kourtellis, Tharaka Alahakoon, Ramanuja Simha, Adriana Iamnitchi, and Rahul Tripathi. Identifying high betweenness centrality nodes in large social networks. *Social Network Analysis and Mining (SNAM)*, 2013.

[81] Rong-Hua Li, Jeffrey Xu Yu, and Jianquan Liu. Link prediction: the power of maximal entropy random walk. In *CIKM*, 2011.

[82] Thomas Locher, Patrick Moor, Stefan Schmid, and Roger Wattenhofer. Free riding in bittorrent is cheap. In *In HotNets*, 2006.

[83] K. Madduri, D. Ediger, K. Jiang, D. A. Bader, and D. Chavarria-Miranda. A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. In *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*, 2009.

[84] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 2006.

[85] Travis Martin, Brian Ball, Brian Karrer, and M. E. J. Newman. Coauthorship and citation patterns in the physical review. *Phys. Rev. E*, 2013.

[86] Liam McNamara, Cecilia Mascolo, and Licia Capra. Media sharing based on colocation prediction in urban transport. MobiCom, 2008.

[87] M. Meulpolder, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips. Bartercast: A practical approach to prevent lazy freeriding in p2p networks. In *IEEE IPDPS (HotP2P)*, May 2009.

[88] Barbara Misztal. *Trust in Modern Societies: The Search for the Bases of Social Order*. Polity, 1996.

[89] A. Mohaisen, N. Hopper, and Y. Kim. Keep your friends close: Incorporating trust into social network-based sybil defenses. In *INFOCOM*, 2011.

[90] B. Clifford Neuman. Scale in distributed systems. In *Readings in Distributed Computing Systems*, 1994.

[91] M. E. J Newman. Scientific collaboration networks. i. network construction and fundamental results. *Phys. Rev. E*, 2001.

[92] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

[93] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. Web graph similarity for anomaly detection. *J. Internet Services and Applications*, 2010.

[94] Eric Paulos and Elizabeth Goodman. The familiar stranger: Anxiety, comfort, and play in public places. CHI, 2004.

[95] Riccardo Petrocco, Johan Pouwelse, and Dick Epema. Performance analysis of the libswift p2p streaming protocol. In *IEEE Conference on Peer-to-Peer Computing (P2P12)*, 2012.

[96] Michael Piatek, Tomas Isdal, Arvind Krishnamurthy, and Thomas Anderson. One hop reputations for peer to peer file sharing workloads. In *NSDI*, 2008.

[97] Michael Piatek, Tomas Isdal, Arvind Krishnamurthy, and Thomas Anderson. One hop reputations for peer to peer file sharing workloads. In *NSDI*, 2008.

[98] Ansley Post, Vijit Shah, and Alan Mislove. Bazaar: Strengthening user reputations in online marketplaces. In *NSDI*, 2011.

[99] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips. Tribler: a social-based peer-to-peer system. *Concurr. Comput.: Pract. Exper.*, 2008.

[100] David M. W. Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness and correlation. Technical Report SIE-07-001, School of Informatics and Engineering, Flinders University, 2007.

[101] Daniele Quercia. *Trust Models for Mobile Content-Sharing Applications*. PhD thesis, University College London, 2009.

[102] Daniele Quercia and Stephen Hailes. Sybil attacks against mobile users: friends and foes to the rescue. In *INFOCOM*, 2010.

[103] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Commun. ACM*, 2000.

[104] K. Ronasi, M.H. Firooz, M.-R. Pakravan, and A. Nasiri Avanaki. An enhanced random-walk method for content locating in p2p networks. In *ICDCSW*, 2007.

[105] Matthew A. Russell. *Mining the Social Web*. O'Reilly Media; Second Edition edition, 2013.

[106] Bruce Schneier. *Liars and Outliers: Enabling the Trust that Society Needs to Thrive*. Wiley, 2012.

