Dipartimento di Informatica e
Scienze dell'Informazione

Exploiting Social Networks in Robust P2P Applications

by

Matteo Dell'Amico

Theses Series **DISI-TH-2008-01**

DISI, Università di Genova
v. Dodecaneso 35, 16146 Genova, Italy `http://www.disi.unige.it/`

Università degli Studi di Genova

Dipartimento di Informatica e
Scienze dell'Informazione

Dottorato di Ricerca in Informatica

Ph.D. Thesis in Computer Science

Exploiting Social Networks in Robust P2P Applications

by

Matteo Dell'Amico

June, 2008

Dottorato in Scienze e Tecnologie dell'Informazione e della Comunicazione
Indirizzo Informatica
Dipartimento di Informatica e Scienze dell'Informazione
Università degli Studi di Genova

DISI, Univ. di Genova
via Dodecaneso 35
I-16146 Genova, Italy
http://www.disi.unige.it/

Ph.D. Thesis in Computer Science (S.S.D. INF/01)

Submitted by Matteo Dell'Amico
DISI, Univ. di Genova
dellamico@disi.unige.it

Date of submission: February 2008

Date of revision: June 2008

Title: Using Social Networks in Robust P2P Applications

Advisor: Giovanni Chiola
DISI, Univ. di Genova
chiola@disi.unige.it

Ext. Reviewers:
Licia Capra
University College London
l.capra@cs.ucl.ac.uk

Paolo Avesani
Fondazione Bruno Kessler, Trento
avesani@fbk.eu

Abstract

With the so-called "Web 2.0" phenomenon, Internet applications heavily based on user-generated content are currently experiencing a huge success. Information created by users get aggregated in order to provide new content with higher added value.

While a P2P infrastructure appears appropriate for applications where the content production is also decentralized, aggregating and reasoning about the content provided by peers is a significant challenge: nodes cannot have complete knowledge of the whole network, selfish users need to be motivated to cooperate through incentives, and the impact of malicious users needs to be limited, so that it is not possible for a small portion of them to jeopardize the entire system.

Webs of trust (WoTs), i.e. social networks that connect users according to their trust relationships, can be used to reason about the reputation of users. Reputation provides a means to minimize the impact of malicious and lazy peers and, on the other hand, to reward producers of high-quality and relevant content. However, since successful peer-to-peer applications have millions of users, webs of trust can become very large and volatile, exceeding the communication and storage capabilities of single peers.

In this work, we explore ways of modeling webs of trust, and working with them in peer-to-peer settings so that measures of reputation (or approximations of them) can be obtained using scalable and decentralized algorithms. This is done by storing, on each peer, a small subset of the WoT representing nodes related to that peer ("neighborhood map"), and by creating a distributed high-level representation of the whole WoT ("network map") where communities are clustered. When a peer needs to evaluate the reputation of another one, different neighborhood maps can be combined in order to obtain a relevant subset of the WoT; moreover, the network map can be used to guide the choice of the right neighborhood maps.

When user-supplied content is abundant, it is important to help users by recommending items that meet their subjective tastes; a new technique, called social filtering, integrates reputation information from webs of trust with traditional recommendation techniques based on collaborative filtering. In this way, users receive recommendations about content they are interested in and can appreciate, and the system is robust with respect to malicious attacks.

To my family

Love all, trust a few. (William Shakespeare)

Acknowledgements

I wish to thank Giovanni Chiola, Marina Ribaudo, and Giuseppe Ciaccio, for their supportiveness and the freedom I had when working on this fascinating job.

I am grateful to all the people I had as friends and colleagues, both at DISI in Genoa and at UCL in London. Afra, Alessandro, Anders, Andy, Anna, Bence, Betta, Chris, Daniele, Daniele, Davide, Dimitris, Elisa, Emanuele, Ettore, Federico, Gabriele, Gianluca, Gianluca, Giovanni, Giuseppe, Ilias, James, Laura, Licia, Marco, Maura, Maurizio, Michael, Michele, Mirco, Mo, Neal, Nicola, Nicoletta, Paola, Paolo, Paolo, Roberto, Salvo, Sonia, Sonia, Sylvia, Valentina, Valentina, Walter, and all the people I forgot: I hope you won't take it as a cliché when I say that spending time with you has really been a pleasure.

Table of Contents

Chapter 1 Introduction **5**

1.1 Incentives to Cooperation 7

1.2 Social Filtering 9

1.3 Thesis Structure 10

Chapter 2 Background **12**

2.1 Social Networks 12

2.1.1 Scale-Free Networks 13

2.1.2 Small-World Networks 14

2.2 Incentives to Cooperation 18

2.2.1 Prisoner's Dilemma 18

2.2.2 Reputation 21

2.2.3 Webs of Trust 22

2.3 Reputation Metrics 25

2.3.1 Trust Propagation Patterns 25

2.3.2 Symmetric and Subjective Metrics 27

2.3.3 Sybilproof Reputation Metrics 28

2.3.4 PageRank 30

2.3.5 Maximal Flow 31

2.3.6 Other Metrics 32

Chapter 3 Modeling Social Networks **33**

3.1 Related Work 34

3.2 Graph Construction 35

3.3 Network Properties 37

3.3.1 Degree Distribution 38

3.3.2 Average path length 41

3.3.3 Clustering 41

3.3.5 Hierarchical structure 43

3.3.4 Scale-Free Small Worlds 44

Chapter 4 Neighborhood Maps **46**

4.1 The Idea 46

4.2 Non-Empty Intersections 48

4.3 Compact Maps 49

4.3.1 Shortest Path Length 50

4.3.2 Personalized PageRank 51

4.4 Security Issues 55

4.4.1 Forged Maps 55

4.4.2 Flooding the web of trust 56

4.5 Experimental Validation 57

4.5.1 The OpenPGP Web of Trust 57

4.5.2 Shortest Path Length 59

4.5.3 Personalized PageRank 61

4.5.4 File-Sharing Simulation 65

Chapter 5 Mapping Small Worlds **68**

5.1 Network Maps 69

5.2 Related Work 70

5.2.1 Small Worlds and Navigability . . . 71
5.2.2 Graph Visualization . . . 72
5.2.3 Koren's Spectral Layout Algorithm . . . 72
5.3 Decentralized Implementation . . . 74
5.3.1 D-Orthogonalization . . . 74
5.3.2 Normalization . . . 77
5.3.3 The Decentralized Algorithm . . . 78
5.4 Experimental Validation . . . 78
5.4.1 Comparison Between Layout Algorithms . . . 81
5.4.2 Decentralized Algorithm . . . 82
5.4.3 Neighbor-of-Neighbor Routing . . . 82
5.5 Ideas for Future Work . . . 86

Chapter 6 Social Filtering **87**
6.1 Philosophy of the Approach . . . 87
6.1.1 Intent – Trust over Users . . . 89
6.1.2 Competence – Trust over Judgements . . . 90
6.1.3 The Combined Approach . . . 90
6.2 Realization of the Approach . . . 91
6.2.1 Evaluating Intent . . . 92
6.2.2 Evaluating Competence . . . 93
6.2.3 Example . . . 96
6.3 Attack Model . . . 98
6.4 Experimental Validation . . . 99
6.4.1 The Datasets . . . 99
6.4.2 Accuracy . . . 100
6.4.3 Robustness . . . 105

Chapter 7 Current Work and Conclusions **109**

7.1 Combining Network and Neighborhood Maps 109

7.2 Decentralized SOFIA 111

7.3 Conclusions 117

Bibliography **119**

Chapter 1

Introduction

The "Web 2.0" term describes most of the latest successful innovative WWW and Internet applications. As Tim O'Reilly argues [O'R05], previous applications were developed borrowing paradigms from pre-existent communication techniques (e.g., television, newspapers, catalogs); Web 2.0 applications are, instead, projected specifically for the Internet. The defining characteristics of this phenomenon are quite fuzzy, and O'Reilly defines it with some patterns that consistently repeat themselves. Among those, the blurring between the producers and consumers of content is arguably the most recurring characteristic, as witnessed by patterns such as "peer production", "the service automatically gets better the more people use it", and "users add value". In traditional mass-media, there is an intrinsic asymmetry between the few professionals who create books, films or newspapers and their vast audience. The Internet, instead, provides a technology where the common users can easily reach the mass; new applications are capitalizing more and more on this capability.

From an economic point of view, digital distribution has become viable for an extremely wide class of content: in his 2006 bestseller *The Long Tail* [And06], Chris Anderson describes the new opportunities that arise when the small actors add up to become a very relevant part of the whole market. This phenomenon appears in traditional markets, such as books (a website like `http://www.amazon.com/` sells a much wider variety of titles than an ordinary bookshop), and is a cornerstone of new ways of expression, like blogs or videos on YouTube. In the latter case, a substantial part of the society is a producer of content.

Many applications go beyond the mere fact that everybody can create new content, and are centered on the cooperation between many users. In a phenomenon that Dan Bricklin christened as "the cornucopia of the commons" [Bri00], it happens that massive access to a common resource makes it more valuable: during normal use, users add new socially-provided content to it. Wikipedia (`http://www.wikipedia.org/`) is a stunning example of how a website that can be edited by everyone can reach a surprising quantity (and

quality) of content. In folksonomies, such as `http://www.del.icio.us/` for web pages, users categorize content by labeling it with simple textual tags; the result is a very efficient system of "social bookmarks". Free software projects, such as the GNU/Linux operating system, show how a decentralized community of enthusiasts and professionals is able to create software that has a quality which rivals the proprietary alternatives. Social networks such as MySpace (`http://www.myspace.com/`) and FaceBook (`http://www.facebook.com/`) are tools that ease the creation of "personal pages", and are used for socializing and keeping in touch with acquaintances. Even the usage patterns of consumers are used to create new content by collaborative filtering systems: they create recommendations for users by analyzing the behaviors of peers with similar tastes.

While the social content production is strongly decentralized, Web 2.0 applications are usually based on a client-server software architecture, where a single organization usually provides the networking and hardware infrastructure, and manages the data of the users. We argue that, in many cases, socially-aware features can be very effective when included in peer-to-peer applications, ameliorating various issues caused by the centralized structure, such as those outlined in the following.

Privacy and data management. For the organizations that provide the applications, ownership of data created by users is one of the main assets. Thus, those organizations are unlikely to give full data access to other companies or users, for personal, business or research purposes. Moreover, users can be concerned with the disclosure policy of their own data. An open decentralized implementation would provide a clear policy of data distribution; systems can be devised where users control which other peers are allowed to read their information, according to their own preferences.

Interoperability. While some companies provide access to data in machine-readable format (e.g., web services), in many cases this still does not happen. Moreover, no explicit way is usually given to recognize which profiles on different services belong to the same user, making it unnecessarily harder to compose and integrate services. Instead of building "yet another collaborative application", developers would be encouraged to add new functionality to an already deployed open P2P infrastructure.

Efficiency. The presence of a central "bottleneck" limits the class of applications that can be implemented; a P2P architecture leverages on the CPU, bandwidth and storage capabilities of all peers to provide applications that need huge quantities of resources.

Although enabling "social features" on P2P systems is very appealing, there are various difficulties with this plan. The main goal of this work is obtaining *robustness* in the presence of lazy, malicious and collusive users. This objective is harder to be fulfilled when service is provided by peers instead of a centralized trusted server; the incentives to cooperation discussed in section 1.1 are our approach to the solution.

Scalable *aggregation* of the many contributions by different users is also crucial: the quality of the content in the eyes of the consumer does not only depend on the quality of the individual contributions, but also on how they are aggregated and filtered. Performing this aggregation and filtering in peer-to-peer systems is a complex issue: while huge P2P systems can have access to lots of resources, there is the additional requirement of a decentralized computation. The *social filtering* technique introduced in section 1.2 proposes to leverage on social knowledge in order to obtain quality recommendations which are resilient to attack.

1.1 Incentives to Cooperation

The "cornucopia of the commons" expression referred above was coined as an antithesis to the "tragedy of the commons" [Har68]. This phenomenon refers to situations where a finite resource such as pasture land or a fishing reserve is kept open to the public: users are tempted to abuse their freedom, inefficiently overusing the resource and ultimately leading to its consumption. All participants would be happier if a more moderate behavior could be enforced, but the absence of incentives towards cooperation generated the "tragedy". Is it possible to give access to a "common" resource so that a new user adds value to it? In other words, can the cornucopia effect beat the tragedy?

The tragedy of the commons is indeed a known problem in P2P applications: in fact, peers can be seen at the same time as servers and clients, on the one hand providing to other peers common resources such as information, bandwidth, storage space and/or CPU processing time, and on the other hand consuming those resources to obtain some service. To act as a client is obviously appealing – after all, nodes chose to join the network in order to benefit from the service; being a server, instead, can be costly, while there is no direct reward to a peer who gave something to others. A well-known measurement done on the Gnutella file-sharing network in 2000 showed that nearly 70% of the nodes were "free riders": they did not share any file [AH00].

Other measures, such as micropayments and simple confidence in the generosity of users, have been proposed [FC05b]. The standard solution for fighting free riding in peer-to-peer networks is however introducing the reciprocity principle. The idea is simple: a node treats a peer better if that peer served it well in the past. This way, peers have to be generous towards others if they want to be reciprocated. Reciprocity has been successfully implemented in various P2P file-sharing applications. The fact that eDonkey and BitTorrent, two protocols that implement this strategy, account for more than half of the total Internet traffic [Cac04, Cac05, Ipo06, Ipo07] is a testament to the effectiveness of this technique.

The reciprocity approach is useful under the assumption that two interacting nodes have

high probability of meeting again in the future: this will motivate a peer to behave well, in the hope that it will be reciprocated the next time they will meet again. In P2P networks, reciprocity is viable whenever nodes usually interact with the same peers (either because the network is small or interaction patterns are highly clustered), and/or mutual interactions have a sufficiently large duration. In the case of eDonkey and BitTorrent, those applications are primarily used to share very large files; in this case, the reciprocity principle works with peers that are downloading the same file: a node can share the pieces that were already downloaded to others, and it can be reciprocated with the complementary pieces that it had not downloaded yet. Unfortunately, in the more general case (known as "asymmetry of interest" [FLSC04]) where a peer A wants some service from B, but B does not need anything from A, A has no way to reciprocate for B's service, and incentives are not enough to foster cooperation.

A solution is found by introducing the notion of *reputation*. We can define reputation as a value which is computed as an aggregation of the assessments of each node: an image of a user's intentions from the point of view of the other peers in the network. If a node has a good reputation (i.e., it is consistently judged as well-intentioned by the other nodes in the network), then it will obtain better services. Opinions that peers have about each other form a *web of trust* (WoT); how this web of trust is formed depends on the characteristics of the application, but it should always be difficult, or costly, to be judged well by honest nodes: a good reputation has to be hard-earned. In the case of free riding, the web of trust can be formed by establishing endorsements from users who enjoy service to those who provide it to them. In this case, the principle of reciprocity is restored: it is not applied anymore between pairs of nodes, but between each node and the rest of the network. If a node provides good services to the rest of the network, then it will build a good reputation. This reputation allows that node to obtain better services. From this point of view, reputation can be seen as similar to money: it can be obtained and spent by respectively giving and receiving service; however, while money depends on a centralized authority which prints it, reputation is a decentralized currency which is "issued" collectively by all members of the community. In chapter 2, we will see that it is sensible to compute reputation subjectively, obtaining different values depending on the "point of view".

Obviously, computing reputation in a large-scale peer-to-peer network is not trivial. Issues that have to be taken in consideration are:

Lazy and malicious users. Just as users are tempted to free-ride, they may not be interested in providing accurate information about their trust. On the other hand, malicious users can put some effort in subverting the system.

Cheap identities. In many cases, it is quick, easy and extremely cheap to join the system and to create a new identity. This is desirable, since it allows the system to

grow faster, and quickly reach the virtuous circle of the cornucopia of the commons. However, malicious users can create new identities to erase bad reputations (*white-washing* [FR01]), or create hordes of "fake" users in order to subvert the functioning of the system (*Sybil attack* [Dou02]).

Lack of global knowledge. It is unfeasible for a single node to have knowledge about the whole web of trust, because it can be both very large (various peer-to-peer networks have millions of participants) and fast changing. For this reason, the analysis made by peers has to be done either in a distributed way or using a small subset of the whole WoT.

In this work, we investigate approaches that help perform scalable and secure computation of reputation in webs of trust, taking into account all the issues highlighted above. Regarding this isse, this thesis provides the following contributions.

- In chapter 3, we give a model to describe real social networks and build large realistic datasets.
- In chapters 4 and 5, we introduce scalable and decentralized algorithms to find trust paths and compute reputation in social networks, without requiring global knowledge.

1.2 Social Filtering

The "long tail" phenomenon implies that users are exposed to an extremely great variety in the offer of content. Users cannot be expected to browse through thousands, or even millions, of potential offers to find out what is most appealing to them. The usual approach for peer-to-peer systems is based on searching: users need to know beforehand what they are looking for, and a search based on keywords helps them in finding it. With this paradigm, the discovery of new content is difficult.

Another, more proactive, approach is based on *recommendations*. Usually, recommendations are produced by collaborative filtering algorithms. They analyze the tastes of users, and then recommend to a given user the items that have been liked by others with similar profiles. Recommendations make it easier to explore the "long tail", since they are able to expose rare content to users that are unaware of them. We are now in the paradoxical situation where centralized systems (e.g., YouTube) are exposing content which is produced in a decentralized way by lots of different users, while decentralized systems are mostly used to distribute, often illegally, items of wide appeal. We think that adopting recommendations in P2P systems can help bring the long tail to them.

Collaborative filtering systems are susceptible to attacks such as "shilling" [LR04]: peers introduce artifact data in the system, in order to boost (or decrease) the rating of some item. Usually, collaborative filtering systems measure their security against the fraction of malicious users in the network. However, in our scenario where cheap identities are available, the number of fake profiles created by an attacker can easily outnumber the honest ones. Our original approach, which we christened as "social filtering"[1] and described in chapter 6, is based instead on adopting a web of trust in order to compute the reputation of users and propagate trust to users which have both similar tastes and good reputation; similarly to other approaches that use webs of trust as a defense from misbehavior [FLSC04, CF05, YKGF06], we use reputation to limit the influence of the malicious users: while identities are cheap, reputation associated with them is expensive.

Perhaps surprisingly, our experimental results show that social filtering, in some real datasets, can obtain better results than traditional collaborative filtering even when there is no attack: this is due to the fact that the web of trust can carry additional information about the preferences of each user: two users that trust each other have a higher probability of having similar tastes, and this provides added value in addition to the judgments they express.

1.3 Thesis Structure

The rest of this thesis is structured as follows.

Chapter 2 introduces a background of related work regarding social networks and reputation systems for peer-to-peer systems.

In chapter 3, a model for social networks that present both of the ubiquitous characteristics of being scale-free and small-world is introduced.

Chapter 4 discusses *neighborhood maps*, a technique that can be used to compute decentralized approximations of metrics on graphs that can be used to compute reputation on webs of trust, without needing complete knowledge of the network.

Chapter 5 describes *network maps*, that is, a way of attributing coordinates to all nodes in a social network so that close nodes in the resulting layout are more likely to be linked. This allows for efficient "routing" on the social network.

Chapter 6 introduces *social filtering*, a method that adds reasoning on social networks to traditional collaborative filtering techniques. The SOFIA algorithm, also discussed in the

[1] In literature, the "social filtering" term is sometimes used as a synonim for Collaborative Filtering. In this work, however, we use this term only to refer to our approach which specifically requires both reputation and taste similarity.

chapter, is an instance of a social filtering algorithm.

In chapter 7, we conclude the work by describing current directions of research based on the techniques presented in this thesis and summarizing the contributions presented here.

Chapter 2

Background

In this work, to obtain novel solutions to the problem of creating secure P2P systems, we borrow ideas from different areas of research, such as the study of complex systems and social networks in particular, ranking of web pages for search engines, collaborative filtering, and graph drawing.

This chapter will discuss work related to social networks and the design of incentives for P2P systems, while what is more closely related with the solutions introduced in each chapter will be referenced in the corresponding Related Work section.

2.1 Social Networks

In recent years, large-scale complex networks as different as food webs, power grids, cellular networks, the WWW, the Internet and social networks have been the focus of intense studies. The catalyst was probably the fact that computers became able to process data about networks made of millions of nodes, and that many of those networks were easily accessible from the Internet.

Webs of trust belong to the category of social networks, that is, networks where nodes represent people and edges reflect some kind of social relationship between them. We are interested in complex networks because these studies provide us with knowledge about their structure, and we use this information to develop algorithms that work efficiently on real-world instances of webs of trust.

Traditionally, those networks were usually modeled with Erdös-Rényi (ER) random graphs [Bol01]. However, the real-world networks were discovered to consistently possess properties which were not observable in standard random graphs. The two most famous examples,

both brought to general attention in the late 1990s, are the *scale-free* and the *small-world* properties. Here, we will succinctly describe them; we refer to [New03] for a more in-depth analysis of complex networks.

2.1.1 Scale-Free Networks

One of the most striking differences between random graphs and many concrete examples of complex networks is their degree distribution. The theory for random graphs predicts a Poissonian distribution for the probability of node degree (i.e., the number of edges touching each node). However, in plenty of cases (among them, the Internet [FFF99], the World Wide Web [AJB99, BKM+00], unstructured P2P networks such as Gnutella [ALPH01, MIF02], etc.), the real observed distribution obeys to a power law: the probability $P(k)$ that a node has degree k is proportional to $k^{-\alpha}$. The α exponent varies, but usually belongs to the $[1.5, 4]$ range.

The impact of the power law on degree distribution is very strong. In random graphs, it is extremely unlikely that a node has a degree which is orders of magnitude higher than the mean. By contrast, in scale-free networks, a large percentage of the edges is directed towards few extremely "popular" nodes, called hubs. For example, in the WWW, the average number of links on each page is measured in tens, but a successful website such as eBay or CNN.com is linked by millions of pages.

The Barabási-Albert Model Albert-László Barabási and Réka Albert, in 1999, proposed a model to explain the scale-free phenomenon [BA99]. In their view, the uneven distribution of degree was due to a phenomenon called "preferential attachment"[1]. Their model explained the evolution of scale-free networks with these simple rules:

- When a new node joins the network, it creates m new edges towards old nodes;
- An old node has a probability which is proportional to its degree of being chosen for an edge.

Networks created in this way have a power-law degree distribution $P(k) \propto k^{-\alpha}$ with $\alpha = 3$. Preferential attachment models a "rich-get-richer" phenomenon, where the mere fact of having more links results in a higher probability of receiving more links. For instance, in the WWW, a page which has received more links will be more easily found and linked

[1] An essentially equivalent model was proposed more than 20 years earlier, in 1976, by Price [Pri76] to account for the scale-free distribution of the number of citations between scientific papers, but did not receive wide attention at the time.

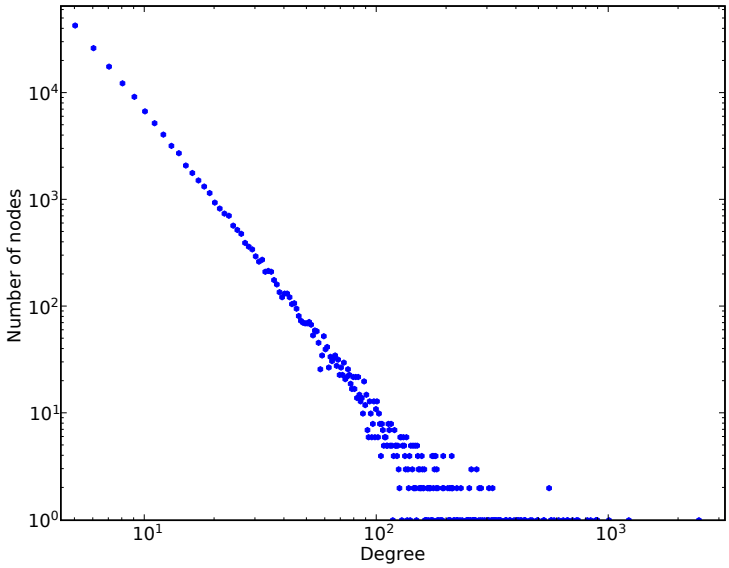


Figure 2.1: Degree distribution of a 150,000 node Barabási-Albert network ($m = 5$). In the log-log scale, the power law corresponds to a straight line.

again by other surfers. In social networks, friends can introduce other friends; this makes the social neighborhood of people who already have many friends grow faster.

Scale-free degree distributions can be also explained as results of iterated vertex copying [KKR+99] and optimization of connectivity versus wiring cost [FiCS03]; while these models can be interpreted differently from preferential attachment, the "rich get richer" statement remains true: in iterated vertex copying, a node that has many links is more likely to have one of them copied; in optimization, nodes that are well connected are better choices for the creation of a new edge, making them connected even better.

Since the degree distribution is very important for the characteristics of networks, the presence or absence of the scale-free property has to be taken in consideration when evaluating algorithms for webs of trust. In chapter 3 we develop a model for small-world networks which are scale-free as well, so that synthetic networks produced with our model can be used to verify how degree distribution affects algorithms applied to social networks.

2.1.2 Small-World Networks

The "small world" term is not new: it dates back to 1967, when the sociologist John Milgram published a famous study on the social network of people living in the USA

[Mil67]. The experiment consisted in sending some item (e.g., a book or a document) to a random person in the USA, and asking them to forward it to another person across the nation. The focal point is that the destination had to be reached via a chain of people that were mutually known personally. To put it in the computer scientist's point of view, people were asked to do "routing" on the social network.

Milgram obtained a surprising result: even if most acquaintances of a person belong to the same community (*cluster*) that person belongs to, the destination could anyway be reached in a very low number of steps. The mean length of the chains that reached destination, in one instance of the experiment [TM69], was found to be 6.2, thus inspiring the famous "six degrees of separation" adage. While sound critique [Kle02] has been moved to Milgram's methodology (in particular, 80% of the items did not reach the target, and therefore conclusions were based only on the few most fortunate cases), it is clear that average path length in known social networks [New03, table II] is consistently low.

The Watts-Strogatz Model In 1998, Duncan Watts and Steven Strogatz proposed a model to explain the small world phenomenon [WS98]. They introduced the *clustering coefficient* measure to capture the "cliquishness of a typical neighborhood". In a highly clustered network, two random nodes that are linked to the same third node are highly likely to be linked between themselves (closing a triangle), since they probably belong to the same community. For this reason, the clustering coefficient is defined for a vertex v as the fraction between the number of links between its k_v neighbors and all possible $\frac{k_v(k_v-1)}{2}$ pairs. The clustering coefficient for the whole network is defined as the average clustering for all nodes.

Erdös-Rényi random graphs have low (logarithmic in function of the number of nodes) path length; however, their average clustering converges to zero when the network grows: it is k/n on a node with average degree k and n nodes. Watts and Strogatz define small worlds in terms of their differences from ER graphs having the same number of nodes and edges: they have comparable average path length, but much higher clustering coefficient. Indeed, intuition suggests that the probability that two friends of mine are also friends does not depend on the size of humanity; put in another way, the clustering coefficient should tend to a nonzero value when the size of the network grows.

Examining the clustering coefficient and average path length of known social networks, it is apparent that – according to the definition of Watts and Strogatz – the small-world property is truly ubiquitous: basically any social network has small world characteristics.

The model that Watts and Strogatz developed is very simple: starting with a ring lattice with n vertexes and k edges per node (i.e., each node is connected to its k nearest neighbors on the ring), each edge is rewired to a random other node with probability p. Obviously, the structure of the network depends heavily on the parameter p. As figure 2.2 shows, for

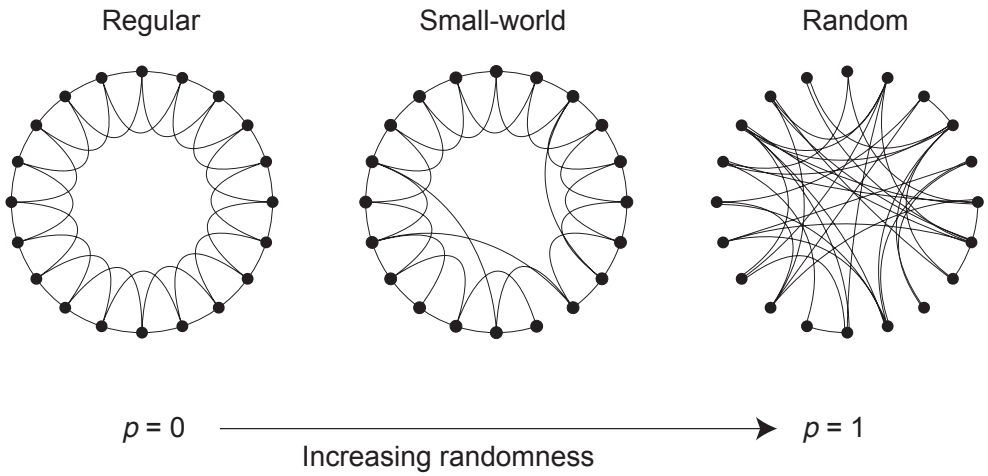


Figure 2.2: Watts-Strogatz small world model. From [WS98].

$p = 0$ the network is a highly clustered lattice with high ($O(n)$) average path length; for $p = 1$, the lattice is completely destroyed and the network becomes a completely random graph. The interesting phenomenon, though, appears for the intermediate values: there is a region where the diameter is very close to the logarithmic size of random network, and yet the clustering is still high. Later work [BW00] proved that the clustering coefficient does indeed converge to a nonzero value, which is dependent only on k and p when the size n of the network grows.

The Watts-Strogatz model can be interpreted in this way: a fraction $1 - p$ of the edges remain local, reflecting an underlying *structure* of the network that causes clustering (in this case, the ring); the remaining fraction p is composed of random edges that are responsible for the short path length. Thus, small worlds are products of both order (network structure) and chaos (the randomness of links).

The Kleinberg Model The Watts-Strogatz model gives good intuition on the reasons why small world networks appear. However, what perhaps is the most surprising result of the experiment of John Milgram was still not explained: short paths connecting people in social networks were not only *existing*: they could also be *found* by people who, of course, did not know the full web of social relations.

The ability of people to *navigate* small-worlds was investigated in 2000 by Jon Kleinberg [Kle00a]. In that work, Kleinberg created a variant of the Watts-Strogatz model based on a 2-dimensional lattice, and evaluated if nodes where able to find short paths knowing only local information: their own coordinates on the lattice, those of their neighbors, and the ones of the destination.

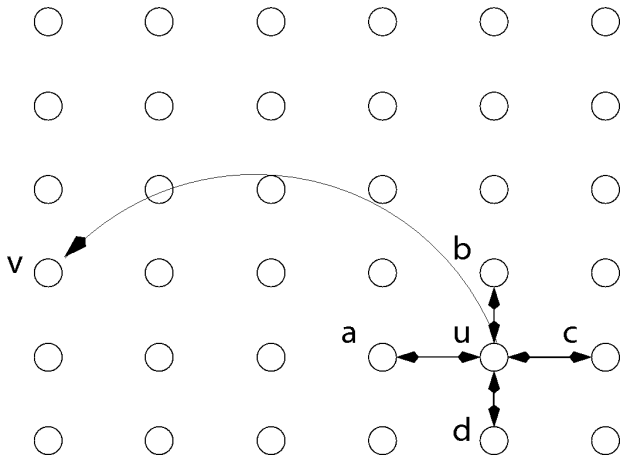


Figure 2.3: Kleinberg small world model. From [Kle00a].

In Kleinberg's model, each node u is connected with its four immediate neighbors (a, b, c and d in figure 2.3) and with a random other node v. Closer nodes are privileged for the random choice, and the probability that a node v is chosen is proportional to $r^{-\alpha}$, where r is the lattice ("Manhattan") distance between u and v, and the $\alpha \geq 0$ parameter is called "clustering exponent". When $\alpha = 0$, the model becomes close to a 2-dimensional variant of the Watts-Strogatz network since all nodes have the same probability to be chosen. On the other hand, when α grows, long-range edges are more and more discouraged.

The algorithm that nodes use to do routing on the social network is closely related with the greedy routing strategy employed in many Distributed Hash Tables, such as Chord [SMK+01]: at each step, the search proceeds towards the neighbor which is closest to the destination.

Kleinberg proved [Kle00b] that greedy routing reaches quickly the destination (in $O((\log n)^2)$ steps) if and only if the clustering exponent α is equal to 2. In all other cases, the routing time is polynomial with respect to the network size. This results prove that only a subset of small worlds are navigable (i.e., nodes are able to use a decentralized algorithm to do routing on the network).

In Chapter 5, we will refer to the reverse problem: Kleinberg studied which networks were navigable assuming a (distributed) knowledge about the whole network structure in the form of lattice coordinates; we will build a decentralized algorithm that attributes coordinates to the nodes in the network, so that the resulting information gives the network the best possible navigability.

2.2 Incentives to Cooperation

The already cited "tragedy of the commons" phenomenon, described by Hardin in 1968 [Har68], takes place when users belonging to a given community are offered a choice between a selfish and an altruistic use of a resource. Altruistic behavior would yield the best outcome for the community as a whole, but users are tempted to be egoistic because that will leave them better off. A familiar example is pollution: while limiting pollution may very well be worth the effort in terms of benefit for the community, the discomfort caused to a person by their own pollution is minimal.

The tragedy of the commons is recurrent in human society: in his original paper, Hardin cites the examples of open pasture, fishing reserves, visitors in parks, pollution, overpopulation, and noise pollution. In all those cases, Hardin advised to remove the choice through coercion, by outlawing egoistical behavior.

In our case, though, it is difficult to enforce a correct behavior: software-imposed limits can be overcome by patching the programs; moreover, since typically the only witnesses to interaction between two users are themselves, it is also difficult to ascertain whether a peer deviated from the mandated behavior. The free riding problem, instead, is usually addressed through *reciprocity*. Instead of removing the choice, it can be argued that we remove the common, in the sense that resources are not given indiscriminately to any user: they are directed preferentially towards those who are altruistic themselves. The incentive towards this discriminating behavior lies in the fact that those who provide good service will also receive good service from others.