[107] Roberta Sinatra, Jesús Gómez-Gardeñes, Renaud Lambiotte, Vincenzo Nicosia, and Vito Latora. Maximal-entropy random walks in complex networks with limited information. In *Phys. Rev. E*, 2011.

[108] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Symposium on Security and Privacy*, 2010.

[109] Gayatri Swamynathan, Kevin C. Almeroth, and Ben Y. Zhao. The design of a reliable reputation system. *Journal Electronic Commerce Research*, 2010.

[110] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *ICDM*, 2006.

[111] Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian. Sybil-resilient online content voting. In *NSDI*, 2009.

[112] Inc. Twitter. Twitter form s-1. `http://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htm`, 2013.

[113] Inc. Twitter. Twitter statistics. `http://www.statisticbrain.com/twitter-statistics/`, 2014.

[114] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. A survey of dht security techniques. *ACM Computing Surveys*, 2011.

[115] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *ACM WOSN*, 2009.

[116] Bimal Viswanath, Mainack Mondal, Allen Clement, Peter Druschel, P. Krishna Gummadi, Alan Mislove, and Ansley Post. In *COMSNETS*, 2012.

[117] Bimal Viswanath, Ansley Post, Krishna P. Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. In *SIGCOMM*, 2010.

[118] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 2006.

[119] Kevin Walsh and Emin Gün Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *NSDI*, 2006.

[120] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, and Ben Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European Conference on Computer Systems*, EuroSys, 2009.

[121] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE TKDE*, 2004.

[122] Weilian Xue, Yaqiong Liu, Keqiu Li, Zhongxian Chi, Geyong Min, and Wenyu Qu. Dhtrust: A robust and distributed reputation system for trusted peer-to-peer networks. *Concurr. Comput. : Pract. Exper.*, 2012.

[123] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y. Zhao, and Yafei Dai. Uncovering social network sybils in the wild. *IMC*, 2011.

[124] Youtube. Youtube statistics. `https://www.youtube.com/yt/press/en-GB/statistics.html`, 2014.

[125] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, 2008.

[126] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. In *SIGCOMM*, 2006.

[127] Nicolaas Zeilemaker, Zekeriya Erkin, Paolo Palmieri, and Johan Pouwelse. Building a privacy-preserving semantic overlay for peer-to-peer networks. In *Information Forensics and Security (WIFS), 2013 IEEE International Workshop on*, 2013.

[128] Niels Zeilemaker, Boudewijn Schoon, and Johan Pouwelse. Dispersy bundle synchronization. Technical Report PDS-2013-002, TU Delft, 2013.

[129] A. Zeng and G. Cimini. Removing spurious interactions in complex networks. *arXiv:1110.5186v1*, 2011.

Summary

Exploiting Graph Properties for Decentralized Reputation Systems

Decentralized reputation systems are the most promising mechanism to establish trust among users in networks such as online markets on mobile devices, P2P file-sharing systems, and social networks, and to provide incentives to users to stay longer and contribute into the networks. In these systems, each user locally stores not only its own past interactions but the past interactions of other users, as well. Using this information, it computes the reputations of other users in order to take decisions about its future interactions. Due to the large number of the participating users, their highly dynamic behavior, the limited resources available at each users, several challenging scalability and security issues arise. In order to face those challenges, we divide the functionality of a decentralized reputation system at each node in three components: the collection of the history of user interactions, its storage and the computation of reputations. We model networks as growing interaction graphs, with nodes representing the users and edges the interactions between two users. By analyzing graphs deriving from real networks (such as Facebook, the author-to-author citation network, P2P file-sharing networks), and synthetic (random and scale-free) networks, we observe their graph properties and their evolution over time.

First, we focus on the computation of reputations. There are two widely used categories of algorithms for computing the reputations: max-flow based computations and random walks. Max-flow based computation is secure against attacks but inaccurate, and increasing its accuracy is computationally intensive. On the other hand, random walks are computationally efficient, accurate but vulnerable to a great range of attacks. We explore and improve both types of computations.