2.2.1 Prisoner's Dilemma

Game theory gives tools to reason about strategic interaction of rational agents, and provides the most widely known formalization of the problem underlying the tragedy of the commons, with the prisoner's dilemma. Here we will not go farther than a brief intuitive introduction to the basic concepts; for a proper introduction to game theory see e.g. [Dut99].

In an instance of interaction between them (*game*), agents can choose between the available *strategies*. The result of the choices of all players is a *payoff* for each one of them, a numeric value which expresses the satisfaction level for the outcome of the strategies; higher payoffs are associated to the preferred results. The most common way of representing a two-player game is the *strategic* (or *normal*) form, which is represented as a matrix where rows correspond to the strategies of the first player, and columns correspond to the strategies of the second one. Each cell contains the payoffs p_r, p_c, respectively for the row and the column player. An example of game in strategic form is shown in table 2.1.

	Dove	Hawk
Dove	0,0	-1,+1
Hawk	+1,-1	-10,-10

Table 2.1: Example of game in strategic form: the "hawk-dove" [SP73] game. This game is symmetric, since players are presented equivalent strategies where payoffs depend on the strategies being played, and not on which player is playing them. Being a "Hawk" is beneficial in the presence of doves, but it is very dangerous when other hawks are around.

	Cooperate	Defect
Cooperate	3,3	0,5
Defect	5,0	1,1

Table 2.2: The Prisoner's dilemma in strategic form with canonical payoffs.

The prisoner's dilemma (PD) is probably the most famous application of game theory; the corresponding Wikipedia page[2] describes it in this way:

> *Two suspects, A and B, are arrested by the police. The police have insufficient evidence for a conviction, and, having separated both prisoners, visit each of them to offer the same deal: if one testifies for the prosecution against the other and the other remains silent, the betrayer goes free and the silent accomplice receives the full 10-year sentence. If both remain silent, both prisoners are sentenced to only six months in jail for a minor charge. If each betrays the other, each receives a five-year sentence. Each prisoner must make the choice of whether to betray the other or to remain silent. However, neither prisoner knows for sure what choice the other prisoner will make. So this dilemma poses the question: How should the prisoners act?*

A strategic form for the prisoner's dilemma is shown in table 2.2. An analysis of the payoff table shows that the two players would be satisfied with the results of mutual cooperation. However, from the point of view of one of the players, defection yields better results no matter what strategy the other player chooses (in game theory terms, the defection strategy *strictly dominates* cooperation). For this reason, it can be concluded that rational agents will always defect, leading to the undesirable equilibrium of two defections, with low payoff for both players.

The PD can be seen as a two-player instance of the tragedy of the commons. In fact, cooperation yields more benefit than cost. However, the problem lies in the fact that the player who pays the costs of cooperation is not the one who enjoys its benefits.

[2] `http://en.wikipedia.org/wiki/Prisoner%27s_dilemma`

Iterated Prisoner's Dilemma While the outcome for the PD is pessimistic, things change if the two players play it repeatedly. The iterated prisoner's dilemma (IPD) is a version of the game where instances of the PD game are repeated, with memory of the previous actions by both players. Now, the goal of the players is to obtain a maximum total payoff at the end of the iterated process. Two formulations of the IPD are known. In one case, the game is iterated for n times; in the other one, there is a probability p that the games stops at each iteration. Game theory predicts that perfectly rational players will always defect in the first case, while there is a space for cooperation in the second case if p is low enough, due to the fact that the threat of retaliation is strong enough.

The game theorist Robert Axelrod, in 1980, proposed an IPD tournament: he invited his colleagues to submit computer programs that played the IPD with 200 iterations, and played each other program in a round-robin tournament. In his 1984 book *The Evolution of Cooperation* [Axe84], Axelrod reported on the results. Quite surprisingly, the winner was the simplest program, Tit-for-Tat, developed by Anatol Rapoport. What Tit-for-Tat did was cooperating at the first move, and replicating the move of every other player at each step. A second edition of the tournament was held in 1984, when all participants were well aware of the outcome of the previous tournament. Even more surprisingly, Tit-for-Tat won again.

By analysing the most successful strategies, Axelrod came to the conclusion that successful strategies had the following characteristics.

Nice. They never defected first, thus not encouraging other nice strategies to defect and deviate from cooperation.

Retaliating. They responded to defections with defections, in order not to be exploited by non-nice strategies.

Forgiving. After retaliation, they were willing to return to cooperation if the other player stopped defecting.

Non-envious. They were not striving to obtain a higher score than the other player. The goal was to maximize payoff, not to overcome an opponent.

The work of Axelrod suggests that, using reciprocative strategies similar to Tit-for-Tat, cooperation can be fostered in P2P applications. Indeed, the extremely popular BitTorrent file-sharing application uses a strategy which is directly inspired from Tit-for-Tat [Coh03]: a "choking" algorithm on a given node A blocks upload towards all but the n peers who are most cooperative with A; a strategy of "optimistic unchoking" uploads data to unknown nodes in the hope of finding new peers with higher upload rate, similarly to the initial cooperation of Tit-for-Tat. Peers that adopt the other "big player" protocol in file-sharing,

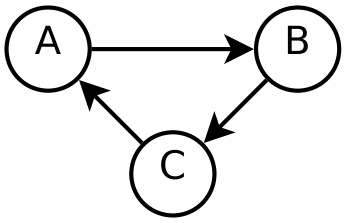


Figure 2.4: Asymmetry of interest. A wants resources from B, B from C and C from A. Direct reciprocation patterns cannot arise.

eDonkey, adopt another kind of reciprocative behavior by giving credits to nodes that upload data, and shortening the wait in queues for those nodes that gained more credits [eMu03]. A proposal for reciprocative behavior in unstructured networks such as Gnutella is SLIC [SGM04].

2.2.2 Reputation

Reciprocative approaches are very effective when the application can be actually modeled as an instance of the iterated prisoner's dilemma. Since BitTorrent and eDonkey are used, in most cases, to share large files, nodes spend a relevant amount of time during the download of a single file. In that period, they can upload to their peers the parts they have already completed, and ask them the ones they do not have yet. If the period is long enough, cooperation can emerge.

While in this kind of applications approaches based on direct reciprocity are successful, in more general cases they are not viable. The problem of *asymmetry of interest* [FLSC04], for example, arises when a node A wants something from B, but B does not want anything from A. In that case, the game does not resemble the prisoner's dilemma anymore, since the only peer who can choose whether to cooperate is B, and A has no way to reciprocate. Cyclic patterns such as the one in figure 2.4 show that in many cases a space from reciprocative cooperation exists, but nodes cannot obtain it just by using direct reciprocation.

Even if interest is symmetric, there is no guarantee that the prisoner's dilemma game played by the peers is indeed iterated: to make reciprocative behavior viable, one of the following conditions must be fulfilled.

Nodes are likely to meet more than once. Either the network has to be quite small, or the requests need to be clustered in sub-communities. However, in both cases nodes are only enjoying the services that are provided by few nodes. Since we are targeting cases where applications are more useful when nodes interact with many different users, this is bad news in our context.

	Cooperate	Defect
Recipient	1,-0.1	0,0

Table 2.3: Donor-recipient game. The donor (row player) can decide whether to donate some resource to the recipient. In this case, the donor has to pay a small cost, while the reward for the recipient is much higher.

Non-atomic interactions. This is the case in file-sharing applications: a long interaction such as a big file upload is split in many small sub-tasks (transfer of small parts), so that nodes can start rewarding cooperative peers before the full transaction is completed. However, this cannot be accomplished when the task cannot be split in verifiable sub-tasks.

To overcome these limits, *indirect* reciprocity is introduced. This principle states that peers can build a good *reputation* by cooperating with other nodes, and those with good reputation will be served better by the rest of the network. In the rest of this work, we will refer to reputation as a numeric value relative to the behavior of a peer as a function of the attestations of all peers in the network. As we will see in the following, there are many different ways to compute reputation, and it is possible (and advisable) that the reputation value is dependent on the peer who is evaluating it.

In 1998, Nowak and Sigmund [NS98] simulated a situation where interest is asymmetric by making agents of a large population meet other random agents and playing the donor-recipient game shown in table 2.3. Reputation was computed as the difference between the number of cooperation and defections in the past history of an agent, and the reciprocative agents decided to cooperate with others when their reputation was higher than a given threshold. Experiments conducted using evolutionary games showed that when the population size exceeded a certain threshold, direct reciprocation was not enough to encourage cooperation, but reciprocative behavior based on reputation was still effective. Later experimentation showed that indirect reciprocation is also effective when applied to humans [MSK02].

The work of Nowak and Sigmund showed that indirect reciprocation is feasible, but adapting it to peer-to-peer systems is not trivial: if each node computes its own reputation, then it is very easy for them to report false values; thus, reputation has to depend on assessments provided from other nodes.

2.2.3 Webs of Trust

After interaction between a node acting as a server and a node acting as a client, the "client" node can judge the quality of service received from the "server" with a *trust*

assessment. The reputation computed for the server node will be a function of all such assessments.

Here, we use the term "trust assessment" because we are interested in determining how much a given user is cooperative. If a "client" peer C obtained services from a "server" S, then C obtains, from its own experience, evidence to support the statement "I trust the behavior of S to be cooperative".

Indeed, past experience is only one of the possible ways in which a peer can obtain new trust assessments. For example, new users that join the network can boost their reputation rapidly by receiving endorsements from friends who trust them. Webs of trust can also be created via automated tools such as ReferralWeb [KSS97]. The only critical property for the security of a reputation system based on webs of trust, as we will see, is that it must be difficult for malicious nodes to be trusted by honest ones.

Since the client node can be malicious and provide dishonest judgments, it is important that the source of the judgment is recognizable: trust assessments can be formalized as a triple containing the identifier of the rating peer, the identifier of the rated peer, and the judgment expressed. Tampering can be avoided, for example, by using a public key infrastructure to sign judgments and using public keys as identifiers. A natural way to reason about such judgments is by seeing them as a graph with peers as vertexes and judgments as edges. We will call such graph a *web of trust* (WoT). The "web of trust" term was first coined for the PGP [Zim95] cryptographic application created in 1991. Since webs of trust can be considered as social networks, knowledge of the properties shown in section 2.1 will help us in the design of algorithms that work on them.

It can also be noticed that lazy users may be uninterested in giving faithful assessments; one way of motivating them is by treating recommendations as just another kind of service; nodes that give good recommendations will then obtain higher reputation. This, by itself, provides an incentive to recommend good peers.

Cheap Identities A relevant problem arises from the fact that, in most open P2P applications, nodes are free to join without the necessity of being identified. This simplifies the procedures to join the network, and encourages growth. However, this also means that a user with bad reputation can easily erase the identity associated with that reputation (*whitewashing*). Eric Friedman and Paul Resnick, in 2001, investigated the phenomenon [FR01] and concluded that

> *"...this introduces opportunities to misbehave without paying reputational consequences. A large degree of cooperation can still emerge, through a convention in which newcomers 'pay their dues' by accepting poor treatment from players who have established positive reputations".*

Indeed, whitewashing implies that newcomers cannot be distinguished from misbehavers. The good news is that reciprocative techniques are still effective [FC05a]. However, new users suffer from a period where their quality of service is low, because they still do not have positive reputation. This is a known phenomenon for users of BitTorrent or eDonkey: performances are quite poor at the beginning, but they become better when the users have built reputation and reciprocative patterns start to emerge. If those application enforced strong identities, "nice" approaches that cooperate with newcomers would be viable, and the problem would not manifest.

Indirect reciprocation can mitigate this bootstrapping problem, since each peer needs to undergo through it only once, in order to build its reputation, instead of needing to repeat it every time a new peer is met. Moreover, endorsements received from friends that are not based on reciprocative behavior can give good reputation even to users that are new to the network.

A consequence of cheap identities is that new users for whom nothing is known are indistinguishable from those who performed badly and resorted to whitewashing. Moreover, in many P2P applications, peers that chose not to provide some service are indistinguishable from those who actually *cannot* provide it (*untraceable defections* [FLSC04]). For this reason, statements of distrust do not appear essential in our webs of trust and are not taken in consideration in this work. We think that they can still be useful, for example to rapidly punish peers who start to misbehave, using techniques such as the ones seen in [GKRT04]. Investigating if and how distrust statements are useful in our case is left as future work.

Since our webs of trust will only contain positive trust assessments (*recommendations*), here we model them as directed graphs with edges labeled with positive weights, each weight reflecting the strength of the recommendation.

Sybil Attack Another consequence of cheap identities is that a single malicious user can create a very large number of identities ("Sybils") and, therefore, control a relevant fraction of the system, thus becoming able to subvert it. John R. Douceur introduced this under the name of "Sybil attack" [Dou02], after the protagonist of a book about multiple personality disorder. Douceur asserted that

> *"... without a logically centralized authority, Sybil attacks are always possible except under extreme and unrealistic assumptions of resource parity and coordination among entities".*

A simple countermeasure such as enforcing one identity per IP address or per cluster of IP addresses [DDCdVP+02] does not cope well with NAT[3] (where different hosts are identified with a single public IP address) or IPv6 (where IP addresses are abundant enough to allow each node to obtain a vast number of identifiers).

We can observe that Sybil attacks are not limited to peer-to-peer networks. False user profiles have been created in collaborative filtering systems to inflate the rating of some items ("shilling" [LR04] or "profile injection" [MBBW07]); fake ("spam") web pages are created to corrupt web page ranking [GGM05]; the absence of strong identities is used by spammers in e-mail systems, IRC chat channels and on Usenet newsgroups to avoid being banned. In this latter case, punitive countermeasures that block entire ranges of IP addresses (blacklists, Usenet Death Penalties) have to be used. Reputation, bringing back costs to the creation of identities that can actually do harm, could be a more fine-grained remedy that does not need to punish many innocent users.

2.3 Reputation Metrics

A reputation metric can be defined as a function that associates numeric reputation values to each node in a web of trust. A great number of choices are possible, and indeed literature provides a wide array of solutions. Many ideas can be borrowed from the field of *link analysis* [BRRT05] techniques, used for ranking of Web pages in search engines. These approaches view links between WWW pages as endorsements of authoritativeness: the author of a web page posts a link if he or she thinks that the destination is relevant to the topic at hand. Both the graph of the web and webs of trust are network of endorsements; therefore, computing reputation and ranking pages by relevance can be seen as two different instances of a more general problem.

2.3.1 Trust Propagation Patterns

Usually, reputation metrics are built iteratively: starting from a trusted node, trust is propagated to other nodes that are connected to the initial one via a particular pattern.

The most used trust propagation pattern is *transitivity*: it comes from the intuition that good nodes will give good recommendations, and thus have links towards other good nodes. So, if A trusts B and B trusts C, then trust on A will be propagated to B and then to C. As shown in figure 2.5, trust is thus propagated along directed paths (*trust paths*) in the web of trust. Usually, longer paths will carry less trust: it is normal to trust friends more

[3]Nodes using NAT can establish connections between them, and thus take part to P2P networks, using protocols such as STUN [RWd+03].

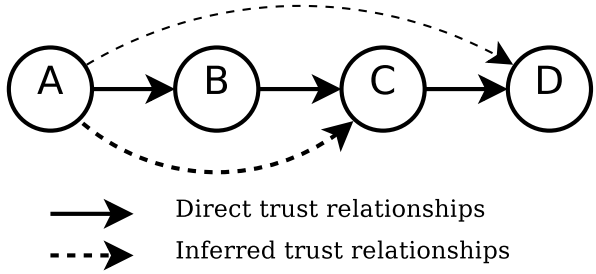


Figure 2.5: Trust transitivity propagation pattern. Trust is propagated from node A to B and then to C and D, according to the transitivity principle. The line connecting A to D is less thick because less trust is propagated through longer paths.

than friends of friends. Moreover, adding paths increases trust: if there are two paths from A to B, then the trust that A infers for B is at least as high as if only one path was present.

Especially when criteria for recommendations are subjective, the principle of transitivity has been criticized: we are not guaranteed to like the friends of our friends [Jö96, Lan03]; however, when the goal is encouraging cooperation, transitive approaches have been shown to be effective [FLSC04].

An alternative to the transitivity pattern is co-citation. This approach evaluates the reputation of each node in two different roles: the *recommender* and the *object of recommendations* (or, more in general, of judgments). Recommended nodes are not necessarily providers of good recommendations, and vice versa. For example, a great chef may not be an unbiased restaurant critic, and a good critic can be terrible in cooking.

The principle underlying the co-citation pattern is the recursive definition of good recommenders as those who endorse good authorities, and good authorities as those which are recommended by good recommenders. Trust is propagated along a zig-zag path, following alternatively forward (from recommenders to judgments) and backwards (from judgments to recommenders) paths, as shown in figure 2.6. It is irrelevant which nodes the judgments X and Y are about: to propagate trust, it is only needed that A and B agree on them.

The focus on recommender trust makes this approach work well especially in the case where judgments are subjective. Indeed, Collaborative Filtering algorithms are often based on this principle, propagating trust between users according to the similarity of their judgments. Instances of ranking algorithm for web pages that use this pattern are HITS [Kle99], SALSA [LM01] and BFS [BRRT05].

A weakness of ranking algorithms that use this principle is that it is quite easy to boost "recommender" trust, simply by copying popular judgments, and then diverting the recommender trust gained in this way towards the desired judgments [Mas05, DM06].

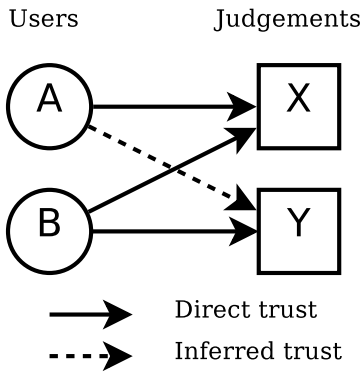


Figure 2.6: Co-citation trust propagation pattern. Trust is transferred from A to items recommended by B because A and B agree in their judgment X.

Because of this vulnerability to attack, we think that co-citation propagation patterns alone are not suitable for webs of trust aimed only at providing incentives to cooperation. However, it has been shown that algorithms using co-citation trust propagation have a result quality which is at least comparable to transitivity [BRRT05], and hybrid approaches can exceed the performance of both [GKRT04]. In chapter 6, we will tackle the case of information filtering, where subjectivity of judgments is high, and design a hybrid system using both patterns of transitivity and co-citation, obtaining a secure and effective filtering algorithm.

2.3.2 Symmetric and Subjective Metrics

Trust propagation patterns derive reputation starting from some given trusted nodes. This choice has a deep impact on the characteristics of the result.

In a P2P network, it seems reasonable to distribute this starting trust equally between all nodes, so that the reputation value is determined only by the link structure of the web of trust and not by the identity of nodes: after all, this corresponds to actually treating nodes as "peers". Cheng and Friedman [CF05] thus define as *symmetric* those reputation functions that return the same values to the same nodes after any renaming. Reputation values computed using symmetric reputation are shared by all peers, and can thus be precomputed and stored, for instance, in a distributed hash table such as Chord [SMK+01]. For example, computing the sum of weights for all recommendations a peer receives (*indegree*) constitutes a simple symmetric reputation metric.

When all nodes are equal for reputation purposes, malicious peers are as important as the honest ones, and this makes symmetric reputation metrics prone to misuse, especially in the case of Sybil attacks, where malicious nodes are abundant. An alternative is to

privilege a small set of nodes that are pre-trusted to be honest, and bias the result of the computation towards the recommendations given by these pre-trusted nodes. Examples of reputation metrics that employ this principle are the one used for the Advogato[4] community [Lev00] and, for P2P applications, EigenTrust [KSGM03]. Unfortunately, there are some contraindications to this idea in our setting. Firstly, this method introduces a centralized element in the system, which in turn implies critical points of failure: when pre-trusted nodes are compromised, reputation information becomes unreliable. Moreover, it becomes much more profitable to compete for the favor of the pre-trusted nodes rather than trying to help others, since a recommendation from a pre-trusted peer is going to be much more effective in terms of reputation. Another characteristic that has to be taken in consideration is that the presence of "super-judges" makes the architecture of the network susceptible to control and censorship from a single organization. PowerTrust [ZH07] is an alteration of EigenTrust: pre-trusted nodes are only used for initialization, and can be replaced by highly trusted "power nodes". This solves some of the issues with EigenTrust, but unfortunately the system appears still prone to abuse if a malicious node ever manages to become a power node or if a honest power node gets compromised.

A third idea is inspired by the quite dull consideration that each node trusts itself. If each node creates a subjective reputation metric by pre-trusting itself, then the issues of both previous approaches disappear. Moreover, subjective reputation metrics are helpful when the criteria of judgments that peers use to express their recommendations are subjective, especially for users belonging to a minority [Mas07, MA07b]. However, since reputation values are now dependent on both the evaluating and the evaluated node, in n-node networks the total number of computed values is n^2. In the case of large-scale networks with millions of nodes, pre-computing and then updating n^2 values after each change in the web of trust appears obviously untreatable. An alternative to pre-computation is calculating reputation values on demand. However, this is still a problem in large networks because the web of trust is very large, and thus it would be costly to store and update it, and analyze it every time this is needed.

In this work we choose to focus on personalized reputation systems, trying to alleviate the mentioned scalability issue. In chapters 4 and 5 we introduce criteria for peers to efficiently compute approximated reputation metrics by only using small subsets of the complete web of trust.

2.3.3 Sybilproof Reputation Metrics

Reputation can provide relief against Sybil attacks only if the reputation metric cannot be altered significantly by the attack itself. Cheng and Friedman [CF05] formalize Sybil

[4] `http://www.advogato.org/`

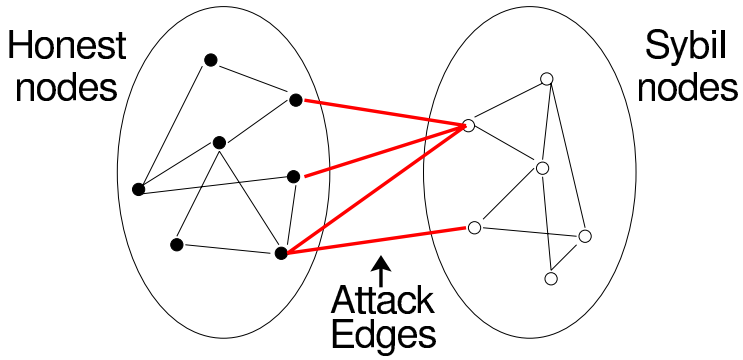


Figure 2.7: Attack edges between honest nodes and the Sybil region. From [YKGF06].

attacks as transformations where a single node S in a graph gets replaced by an arbitrary network of Sybils; Sybils can create an arbitrary network between themselves, but they cannot create new edges linking the Sybil network with the "honest" part: the only allowed edges are those that were already present before the attack. These definitions allow to formalize a definition for functions that are resistant to Sybil attacks. *Rank-sybilproof* functions are those for which Sybil nodes cannot escalate the reputation ranking: if a node X had higher reputation than S, no Sybil nodes has higher reputation than X after the attack. For *value-sybilproof* functions, no Sybil is able to obtain a higher ranking than S.

Cheng and Friedman formally proved two important properties: first, symmetric reputation functions are never neither value- nor rank-sybilproof; second, it exists a class of subjective reputation metrics that propagate trust along disjoint paths (using the transitive pattern) that are sybilproof. For example, the shortest path from the evaluating peer to the evaluated one is both rank- and value-sybilproof, and maximal flow with the evaluating peer as source and evaluated peer as sink is value-sybilproof. These results give further motivation to our choice of working on subjective trust metrics.

An ingenious way of protecting against Sybil attacks using webs of trust without resorting to reputation is SybilGuard [YKGF06]. The idea is based on the fact that paths in the WoT that connect honest nodes and Sybils must eventually cross an *attack edge* (figure 2.7) that connect a honest but mistaken node and a Sybil. As the definition of Cheng and Friedman also recognizes, attack edges cannot be created at will by the attacker; the number of attack edges can be used as a measure of the strength of the attack, and is the upper limit to the number of Sybils that Sybilguard will let into the system. To quote Stephen King, "the trust of the innocent is the liar's most useful tool".

The observation about attack edges helps understanding why ranking algorithms such as MaxFlow are resilient to Sybil attacks: the maximal flow between a honest node and a Sybil cannot exceed the number of attack edges.

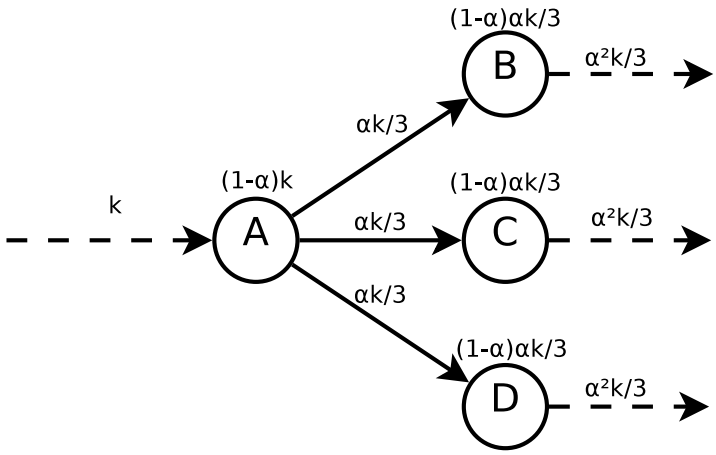


Figure 2.8: Trust propagation on PageRank. Node A has a probability k to be reached by the random walk; a part $(1-\alpha)k$ is kept as the reputation of A itself, while the remaining αk is split equally between the edges towards B, C and D. Those edges will as well keep a proportion $(1-\alpha)$ of it and propagate the rest to other nodes. If there are cycles, a part of that weight can return back to A.

2.3.4 PageRank

PageRank [PBMW98] is the most famous example of ranking algorithm for the Web, being used to rank Google search results. It employs the *trust transitivity* principle. In terms of the graph of the web, it is interpreted by stating that authoritative pages link other authoritative pages. PageRank simulates a random walk over the WWW graph (or a web of trust), starting from a random node and stopping after each step with a probability $1-\alpha$ (a usually recommended guideline value is $\alpha = 0.85$). Nodes are then ranked according to the probability that this random walk stops at them[5]. One of the virtues of PageRank is its quite fast convergence: in few (dependent from the parameter α) iterations having a cost that is proportional to the number of edges in the network, the algorithm converges to the solution.

In PageRank, each node is "responsible" for the probability k that the random walk eventually reaches it; of this quantity, a proportion $(1-\alpha) \cdot k$ is kept as the reputation for that node; the remaining αk quantity is split between all outgoing edges, as shown in figure 2.8.

PageRank is defined on unweighted networks, but it is trivial to extend it to the weighted case we are focusing on: on any given node, the probability that a link is chosen for the next step becomes proportional to its weight.

[5]The most common PageRank definition corresponds to the *equilibrium distribution* of an infinite random walk, with a $1-\alpha$ probability of jumping to a random node. The two definitions are equivalent.

Since it is a symmetric reputation metric, the work of Cheng and Friedman implicitly proves that PageRank is subject to Sybil attacks. Further work shows that the damage due to manipulation can be very high even with simple strategies and a low number of Sybils [FC06]. However, we are not aware of analyses of Sybil attacks in a simple variant, described in the same 1998 work that introduced PageRank: *Personalized PageRank* (PPR). In PPR, the starting point of the random walk is a privileged node; the resulting ranking will be biased towards endorsements expressed by that node.

Personalized PageRank is not sybilproof, because a malicious node S can create loops that inflate its weight beyond its original value. However, we can easily prove that the impact of malicious nodes on PPR is limited. In fact, a malicious node cannot divert to itself or to colluders a total weight which is more than the probability k that the random walk reaches it. This means that, if the node S initiates a Sybil attack (having, before the attack, a reputation value of at least $(1-\alpha)k$), the *total weight* of the Sybil region cannot exceed k. With our result, we limit the amount of damage that a Sybil coalition may inflict on PPR, in line with the considerations about resilience to attack of subjective reputation metrics.

A particularity of PageRank is that the amount of trust that nodes propagate is independent of their degree. For this reason, the only endorsement given by a lazy (or, maybe, very selective) node propagates ten times more reputation than each of the endorsements given by equally reputable nodes with degree 10. To mitigate this effect, Appleseed [ZL04], modifies Personalized PageRank so that each node has a link pointing back to the source (i.e., the evaluating node). With this alteration, nodes with lower degree propagate less reputation. The strength of this effect can be configured by adjusting the weight of these backward edges.

PageRank (and variations on it) has been extensively used as a tool to compute reputation in webs of trust. For a review, we refer to [HCH07].

2.3.5 Maximal Flow

Another popular reputation metric is maximal flow. Besides being proven (value-)sybilproof, its resilience to attack is well known [Lev00, FLSC04]. Unlike PageRank, MaxFlow is slow to compute. This encourages even more for the usage of the approximated, but more efficient, techniques of chapters 4 and 5.

Since MaxFlow is value-sybilproof, in the presence of a Sybil attack operated from S, no Sybil will obtain a ranking which is higher than the previous ranking of S. This reputation metric is not rank-sybilproof (Sybils can surpass honest nodes) simply because of the fact that a node, during the attack, can "revoke" endorsements to other nodes, by cutting paths that reach them. We think this cannot be considered a weakness of the metric, since it is natural that a node loses reputation if the recommendations it receives disappear. All

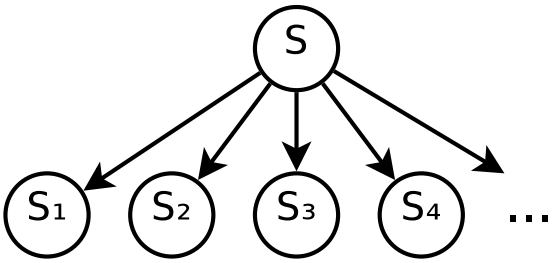


Figure 2.9: Attack to create unlimited total reputation in the Sybil network. The initiator S of the attack can create an unlimited number of Sybils $S_0 \dots S_i$, each with a finite reputation value which is not dependent on the number of Sybils.

reputation obtained from paths not passing through S remains unchanged.

It has to be taken in consideration that, unlike what enunciated for PageRank, MaxFlow will not limit the total reputation earned by the Sybil region: it is trivial for an attacker to obtain an arbitrary total reputation by creating enough Sybils, as shown in figure 2.9.

If the total weight of the Sybil region is relevant (for example, when peers are performing a vote weighted by their reputation), or if the computational requirements of MaxFlow are excessive, using PPR appears a good choice. If, on the other hand, the possibility of creating a set of fake users all with the reputation of the attacker is not abusable, then MaxFlow is advisable for its better resistance to attacks.

2.3.6 Other Metrics

When the whitewashing phenomenon is not present, it becomes interesting to take in consideration negative judgments as well as positive ones, and try to *predict* the assessment that would be given to a node rather than just isolating the best nodes. TidalTrust [Gol05] and MoleTrust [AMT05] strive to do this by propagating trust via a breadth-first visit of the web of trust; the judgment for a node is obtained as the result of a "vote" by trusted nodes.

Absence of whitewashing also enables to differentiate between malicious peers and those that are not trusted simply because they are not known. To do this, peers can use subjective logic, which combines information about belief, disbelief and uncertainty about a given statement (e.g., I trust peer X) [JHP06]. In our case, though, this idea unfortunately does not provide added value, because malicious nodes are able to whitewash disbelief into uncertainty.

Chapter 3

Modeling Social Networks

This chapter[1] introduces a model that encompasses networks that are both small-world and scale-free. As already discussed in section 2.1, these characteristics are ubiquitous in real-world social networks; this obviously motivates the need for models that synthesize both phenomena. Models are useful for two reasons: first, they give an explanation for the occurrence of the observed features; second, they can give an endless supply of synthetic networks for experimentation, as opposed to the limited number that can be obtained from real network data.

Our model is a generalization of the Barabási-Albert [BA99] algorithm. Nodes are placed on a ring; when a new node i joins the network in a random position on the ring, it creates m links towards old nodes. The probability p_{ij} that the old node j is selected as the destination for the link is dependent both on the j's degree k_j and on the distance d_{ij} on the ring (the length of the shortest circular arc connecting i and j), according to the following equation:

$$p_{ij} \propto \frac{(k_j)^\alpha}{(d_{ij})^\sigma}. \tag{3.1}$$

The α and σ parameters govern the relative importance of the two parts of the equation: $(k_j)^\alpha$ models "preferential attachment" as defined by Barabási and Albert and $(d_{ij})^\sigma$ is responsible for the small-world structure: σ plays the role of the "clustering exponent" that was introduced by Kleinberg [Kle00a].

When the two parameters assume values $\alpha = 1$ and $\sigma = 0$, the model reduces to the Barabási-Albert model; our simulation results show that the model produces scale-free,

[1] The work described in this chapter has been presented at the second European Conference on Complex Systems [Del06a]; an extended version is available as a technical report [Del06b].

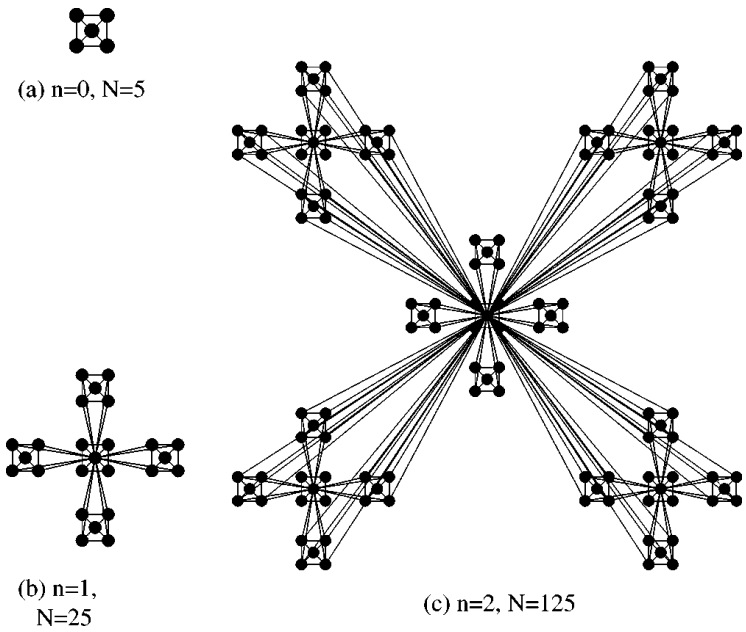


Figure 3.1: Ravasz-Barabási model for hierarchical networks. From [RB03].

small-world networks when $\alpha = \sigma > 1$.