In Chapter 2, we show that for max-flow based computation, using the most central nodes of the network as start points of the max-flow algorithm increases its accuracy on the computed reputations, since the majority of nodes is reachable in a short distance by the most central nodes. However, identifying the most central nodes in a large network is computationally expensive due to the involvement of the all-pair shortest path problem. Our analysis in growing synthetic and real-world graphs reveals that in most reputation networks there are only a few central nodes that tend to keep their central position over time. Based on this observation, we evaluated approximate methods for identifying the most central nodes in our growing networks. Then, we show the effectiveness of using as starts points the most central nodes as computed by the approximate methods on the

computed reputations.

In random-walk based computations, nodes visited during a random walk treat all their neighbors equally, ignoring any properties they may have. Nevertheless, we show that the properties of a node such as its activity, and its position on the graph, indicate accurately its reliability, and that random walks exploiting these properties are more resilient against attacks than simple random walks. Through extensive analysis, we identify the properties of nodes indicative of their reliability and we bias random walks towards the most reliable nodes. Each node initiates its own random walks and computes its own personalized reputations for the other nodes. We evaluate biased random walks against free-riding and sybil attacks in growing synthetic and real-world graphs. We show that biased random outperform significantly random walks against attacks in all our graphs. Furthermore, we observe that the properties indicative of the reliability of nodes depend on the structure and the construction process of the corresponding graph.

In Chapter 4, we focus on the storage of the history of interactions at each node. Using the complete history of interactions on the computation of reputations is prohibitive due to its resource requirements. Furthermore, the complete history of interactions accumulates old information, which impedes the nodes from capturing the dynamic behavior of the system when computing reputations. We observe that all the historical interactions are not of the same importance due to their age, their strength, their reliability and their position in the graph, and we propose an algorithm for removing the least important interactions. We explore its effect on the computed reputations by both max-flow and random-walk based algorithms. Furthermore, we show its effectiveness in synthetic and real-world graphs.

Finally, we focus on the collection of the history of interactions. In a decentralized reputation system, each node collects information by other users. Most decentralized collection algorithms are based on epidemic protocols without assessing the relevance and trustworthiness of the exchanged information. Nevertheless, the past interactions of nodes can be indicative of similarity and trust between two nodes. Based on this observations, we propose a decentralized algorithm based on random walks where each node uses its own part of the history of interactions to navigate in the network and collect parts of the histories of interactions of other nodes. We integrate to it different types of random walks. We emulate the interactions in growing synthetic and real-world graphs and we explore the efficiency of the integrated random walks in terms of scalability, accuracy and robustness against Sybil attacks.

Samenvatting

Exploiting Graph Properties for Decentralized Reputation Systems
Gedecentraliseerde reputatie systemen zijn het meest veelbelovende mechanisme voor het bewerkstelligen van vertrouwen tussen gebruikers binnen verschillende typen netwerken, zoals online markten op mobiele apparaten, P2P systemen voor het delen van bestanden en sociale netwerken. Daarnaast kunnen deze reputatie systemen een beloning bieden aan gebruikers om langer in het netwerk te blijven en om meer bij te dragen aan het netwerk. In deze systemen slaat elke gebruiker niet alleen zijn eigen interactiegeschiedenis op, maar ook de interactiegeschiedenis van andere gebruikers. Met deze informatie kan de reputatie van andere gebruikers worden berekend, die daarna kan worden gebruikt om beslissingen te nemen over toekomstige interacties. Vanwege het grote aantal deelnemende gebruikers, hun zeer dynamische gedrag en de beperkte bronnen beschikbaar bij elke gebruiker, ontstaan er verscheidene uitdagende problemen op het gebied van schaalbaarheid en beveiliging. Om deze problemen aan te pakken, verdelen we de functionaliteit van een gedecentraliseerd reputatie systeem op elke knoop in drie componenten: de verzameling van de geschiedenis van alle interacties tussen gebruikers, de opslag hiervan en de berekening van reputaties. We modelleren netwerken als groeiende interactie-grafen, waarin knopen de gebruikers en lijnen de interacties tussen twee gebruikers voorstellen. Door de analyse van grafen die zijn afgeleid van echte netwerken (zoals Facebook, de auteur-naar-auteur citatie graaf en P2P netwerken) en kunstmatige (willekeurig en schaalvrij) netwerken, bestuderen we de eigenschappen van de graaf en hun evolutie.