Our synthetic networks can be labeled as small worlds for values of $\sigma > 1$; power-law degree distribution is observed when $\alpha = \sigma > 1$. With $\sigma > 2$, the model produces hierarchical networks [RB03].

3.1 Related Work

Various different models that generate networks with high clustering and power-law degree distribution have been proposed in the literature; in this section, we discuss some of them.

Hierarchical networks [RB03] are deterministically built by starting with a network consisting only of a small module, and iteratively copying n times the whole network, connecting the peripheral nodes of all copies to the central node of the starting module. This approach produces networks where clustering on nodes is inversely proportional to their degree; this feature is observed in many cases of real networks and, as will be shown in section 3.3, is also present in our model. As it can be seen in figure 3.1 this model, however, creates networks with a very regular topology that do not resemble real complex networks.

In other works [New01, HK02], high clustering is obtained simply by "closing triangles" in networks created following the preferential attachment rule: pairs of nodes with a common

neighbor are found, and new links that connect those neighbors are created. The model of Blanchard, Kruger and Rushhaupt [BKR05] shows how this mechanism, by itself, is capable of producing scale-free networks without the need of preferential attachment. This mechanism can be interpreted with the fact that, in many cases, new links are created after navigation on the network itself. For instance, we can get a new connection in a social network when an old friend introduces to us a friend of theirs; we may add a new link to our WWW page because we have seen that page linked in a page that we had already read (and linked). While this mechanic is realistic, it is reasonable to think that it coexists with the one we are describing. For example, we may get to know a person because he or she lives near us, even if we do not share common friends.

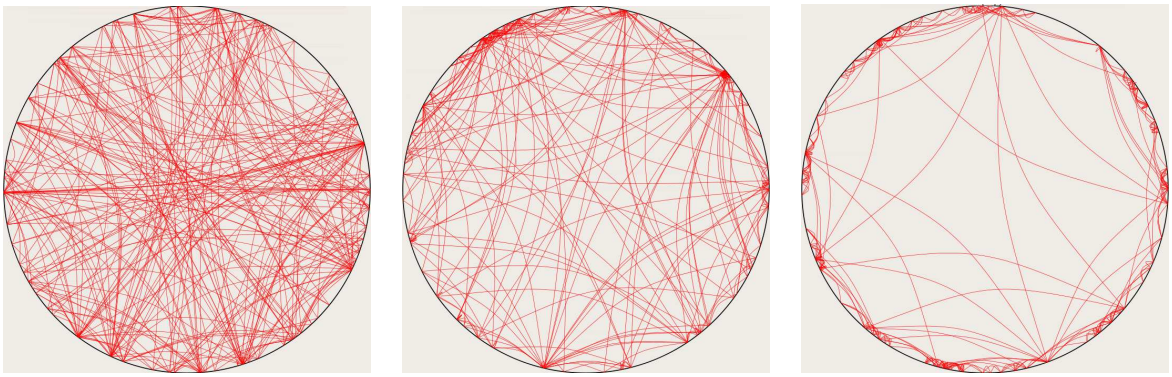
High clustering in power-law networks can also be a consequence of the optimisation of diameter versus wiring cost [MG01], and of preferential attachment networks when nodes, at some period in time, get "old" and cannot be linked anymore by new nodes [KE02].

Our approach, in which preferential attachment is dependent both on node degree and on Euclidean distance, has been tackled by other researchers. Manna and Sen [MS02] focused on link length distribution; Xulvi-Brunet and Sokolov [XBS02] studied degree distribution and average path length. Both papers restricted their analysis to the case where the α parameter of Equation 3.1 is equal to 1. Yook, Jeong and Barabási [YJB02] model the structure of the network of Internet routers with this approach. We are not aware of any other work, however, that has studied either clustering in such networks (showing that this model produces small-world networks) or the characteristics of networks when α gets large values and grows towards infinity.

3.2 Graph Construction

As claimed in section 2.1.2, small worlds can be interpreted as a consequence of the *affinity* between the nodes in the network that belong to the same cluster, reflected in some structure which is the basis from which the network itself is built: a ring lattice in the Watts-Strogatz model, a 2-dimensional lattice for the Kleinberg model. Groups of affine nodes are more likely to be linked together, and therefore to form clusters. For example, in social networks, two people are more likely to know each other if they live nearby, or if they share some interests or hobbies. In the graph of the WWW, two pages are more likely to be linked if they have a similar subject.

On the other hand, the Barabási-Albert model proposes that the node degree controls the way nodes choose which links to establish, according to the rich-get-richer principle. In our approach, we claim that networks evolve accounting for both factors, affinity and degree. For example, these two factors can model the evolution of the WWW: a user is likely to read, and link, Web pages close to their areas of interest (affinity) while, on the other hand,



$\sigma = 0$ (Barabási-Albert) $\sigma = 1$ $\sigma = 3$

Figure 3.2: Graphs produced using $n = 100$, $m = 3$, $\alpha = 1$ and different values of σ. While the number of edges, $nm = 300$, is the same in the three graphs, the ones having a higher value of σ have fewer long-range edges, resulting in a less cluttered picture.

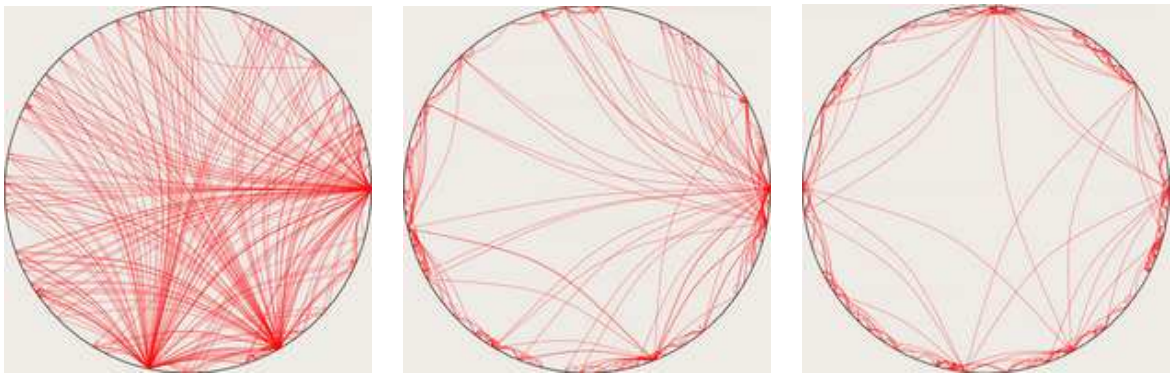
these pages need to be found in the first place, and "hubs" with many incoming links are more likely to be discovered, yielding also to preferential attachment. This is in accordance with typical surfing: when navigating, nodes with many incoming links are more likely to be encountered; when using search engines, pages that are often linked are ranked higher and, again, easier to be found.

In order to show the evolution of our model, we adopted the choice of positioning nodes randomly on a ring. This choice is, however, arbitrary; it is justified only by reasons of simplicity and symmetry. When modeling particular kinds of networks, different topologies can be better suited to the case: for example, a fractal is better suited to model the distribution of population (or computing equipment) over space [YJB02].

Our algorithm for building the graph is dependent on four parameters: two of them, α and σ, have already been introduced and control respectively the relevance of degree and distance on the ring for the preferential attachment. The other two control the size of the network: n is the number of nodes, and m is the number of edges a new node creates when entering the network.

The network is built starting with an empty graph. At each step $i \in 1 \dots n$, a new node v_i joins the ring in a random point; when the network does not yet contain m nodes ($i \leq m$), new nodes connect to all existing nodes. If $i > m$, instead, the probability p_{ij} that each node $j < i$ is picked as a neighbor for the new node gets computed according to equation 3.1; m nodes are introduced as neighbors of i in this way.

Figure 3.2 shows some graphs produced using this algorithm; the pictures show clearly that higher values of σ discourage the creation of long-range edges.



$\gamma = 0.5$ $\gamma = 1$ $\gamma = 2$

Figure 3.3: Graphs produced using $n = 100$, $m = 3$, $\alpha = \sigma = \infty$ and different values of γ. Graphs having low values for γ produce gelation (most nodes in the network with $\gamma = 0.5$ are connected only to the three hubs), while higher values have the effect of permitting less long-distance edges.

The results of section 3.3 will show that the $\frac{\sigma}{\alpha}$ ratio is important in the definition of the graph characteristic. Labeling this value as γ, when $\alpha \neq 0$, equation 3.1 can be rewritten as

$$p_{ij} \propto \left(\frac{k_j}{(d_{ij})^{\gamma}} \right)^{\alpha}. \tag{3.2}$$

An interesting case arises when the limit for $\alpha \to \infty$ is studied. When α grows, the highest probability in Equation 3.2 grows to 1; the nodes chosen for linking thus become the m ones that maximise

$$\frac{k_j}{(d_{ij})^{\gamma}}. \tag{3.3}$$

This case is interesting, since the edge selection becomes simpler and deterministic, the only random component being the choice of positions on the ring for new incoming nodes. With an abuse of notation, we will denote this case as having $\alpha = \sigma = \infty$, and we will study its behavior when varying the parameter γ. The following section will show that the family of graphs we obtain (see also figure 3.3) is by itself interesting, and produces small-world scale-free graphs when $\gamma = 1$.

3.3 Network Properties

This section introduces experimental results on how the choice of parameters influences the network characteristics of degree distribution, average path length, clustering and hierarchical structure. For a study of link length distribution, we refer to [MS02].

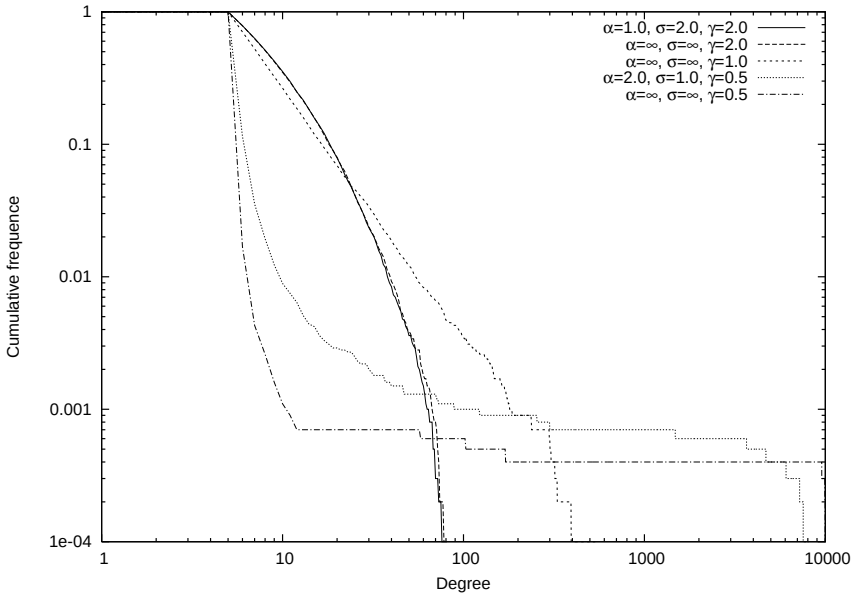


Figure 3.4: Impact of γ on the degree distribution. When $\gamma = 1$, the degree distribution is close to a power law, while other values produce markedly different distributions. When $\gamma < 1$, the network tends towards gelation, more significantly way when α and σ get larger.

3.3.1 Degree Distribution

Xulvi-Brunet and Sokolov [XBS02] studied this model for the case of $\alpha = 1$ and $\sigma \geq 0$, showing that the degree distribution has a power-law behavior when $0 \leq \sigma \leq 1$, while it follows a stretched exponential function having the form of $P(k) = a \exp(-bk^{\gamma})$ for $\sigma > 1$. The average path length increases logarithmically with the network size.

According to our simulation results applied to cases with $\alpha > 1$, the value of $\gamma = \frac{\sigma}{\alpha}$ appears fundamental for the topology of the network. As figure 3.4 shows, networks with $\gamma < 1$ tend towards gelation (i.e., most nodes are connected to - and only to - the same m hubs; those hubs have thus a degree that is close to n). When $\gamma = 1$, the degree distribution follows a power-law behavior. With $\gamma > 1$, power-law characteristics are lost.

Figure 3.5 shows that, when $\alpha = \sigma \geq 1$, the networks produced are always power laws, even if values close to 1 exhibit a significant deviation for the higher-degree nodes.

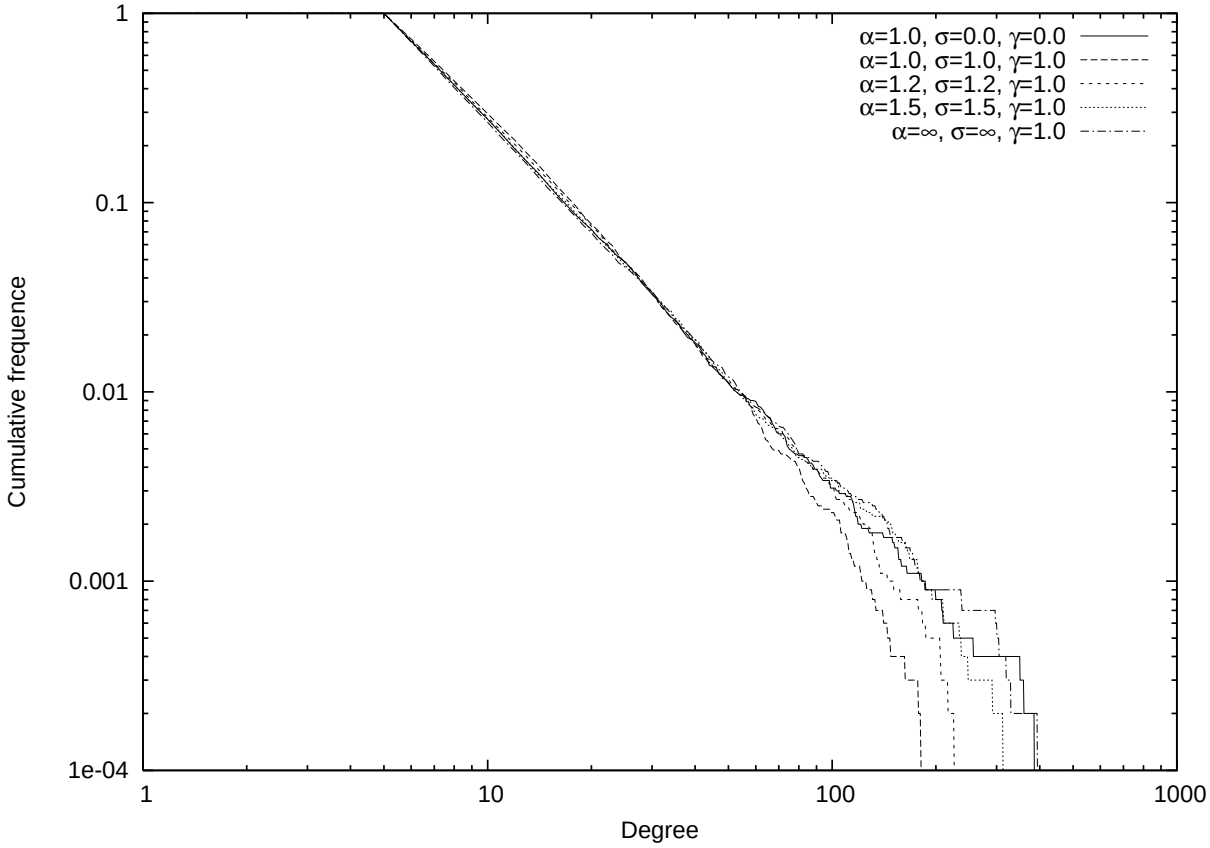


Figure 3.5: Cumulative distribution of degree when $\gamma = 1$. The cumulative frequency for a degree k is computed as $\int_k^\infty p(x)dx$, where $p(x)$ is the frequency of nodes having exact degree x. Sampled networks have $n = 10000$ and $m = 5$. When $\alpha = \sigma$ is larger than approximately 1.5, the degree distribution follows the power law (a straight line in this logarithmic scale) obtained with the Barabási-Albert model ($\alpha = 1, \sigma = 0$). When the $\alpha = \sigma$ values get closer to 1, there is a significant deviation from the power law for the higher-degree nodes.

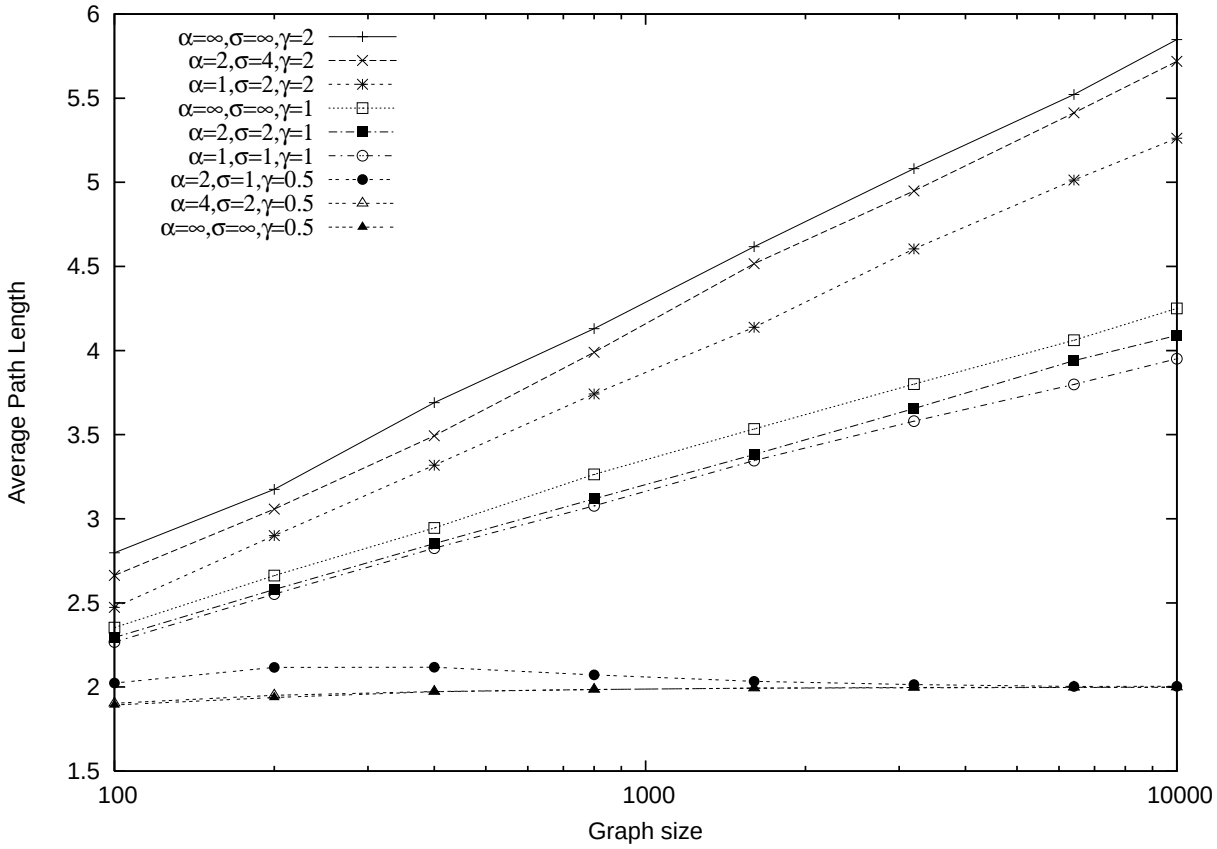


Figure 3.6: Evolution of average path length for different values of α and σ. It has been evaluated at different stages of evolution of the same network with $m = 5$; the average path length has been computed between 10000 random node pairs on the same network. The most significant factor is the γ value; in networks with $\gamma \geq 1$ the path length is increasing logarithmically in size (a higher value of γ implies higher path length); networks with $\gamma < 1$ tend towards gelation and an average length of 2 (almost every node is connected to a hub).

3.3.2 Average path length

The average path length is close to 2 for large gelated networks, since any two random nodes are with a high probability connected to a large hub. A logarithmic growth is obtained for the other cases (figure 3.6), compatibly with what is expected from small worlds.

3.3.3 Clustering

Two different definitions of clustering for a graph G are found in literature [New03, equations 3.3 and 3.5]:

$$C_0(G) = \frac{3 \cdot \text{number of triangles in } G}{\text{number of connected triples in } G}; \tag{3.4}$$

$$C_1(G) = \frac{1}{n} \sum_{i \in V} c_1(i), \text{ where } c_1(v) = \frac{\text{number of triangles connected to } v}{\text{number of triples centered in } v}. \tag{3.5}$$

The definition introduced by Watts and Strogatz and discussed in section 2.1.2 corresponds to C_1. The main difference between these two quantities resides in the fact that C_1 accounts equally for all vertexes, while C_0 gives a higher weight to higher-degree nodes, since those nodes are part of a higher number of connected triples. Since degree can be distributed very unevenly (this is actually the case in scale-free networks), these two measures can differ significantly. For example, the network of actors where edges represent acting in the same film has $C_0 = 0.20$ and $C_1 = 0.78$, while the co-authorship graph in physics papers has $C_0 = 0.45$ and $C_1 = 0.56$ [New03, table 3.1].

Networks created with $m = 1$ obviously have $C_0 = C_1 = 0$ because no triangles can be formed. In general, for all other cases, we experimentally discovered that networks created with $m > 1$ have $C_1 > C_0$. This is due to the fact that nodes having a large degree attract more long-range connections, and contribute less to the clustering. Since C_0 assigns more weight to high-degree nodes, the resulting total clustering is lower. In figure 3.7, we show that an increase in m increases clustering and decreases average path length.

A higher value of σ increases both clustering and network average path length. Anyway, while clustering rapidly approaches high values, the network average path length remains small (figure 3.8) and logarithmic with respect to the network size (figure 3.6). Thus, using $m > 1$ and $\sigma > 1$, the result is a highly clustered small-world network.

In many real systems, clustering values are conjectured to be roughly independent from the size of the network itself; this implies that clustering tends to a nonzero value when $n \to \infty$ [AB02, section III.F]. In our experiment, we calculated the clustering values for

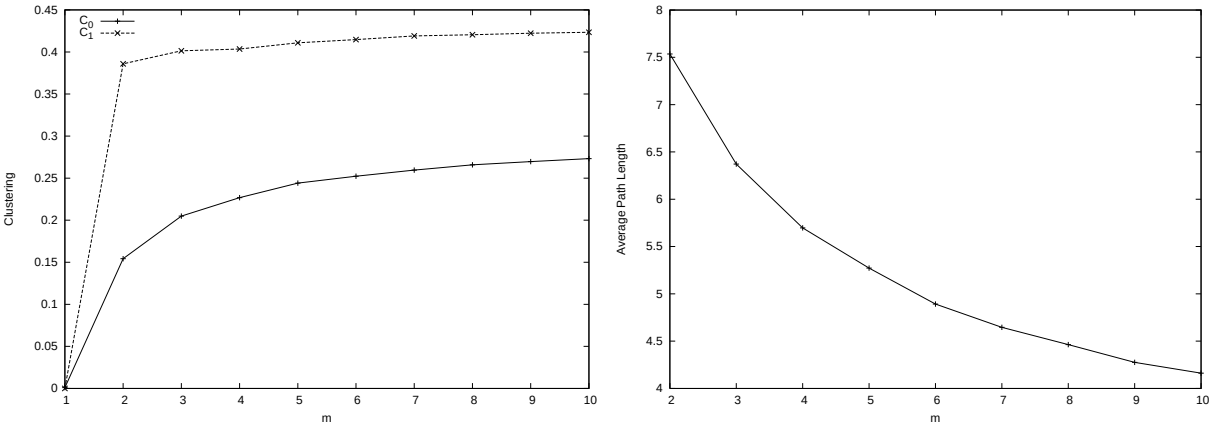


Figure 3.7: Impact of m on clustering (left) and path length (right). Networks analysed in the above graphs have $n = 10000$, $\alpha = 1$, $\sigma = 2$. The maximum possible value for m is $m = n$; in that case a complete network would be formed, with $C_0 = C_1 = 1$ and average path length 1.

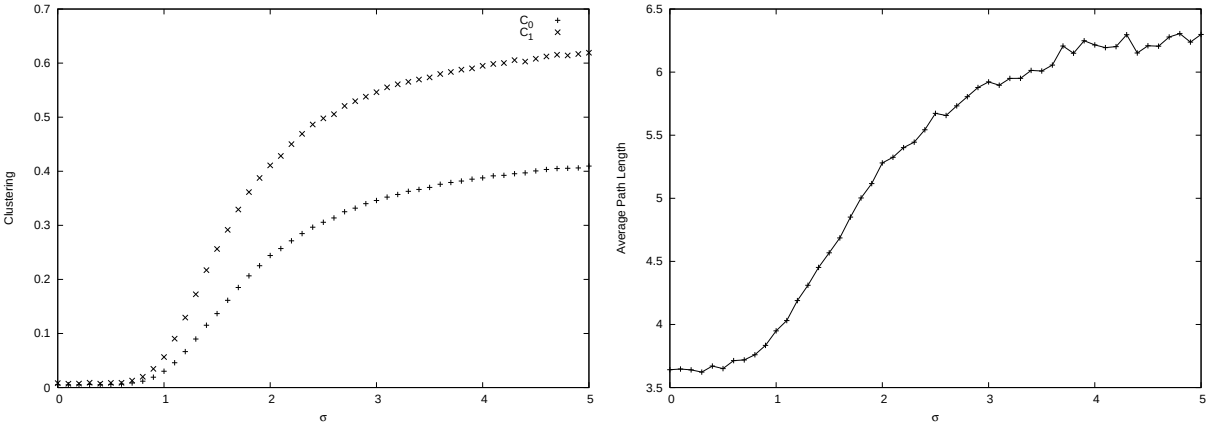


Figure 3.8: Impact of σ on clustering (left) and average path length (right). Networks analysed in the above graphs have $n = 10000$, $m = 5$, $\alpha = 1$. The value of both clustering and average path length starts increasing noticeably when $\sigma > 1$. The asymptotic limit value, obtained by selecting the m closest nodes at each iteration, has $C_0 = 0.45$, $C_1 = 0.64$, and an average path length of 6.66.

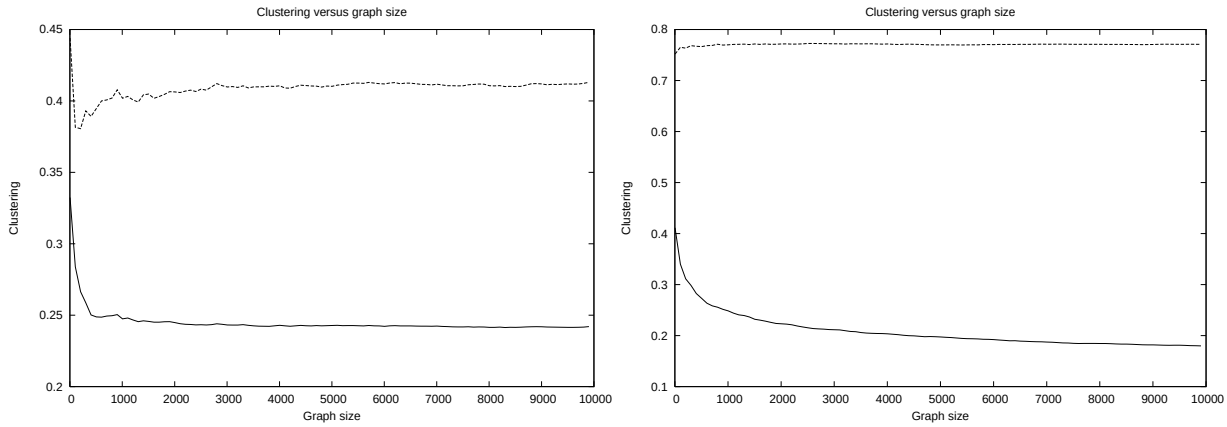


Figure 3.9: Clustering with respect to network evolution. Analysed networks have $m = 5$, $\alpha = 1$, $\sigma = 2$ (left) and $m = 5$, $\alpha = \sigma = \infty$, $\gamma = 1$ (right). These results suggest that clustering approaches a nonzero limit when the network size grows towards infinity.

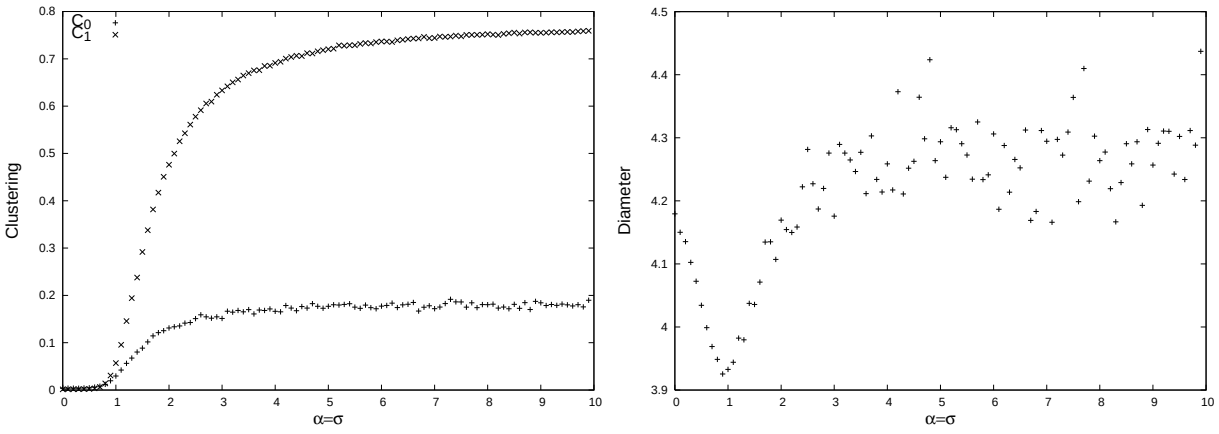
the network during the network generation itself. The results, shown in figure 3.9, suggest that in our model the clustering values either converge to a finite value or go to zero very slowly.

According to the results seen so far, the case of $\gamma = 1$ seems particularly interesting, since the resulting networks are small worlds with power law degree distribution. In figure 3.10, the effect of variation of $\alpha = \sigma$ on clustering and average path length is shown. The graphs show that these parameters are not heavily influenced by variation of $\alpha = \sigma$ when this value gets larger than approximately 5, and converge to the values obtained for $\alpha = \sigma = \infty$.

An interesting phenomenon, which has not been explained yet, is that the average path length has a clear minimum for $\alpha = \sigma = 1$.

3.3.5 Hierarchical structure

A hierarchical structure [RB03] is present when small groups of nodes are organised hierarchically in larger and larger groups. This can be measured by the fact that clustering on a node (the $c_1(v)$ value defined in Equation 3.5) is inversely dependent to the degree of that node. Some networks having $c_1(v) \sim k_v^{-1}$ are the actor network (connecting two authors if they acted in the same movie), the semantic web (connecting words if listed as synonyms in the Merriam-Webster dictionary), and the Internet at the autonomous system level. The networks we generate have a hierarchical structure when σ is large enough, according to the observations of figure 3.11.



3.3.4 Scale-Free Small Worlds

Figure 3.10: Impact of $\alpha = \sigma$ on clustering (left) and average path length (right) when $\gamma = 1$. Networks analysed in the above graphs have $n = 10000$, $m = 5$, $\gamma = 1$. Results obtained when $\alpha = \sigma = \infty$ are $C_0 = 0.176$, $C_1 = 0.770$, average path length 4.27. The C_1 value continues to increase when C_0 is already stabilised (approximately, when $2 \leq \alpha = \sigma \leq 5$), yielding a more relevant difference between C_0 and C_1 when α and σ are large. Average path length is not heavily influenced by the variation of α and σ, but the graph clearly shows a minimum average path length when $\alpha = \sigma = 1$.