Als eerste focussen we op de berekening van reputaties. Er bestaan twee veelgebruikte categorieen van algoritmes voor het berekenen van reputaties: algoritmes die zijn gebaseerd op het max-flow principe en random walks. Berekeningen gebaseerd op maxflow zijn veilig tegen aanvallen maar onprecies, en het verbeteren van die precisie is een kostbare taak. Daarentegen zijn random walks goedkoop in uitvoer en bieden ze een hoge precisie, maar zijn ze tegelijkertijd vatbaar voor een grote variëteit aan aanvallen. We verkennen en verbeteren beide type berekeningen.

In hoofdstuk 2 laten we zien dat het kiezen van de meest centrale knopen van een netwerk als startpunten van het maxflow-algoritme de precisie van de berekende reputaties verhoogt, omdat de meerderheid van de knopen dan bereikbaar is via een korte afstand vanaf deze startpunten. Helaas is het identificeren van de meest centrale knopen

in een groot netwerk duur vanwege het alle-paren kortste-pad probleem. Onze analyse van groeiende kunstmatige grafen en grafen uit de echte wereld onthult dat in de meeste reputatie netwerken slechts een klein aantal centrale nodes bestaat dat deze centrale positie ook daadwerkelijk vasthoudt. Gebaseerd op dit idee, hebben we methoden geevalueerd voor het benaderen van de meest centrale knopen in onze groeiende netwerken. Daarnaast hebben we de effectiviteit aangetoond van het gebruik van dergelijke methoden op de berekende reputaties.

In berekeningen die op random walks gebaseerd zijn, worden alle knopen bezocht tijdens de walk als gelijke beschouwd, en dus worden hun eigenschappen genegeerd. Wij tonen aan dat de eigenschappen van een knoop, zoals zijn activiteit en positie binnen de graaf, een goede indicatie geven van de betrouwbaarheid van de knoop. Daarnaast tonen wij aan dat random walks die rekening houden met deze eigenschappen beter bestand zijn tegen aanvallen dan traditionele random walks. Door uitvoerige analyse identificeren we welke eigenschappen de betrouwbaarheid van een knoop bepalen en we geven onze random walk een voorkeur naar de knopen met de hoogste betrouwbaarheid. Elke knop start zijn eigen random walks en berekent zijn eigen gepersonaliseerde reputaties voor de andere knopen. We evalueren hoe random walks met voorkeur reageren op free-riding en sybil aanvallen in groeiende grafen die zijn gebaseerd op kunstmatige en echte netwerken. We tonen aan dat random walks met voorkeur veel beter dan random walks reageren op aanvallen in al onze netwerken. Verder observeren we dat de eigenschappen die de betrouwbaarheid van de knopen bepalen afhangen van de structuur en het constructieproces van de bijbehorende graaf.

Vervolgens richten we ons op de opslag van de interactiegeschiedenis op elke knoop. Het gebruik maken van de volledige interactiegeschiedenis is niet mogelijk vanwege de vereiste capaciteit. Verder bevat de volledige interactiegeschiedenis veel oude informatie, die het moeilijk maakt voor de knopen om het dynamische gedrag van het systeem op te vangen tijdens het berekenen van de reputaties. We observeren dat niet alle historische interacties even belangrijk zijn vanwege hun leeftijd, sterkte, betrouwbaarheid en positie in de graaf, en we presenteren een algoritme voor het verwijderen van de minst interessante interacties. We verkennen het effect op de berekende reputaties door zowel max-flow als random-walk gebaseerde algoritmes. Verder tonen we de effectiviteit in kunstmatige en echte grafen aan.