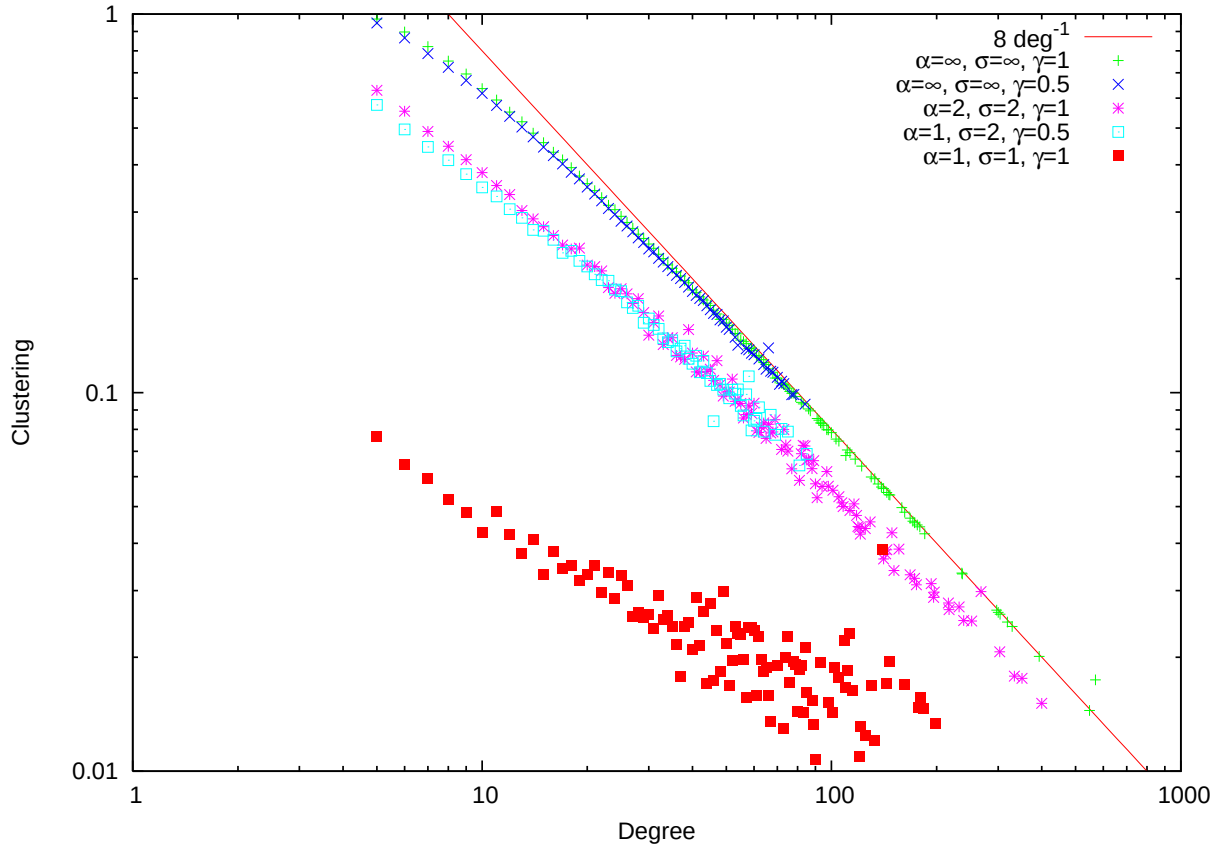


Figure 3.11: Clustering on nodes with respect to degree. The test graphs have $n = 10000$ and $m = 5$. When σ grows, the distribution of c_1 on nodes gets closer to proportionality with k^{-1} (the continuous line in the graph), conforming to the definition of hierarchical networks. Different values of α seem to have little effect on the points in the graph, influencing only the presence or absence of nodes having a particular degree.

Chapter 4

Neighborhood Maps

To compute reputation in a scalable way in webs of trust, one possible solution is to approximate the web of trust with a more manageable subset of it. In this chapter[1] we propose such a technique, storing on each node a small subset (*neighborhood map*) of the whole web of trust, containing only nodes that are reachable with short paths; when a node A wants to compute the reputation of B, the reputation is then computed using information from the maps of A and B.

Since shorter paths are the most important in providing an approximation for reputation metrics, the approximation provided by neighborhood maps is often reasonable when their size is proportional to the square root of the network size.

We test the effectiveness of neighborhood maps using data from the OpenPGP web of trust, a real-world network of trust relationships, and by developing a simple simulation of a file-sharing network using an evolutionary approach.

4.1 The Idea

Neighborhood maps are a data structure that contains a subgraph of the web of trust, consisting of the identifiers of the contained nodes and the links that connect them. Since social networks are usually sparse, a suitable way to represent them is through the use of adjacency lists.

This idea is inspired by the fact that, since shorter paths are usually more significant

[1]This chapter presents and expands work in press for Concurrency and Computation: Practice and Experience [Del08]; a previous version has been presented at the second International Workshop on Hot Topics in P2P computing [Del06c].

when evaluating reputation in webs of trust, they are sufficient to approximate reputation metrics.

In some cases, trust paths are found by exploring the web of trust (with, e.g., a breadth-first visit) until the desired node is found [Ati02, LSB03]. The problem with this approach is that it does not scale: obviously, in a n-node network, $O(n)$ nodes have to be contacted in order to find a given random node. However, we can exploit the fact that, when a node A is trying to compute the reputation of B, B is eager to provide evidence that can support its reputation.

Nodes can cache a (reasonably small) subset of the whole web of trust, containing short paths to and from themselves; since these caches contain information about the "neighborhood" of nodes in the web of trust, we call them *neighborhood maps*. The reputation metric can then be evaluated on the union of the two neighborhood maps of A and B. For example, if there are six degrees of separation from A to B, it would be sufficient to store nodes at distance up to 3 in the maps of A and B in order to find the paths of length 6 that connect the two nodes.

Another intuition on why small maps are sufficient to find paths between nodes is based on the birthday paradox. If samples are drawn (with replacement) from a large population of n items until one of them is drawn twice, the number of samples needed to extract two times the same item is very small with respect to n, and on average proportional to $\sqrt{n}$ [Sch96]. In our case, the phenomenon is beneficial, since it implies, as it will be more formally shown in section 4.2, that two neighborhood maps need to have size proportional to the square root of the number of nodes to have nodes in common, and thus to find trust paths on webs of trust. In large networks with millions of nodes, neighborhood maps will only need to cache information about thousands of peers.

Neighborhood maps can be built efficiently, requiring only communications between neighbors in the web of trust. For example, maps containing all neighbors-of-neighbors (i.e., nodes at distance 2) can be built in one step if each node sends the list of their neighbors to each one of them; maps containing all nodes at distance up to 3 can be built by sending neighbors-of-neighbors maps, and so on. Here, we will consider the case of static webs of trust; in the more realistic case of dynamically changing webs of trust, a node can update its map simply by periodically asking its neighbors to re-send their maps.

In algorithm 1, we describe a simple algorithm that can be used to build neighborhood maps.

Algorithm 1 Map construction algorithm

Parameters: the desired size k of the neighborhood map.
Return value: on each node A, a WoT subgraph M_A that contains nodes close to A.

On each node A, execute the following code:

$M_A \leftarrow$ the WoT subgraph containing A and its neighbors

while sizeof(M_A) $< k$ **do**
  $M'_A \leftarrow$ the WoT subgraph containing A and its neighbors
  for all neighbors X of A **do**
    Obtain the current value of M_X from node X
    Add M_X to M'_A
  end for
  $M_A \leftarrow M'_A$
end while
if sizeof(M_A) $> k$ **then**
  Remove from M_A k elements at maximal distance from A
end if
return M_A

4.2 Non-Empty Intersections

Neighborhood maps are useful for finding paths in webs of trust if, when combined, they do not create two disconnected components around the evaluator A and the evaluated node B. In other words, the intersection between the sets of nodes in the two webs of trust has to be non-empty.

In this section, we will focus on the simplified case where neighborhood maps are composed by a random selection of nodes[2].

In a network with n nodes, the probability p that two random subsets of k nodes have at least one node in common is

$$p = 1 - \frac{\binom{n-k}{k}}{\binom{n}{k}} = 1 - \prod_{i=0}^{k-1} \frac{n-k-i}{n-i} \geq 1 - \left(\frac{n-k}{n}\right)^k. \tag{4.1}$$

[2]This assumption is not true even in random networks because they have an unequal degree distribution; however, our experimental results show that these values provide a good qualitative approximation for ER graphs.

Probability	Map size
0.5	$0.83\sqrt{n}$
0.9	$1.52\sqrt{n}$
0.99	$2.15\sqrt{n}$
0.999	$2.73\sqrt{n}$

Table 4.1: Probability of non-empty intersections with respect to map size for $n \to \infty$, under the assumption of random maps.

To observe the behavior of this probability when n grows, we study the limit for $n \to \infty$ of the right member of equation 4.1:

$$\lim_{n\to\infty} p \geq \lim_{n\to\infty} \left(1 - \left(\frac{n-k}{n}\right)^k\right) = 1 - \lim_{n\to\infty} \left(\left(\frac{n-k}{n}\right)^n\right)^{\frac{k}{n}} =$$

$$= 1 - \lim_{n\to\infty} \left(e^{-k}\right)^{\frac{k}{n}} = 1 - \lim_{n\to\infty} e^{-\frac{k^2}{n}}. \tag{4.2}$$

The value of equation 4.2 depends on $\lim_{n\to\infty} \frac{k^2}{n}$. If $\frac{k^2}{n}$ grows to infinity (i.e., k grows faster than $\sqrt{n}$), then the probability p that non-empty intersections (and trust paths) are found converges to one; on the other hand, if k grows slower than $\sqrt{n}$, then neighborhood maps contain disjoint sets of nodes.

By choosing a value of $k = t\sqrt{n}$, the probability p of having a non-empty intersection is bounded by

$$p \geq 1 - e^{-t^2}.$$

To have non-empty intersections with probability p between two maps, it is thus necessary to have maps of size at least

$$t = \sqrt{-\ln(1-p)} \cdot \sqrt{n}.$$

Table 4.1 compares sizes of neighborhood maps with the probability of having non-empty intersections.

4.3 Compact Maps

Let us denote the reputation of node B from the point of view of A as $R(A, B)$. In some cases, given a node X belonging both to A and B's maps, the reputation score due to paths that pass through X can be expressed as some function $f\left(R(A, X), R(X, B)\right)$. Since only the $R(A, X)$ and $R(X, B)$ values are needed to compute our desired reputation vaule, storing the whole subgraph representing trust relationships between nodes in the map is

not necessary. We can instead use smaller structures – *compact* maps – that, on a given node A, store:

- the reputation values $R(A, X)$ of close nodes from the point of view of node A (*outgoing* sub-map);
- the reputation of A, $R(X, A)$ from the point of view of close nodes (*incoming* sub-map).

The desired reputation value $R(A, B)$ can be estimated by finding nodes that appear both in A's outgoing map and in B's incoming one. In this section, we show how it is possible to compute compact maps that approximate the shortest path length and Personalized PageRank.

By using compact maps, we lose track of which particular trust paths are used to construct reputation. We think that this does not represent in itself a robustness issue, as long as each node retains control on the outgoing trust paths by being able to choose outgoing links. Other security issues regarding the possibility of creating forged maps or "denial-of-service" attacks are discussed in section 4.4.

4.3.1 Shortest Path Length

The shortest path length is a valid, if simplistic, way of computing reputation on webs of trust. It is quite natural to think that nodes recommended by a "friend" (i.e., at distance 2) can be trusted up to a certain degree, those at distance 3 to a lesser degree, and so on[3].

In this case, the neighborhood map for a node A just needs to store the *distance* of nodes that *can be reached* from A via short paths ("outgoing" sub-map) and about nodes that *can reach* A with short paths ("incoming" sub-map).

When a node A wants to compute its distance from B, nodes that belong both to A's "outgoing" map and B's "incoming" map are taken in consideration. For each node X belonging to the intersection, the sum of the distance from A to X and the distance from X to B measures the length of a path from A to B. The shortest such length is used as an estimation of the path length from A to B.

Example Let us refer to the example graph of figure 4.1 and consider the case of node A evaluating its distance from node G. The parameter k that specifies the size of the neighborhood maps is set to 4.

[3]For example, shortest path length is used in the OpenPGP web of trust to help users estimate the reliability of a user; see e.g. `http://www.lysator.liu.se/~jc/wotsap/` and `http://pgp.cs.uu.nl/`.

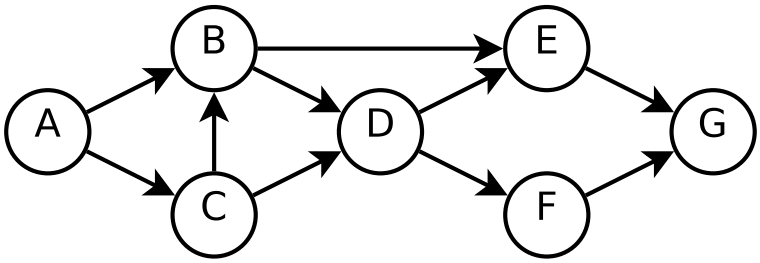


Figure 4.1: Example graph.

To build its "outgoing" map, node A asks its neighbors B and C for their list of neighbors; in this way, A knows that D and E are reachable via 2-length paths. Since A's map can only store four nodes, one of them has to be discarded; let us suppose that the information about D is kept and E is discarded, so that the map for "outgoing" paths from A is $[A \to 0, B \to 1, C \to 1, D \to 2]$. Analogously, G creates its map for "incoming" paths; one node between D and B has to be discarded. Let us suppose that G's "incoming" map is $[G \to 0, E \to 1, F \to 1, B \to 2]$.

The only node that belongs to both maps is B, and the path length found is 3, corresponding to the $A \to B \to E \to G$ path. If F discarded the information about B and kept D in its map instead, the result would have been 4, representing for example the $A \to B \to D \to E \to G$ path. In this case, the result would have been an underestimation of the correct reputation value.

4.3.2 Personalized PageRank

Personalized PageRank, described in section 2.3.4, can be approximated with compact maps. Given a parameter α, the ranking $R(A, B)$ is the probability that a random walk that starts at A and stops at every step with probability $1 - \alpha$ finishes at B. If $R(A, X)$ and $R(X, B)$ are known, the probability that a random walk starting at A finishes in B *passing through* X is

$$\frac{\alpha}{1-\alpha} R(A, X) \cdot R(X, B). \tag{4.3}$$

In fact, $\frac{\alpha}{1-\alpha} R(A, X)$ if the probability that the random walk arrives at X *and continues*; $R(X, B)$ is the probability that it afterwards stops at B.

Constructing Neighborhood Maps Denoting w_{AB} as the weight of edge connecting nodes A and B and $N(A)$ as the set of neighbors connected to A via outgoing links, the

following equation holds:

$$R(A,X) = \begin{cases} (1-\alpha) + \alpha \dfrac{\sum_{Y \in N(A)} w_{AY} R(Y,X)}{\sum_{Y \in N(A)} w_{AY}} & \text{if } a = b, \\ \alpha \dfrac{\sum_{Y \in N(A)} w_{AY} R(Y,X)}{\sum_{Y \in N(A)} w_{AY}} & \text{otherwise.} \end{cases} \tag{4.4}$$

Equation 4.4 expresses $R(A,X)$ as a function of $R(Y,X)$ for all neighbors Y of A. In algorithm 2, each node exploits this equation to compute "outgoing" maps needing only information from their neighbors. To keep the size of neighborhood maps limited, after each iteration only the k largest reputation values are kept. The algorithm returns, for each node A, a dictionary (i.e., a mapping implementable with a hash-table) O_A containing an approximation for the nodes X with highest $R(A,X)$ values.

Denoting as $N^{-1}(B)$ the set of all nodes with links directed to B, the following equation is also satisfied:

$$R(X,B) = \begin{cases} (1-\alpha) + \alpha \sum_{Y \in N^{-1}(B)} \dfrac{w_{YB}}{\sum_{Z \in N(Y)} w_{YZ}} R(X,Y) & \text{if } a = b, \\ \alpha \sum_{X \in N^{-1}(B)} \dfrac{w_{YB}}{\sum_{Z \in N(Y)} w_{YZ}} R(X,Y) & \text{otherwise.} \end{cases} \tag{4.5}$$

Just as equation 4.4 expressed $R(A,X)$ in function of all values of $R(Y,X)$ for $Y \in N(A)$ and allowed A to compute its outgoing maps based on those of nodes in $N(A)$, equation 4.5 expresses $R(X,B)$ as a function of all $R(X,Y)$ for all $Y \in N^{-1}(B)$. Analogously to the case of maps for outgoing paths, algorithm 3 computes maps for "incoming" paths. We denote the map for node B as I_B.

Estimating Personalized PageRank Let us say that peer A wants to evaluate peer B's PageRank value (i.e., $R(A,B)$). We call $\mathcal{X}$ the set of nodes that are both keys of O_A and I_B; to approximate PageRank, we will take in consideration random walks that start at A and finish at B passing through a node in $\mathcal{X}$. Recalling equation 4.3 and the fact

Algorithm 2 Outgoing compact map construction for PageRank.

Parameters: the desired size k of the neighborhood map.
Return value: a dictionary O_A such that $O_A[X]$ contains an approximation of $R(A, X)$.

On each node A, execute the following code:

$d_A \leftarrow \sum_{Y \in N(A)} w_{AY}$ /* *A's "weighted degree"* */

while values have not yet converged **do**

$O_A[A] \leftarrow 1 - \alpha$

for all $Y \in N(A)$ **do**

Obtain O_Y from Y

for all keys X of dictionary O_Y **do**
if $X \notin O_A$ **then**
$O_A[X] \leftarrow \alpha \frac{w_{AY}}{d_A} O_Y[X]$
else
$O_A[X] \leftarrow O_A[X] + \alpha \frac{w_{AY}}{d_A} O_Y[X]$
end if
end for
end for

Keep in O_A only the k elements corresponding to the largest values.

end while

return O_A

Algorithm 3 Incoming compact map construction for PageRank.

Parameters: the desired size k of the neighborhood map.
Return value: a dictionary I_B such that $I_B[X]$ contains an approximation of $R(X, B)$.