Tenslotte focussen we op de verzameling van de interactiegeschiedenis. In een gedecentraliseerd systeem, elke knoop verzamelt informatie van andere gebruikers. De meeste gedecentraliseerde verzamelalgoritmes zijn gebaseerd op epidemische protocollen waarin geen rekening wordt gehouden met de relevantie en betrouwbaarheid van de uitgewisselde informatie. Niettemin kan de interactiegeschiedenis van knopen indicatief zijn voor gelijkheid en vertrouwen tussen twee knopen. Gebaseerd op deze observaties, stellen wij een gedecentraliseerd algoritme, gebaseerd op random walks, voor waarin elke knoop zijn

eigen deel van de interactiegeschiedenis gebruikt om binnen het netwerk te navigeren en om delen van de interactiegeschiedenis van andere knopen te verzamelen. We integreren dit algoritme voor verschillende type random walks. We emuleren de interacties in groeiende kunstmatige en echte grafen en verkennen de efficiëntie van de geintegreerde random walks op het gebied van schaalbaarheid, precisie en robuustheid tegen Sybil aanvallen.

Curriculum vitae

Dimitra Gkorou was born in Athens, Greece, on May 29, 1984. She studied computer science at the Computer Engineering and Informatics Department (CEID) of University of Patras, Greece from which she received her diploma in 2008. Her diploma thesis was fulfilled with the High Performance Information Systems Laboratory of CEID under the guidance of Prof. Efstratios Gallopoulos. In order to accomplish a part of her diploma thesis, Dimitra had an internship with the Signal Processing Laboratory of École Polytechnique Fédéral de Lausanne in Switzerland, under the guidance of Prof. Pascal Frossard. In her thesis, she studied and developed variations of the power iteration and gossip algorithms for the distributed averaging problem.

In 2009, she moved to the Netherlands to pursue a PhD in computer science with the Parallel and Distributed Systems Group of Delft University of Technology. Her research, which was guided by Prof. Dick Epema and Dr Johan Pouwelse, focused on designing decentralized reputation systems and eventually, lead on this thesis. In order to face the problem of identifying relevant and trustworthy information for computing the reputations, she proposed algorithms for collecting, storing and aggregating the history of user interactions in reputation systems by exploiting their network structure. Dimitra currently works as a data scientist at Xomnia.

Publications:

- Dimitra Gkorou, Johan Pouwelse, and Dick Epema, Trust-based collection of information in decentralized networks. *Under review*.
- Dimitra Gkorou, Tamás Vinkó, Johan Pouwelse, and Dick Epema, Leveraging node properties in random walks for robust reputations in decentralized networks. In *Proc. of IEEE International Conference on P2P Computing*, 2013.
- Dimitra Gkorou, Tamás Vinkó, Nitin Chiluka, Johan Pouwelse, and Dick Epema, Reducing the history in decentralized interaction-based reputation systems. In *Proc. of IFIP International Networking Conference*, 2012.
- Nitin Chiluka, Nazareno Andrade, Dimitra Gkorou and Johan Pouwelse, Personalizing EigenTrust in the face of Communities and Centrality Attack. In *Proc. of*

IEEE Advanced Information Networking and Applications, 2012.

- Dimitra Gkorou, Johan Pouwelse, and Dick Epema, Betweenness centrality approximations for an internet deployed p2p reputation system. In *Proc. of IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (HotP2P)*, 2011.

- Dimitra Gkorou, Johan Pouwelse and Dick Epema, Efficient Approximate Computation of Betweenness Centrality. In *Proc. of the 16th annual conf. of the Advanced School for Computing and Imaging (ASCI)*, 2010.

- Effrosyni Kokiopoulou, Pascal Frossard, and Dimitra Gkorou, Optimal Polynomial Filtering for Accelerating Distributed Consensus. In *Proc. of IEEE International Symposium on Information Theory*, 2008.