On each node B, execute the following code:

for all $Y \in N^{-1}(B)$ **do**
  $d_Y \leftarrow \sum_{Z \in N(Y)} w_{YZ}$ /* *Y's "weighted degree"* */
end for

while values have not yet converged **do**

  $I_B[B] \leftarrow 1 - \alpha$

  for all $Y \in N^{-1}(B)$ **do**

    Obtain I_Y from Y

    for all keys X of dictionary I_Y **do**
      if $X \notin I_B$ **then**
        $I_B[X] \leftarrow \alpha \frac{w_{YB}}{d_Y} I_Y[X]$
      else
        $I_B[X] \leftarrow I_B[X] + \alpha \frac{w_{YB}}{d_Y} I_Y[X]$
      end if
    end for

  end for

  Keep in I_B only the k elements corresponding to the largest values.

end while
return I_B

that $O_A[X]$ and $I_B[X]$ approximate respectively $R(A, X)$ and $R(X, B)$, we compute the approximated PPR value $\hat{R}(A, B)$ as

$$\hat{R}(A, B) = \frac{1 - \alpha}{\alpha} \sum_{X \in \mathcal{X}} O_A[X] I_B[X]. \tag{4.6}$$

Using this method, and differently from non-compact maps, walks that pass through different nodes of $\mathcal{X}$ are accounted for more than once. The impact of this factor on accuracy will be examined in the experimental results.

4.4 Security Issues

Obviously, a system that distinguishes cooperative and malicious behavior is useful only if it is itself immune from tampering by malicious users. Here we will consider two kinds of attacks: forged maps, where neighborhood maps do not reflect the web of trust, and flooding attacks, where the attacker fills the web of trust with useless information to make neighborhood maps less efficient.

4.4.1 Forged Maps

In order to obtain higher ranking, or in general to subvert the regular evaluation of reputation metrics, malicious nodes may be interested in communicating forged data to their peers during the map creation step.

Nodes reachable via outgoing paths, or in compact maps the "outgoing" sub-maps, are not really problematic because each node is actually supposed to control the outgoing paths in the web of trust. The opposite is true for incoming paths: for example, a node may declare that it is trusted by reputable nodes just to obtain a higher reputation value. Depending on the actual scenario in which neighborhood maps are used, various solutions can be adopted.

Use an undirected web of trust. We assume to work with a directed graph, where an edge $A \to B$ implies that A trusts B, but the opposite is not necessarily valid. If the network is undirected, then an edge between A and B implies mutual trust between the two users. If this approach is used, some information has to be thrown away. However, in many cases, trust is reciprocal: in the OpenPGP web of trust discussed in section 4.5.1, if an edge $A \to B$ exists, the reverse edge $B \to A$ exists as well in 68.6% of the cases.

Signed links. If compact maps are not used, node identifiers can be public cryptographic keys and links can be signed. The cost of this solution is paid with storage space and verification of signatures.

Delegate the construction of maps. If a DHT is available, the maps for node n can be computed by k nodes identified by a family of hash functions $h_1(n), \ldots, h_k(n)$. Redundancy solves the problems generated by offline nodes, and discrepancies are resolved with voting. This idea has been introduced for EigenTrust [KSGM03].

Ask confirmation from origins of trust links. Another option is just to verify that trust paths that have been found actually exist, by asking the authors of the relevant links that are found in standard maps. If compact maps are used, $I_A[X]$ and $O_X[A]$ should approximate the same value $R(A, X)$. Thus, "incoming" maps can be used only to find the intersection nodes $\mathcal{X}$, but equation 4.6 can be modified to only use the more secure values from "outgoing" maps:

$$\hat{R}(A, B) = \frac{1-\alpha}{\alpha} \sum_{X \in \mathcal{X}} O_A[X] O_X[B]. \tag{4.7}$$

4.4.2 Flooding the web of trust

The flooding attack is an instance of collusive/Sybil attack, resulting in a denial of service: a high number of fake nodes are created, and they all give high ranking to the attacked node. In this way, the neighborhood map of the target becomes full of references to insignificant nodes, and thus useless.

To avoid this kind of attack, nodes can give precedence to nodes they themselves trust (i.e., there are outgoing paths that reach them in the web of trust). Since Sybils will have low reputation, they will not be taken in consideration. For "standard" maps, this can be accomplished simply by visiting only outgoing paths. Since social networks are highly clustered, most short incoming paths are found anyways.

In compact maps, instead of cutting the I_A map to the k elements with highest reputation, a priority value $P(A, X)$ is defined for each pair of nodes (A, X):

$$P(A, X) = \begin{cases} O_A[X] \cdot I_A[X] & \text{if } X \in O_A, \\ \min_{Y \in O_A} (O_A[Y]) \cdot I_A[X] & \text{otherwise.} \end{cases} \tag{4.8}$$

A node X is kept or cut from neighborhood maps depending on this priority value. In this way, highly reputable nodes have a higher precedence and are still kept in neighborhood maps even in the presence of flooding. As defined, the priority value for Personalized

PageRank reflects the approximated probability that a random walk starting from A gets to X and then to A again. The first part of the path punishes Sybils, since they will be reached by random walks with low probability.

4.5 Experimental Validation

In this section we validate the technique proposed, both on real data from the OpenPGP web of trust and synthetic data from our model of chapter 3. We evaluate how the approximated ranking produced using neighborhood maps is different from the exact one; using a simulation based on evolutionary games, we show that neighborhood maps provide incentives that are sufficient to foster cooperation without requiring full knowledge of the web of trust.

4.5.1 The OpenPGP Web of Trust

In order to evaluate our algorithm on significant data, we have chosen the OpenPGP web of trust as our test case. We feel data from this network can be more significant to our purposes than other social networks available on the network (e.g., the FOAF project[4]), since it represents a network where edges represent *trust*, rather than simple acquaintance.

OpenPGP [CDFT98] is an open protocol for privacy and authentication, using asymmetric key cryptography. There is no central certification authority binding persons to their respective public keys: the users themselves sign the keys of other users (usually, after a real-life meeting) to attest their correspondence to an identity. The set of such information forms the web of trust defined as the directed graph having keys as nodes and trust relationships as edges. Based on (part of) the data in the web of trust, users decide whether to trust or not the authentication of other users.

We are using data taken from the strongly connected component of the OpenPGP web of trust on May 9, 2005[5]. It is a scale-free graph having 27,398 nodes and 246,355 edges (with an average of 8.99 outgoing edges per node). Figure 4.2 shows its degree distribution. The graph is a small-world network, with a mean shortest distance between nodes of 5.96, and a clustering coefficient (according to the C_1 definition of equation 3.5 at page 41) of 0.371.

In figure 4.3, the distribution of distances of this network is plotted against the one of a $G(n, m)$ random directed network with the same number of nodes and edges (mean shortest distance 4.89). The random one has a much narrower shape, due to the fact

[4] `http://www.foaf-project.org/`

[5] Daily dumps of the whole OpenPGP web of trust can be retrieved from `http://www.lysator.liu.se/~jc/wotsap/`.

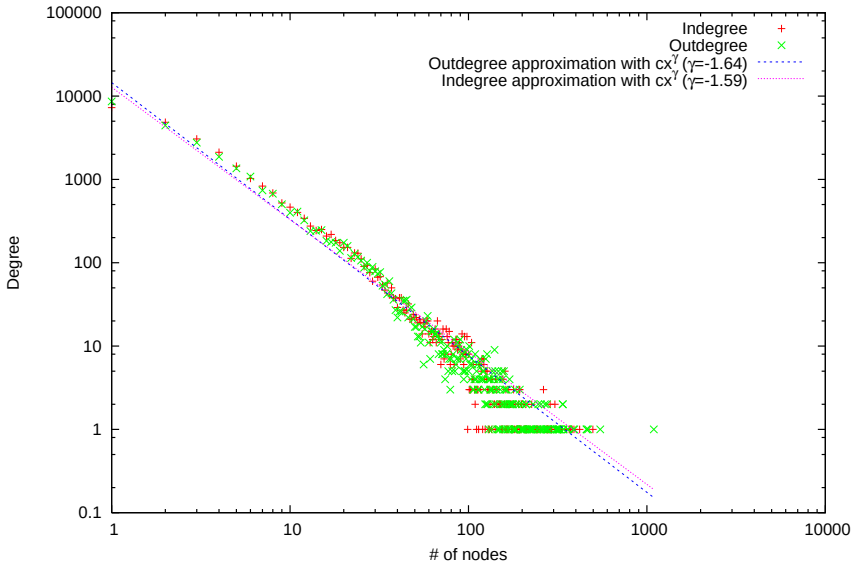


Figure 4.2: OpenPGP degree distribution. Both in-degree and out-degree follow a power law.

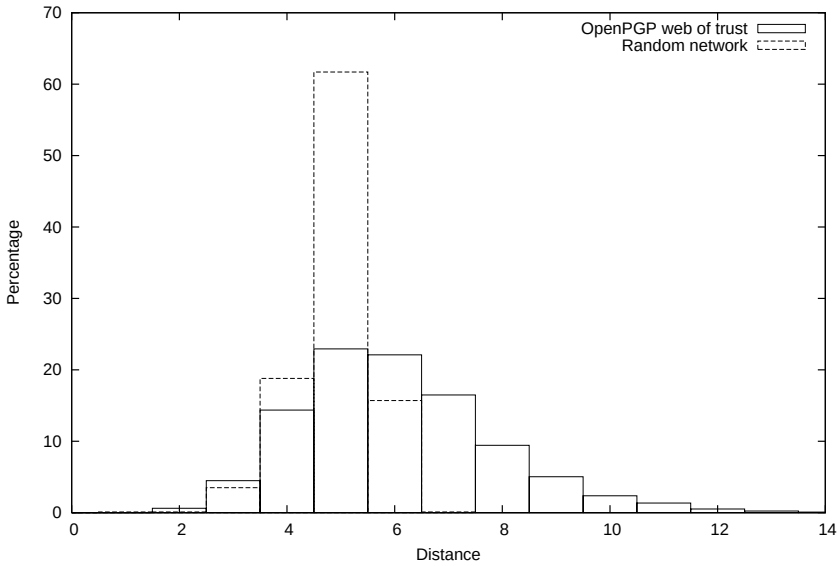


Figure 4.3: OpenPGP distance distribution.

that "peripheral" nodes with very few links in the power-law network are less likely to be connected by short paths.

The low average distance is crucial, since we may hope to have an intersection between maps of different nodes only if they are not "too far away" on the network. The power-law degree distribution has a beneficial effect for the nodes that are at short distance from a hub, since they are likely to be found in many different maps; the "peripheral" nodes that have a higher average distance, on the contrary, will suffer their isolation due to their low degree.

High clustering poses us some problems, since "close" pairs of nodes will have more intersections in the neighborhood maps, and "far" ones will have fewer, resulting in a lower probability of finding intersections in more difficult cases.

4.5.2 Shortest Path Length

To study how the estimation of shortest path length evolves with respect to the map size parameter k, we analyze:

1. how the probability of finding a trust path (i.e., having a non-empty intersection between maps) evolves;
2. what is the error in the estimation of distance.

Non-Empty Intersections If a non-empty intersection between two neighborhood maps exists, then a path has been found, and thus an upper bound on the distance between nodes is established. As shown in the example of section 4.3.1, however, the path found is not necessarily the shortest one.

In figure 4.4, the probability of encountering a non-empty intersection in two neighborhood maps is shown against the size of the maps. The results are obtained after running the experiments for 10,000 random pairs of nodes, and verifying the size needed to obtain non-empty intersections in all cases.

While the curves for the random network and the theoretical prediction for random maps have a similar shape, the curve for the scale-free small-world networks (the PGP web of trust and our social network model) significantly deviate from the other two; we speculate that their structure has two important effects.

- It is likely that two different nodes are connected (or at short distance from) the same hub, thus making it significantly easier to find a node in common between the two maps.

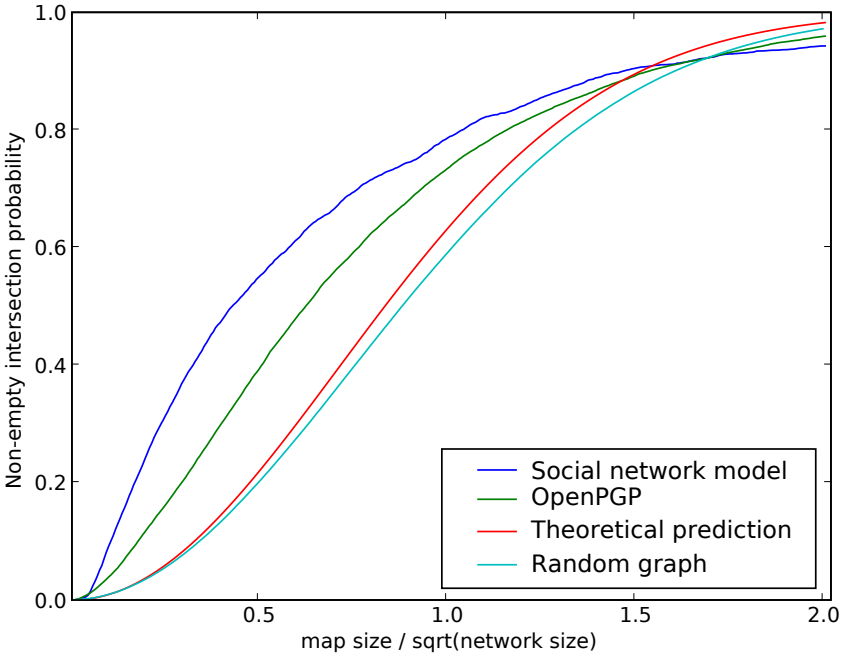


Figure 4.4: Non-empty intersection probability on three different networks with respect to the theoretical prediction of section 4.2: the small-world scale-free social network model from chapter 3 (parameters: $\alpha = \sigma = \infty, \gamma = 1, n = 27,398, m = 5$), the OpenPGP web of trust and a $G(n, m)$ random graph with the same number of nodes and edges as the OpenPGP network.

- Nodes that are harder to reach belong to highly clustered sub-communities; the information in neighborhood maps is more redundant, since most nodes in them belong to the same community.

In general, the benign effect of hubs appears stronger than the negative influence of clustering; in networks having more than 27,000 nodes, maps with size $2\sqrt{n} \simeq 330$ are usually big enough to find trust paths.

Approximation When no intersection is found between two neighborhood maps, a guess has to be made anyway; since we are working with small-world networks, a low estimation is likely to be correct. Thus, we use the shortest path length outside the range that can be computed using the neighborhood maps. For example, let us suppose that nodes A and B find no intersection in their neighborhood maps, the farthest node in the map of A has distance 3 and the farthest one from B has distance 4. The estimated distance between A and B is then 4+3+1=8.

Figure 4.5 shows how the accuracy of our shortest path length estimation progresses with respect to the neighborhood map size; the relative error is averaged over 10,000 runs. In networks with high clustering, the accuracy achieved is lower than in random networks; however, the relative error drops exponentially when the map size increases. In our results, maps of size $2\sqrt{n}$ are enough to obtain a 1% error; when the map size becomes $6\sqrt{n}$, the error rate drops below 0.01%.

4.5.3 Personalized PageRank

To verify how efficient our approximation of Personalized PageRank (PPR) is, we compared the rankings produced by it on the whole web of trust and when using compact maps. To evaluate the similarity of rankings, we computed Kendall's τ correlation coefficient [Ken38, Abd07] between PPR and our approximation. Given two random items and two different rankings, $P(\text{same})$ is the probability that the two items are ranked in the same order; $P(\text{different})$ is the probability that they are ranked instead in reverse order. The correlation coefficient τ is defined as

$$\tau = P(\text{same}) - P(\text{different}).$$

Two equivalent rankings will therefore have $\tau = 1$, while $\tau = -1$ for completely discordant rankings.

Impact of network topology Figure 4.6 plots the correlation between PPR and its approximation computed with neighborhood maps. The results suggest that ranking is

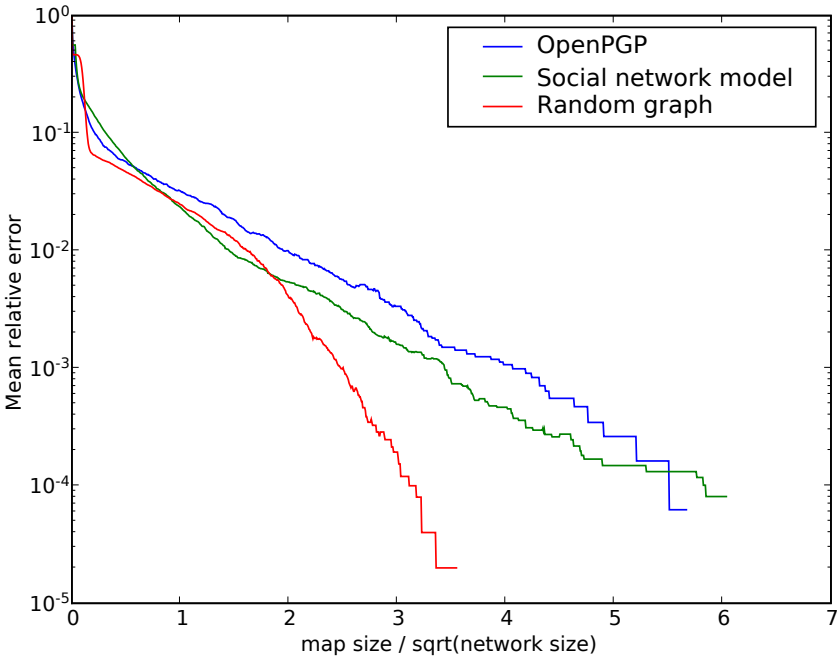


Figure 4.5: Relative error plot. The test networks are the same of figure 4.4.

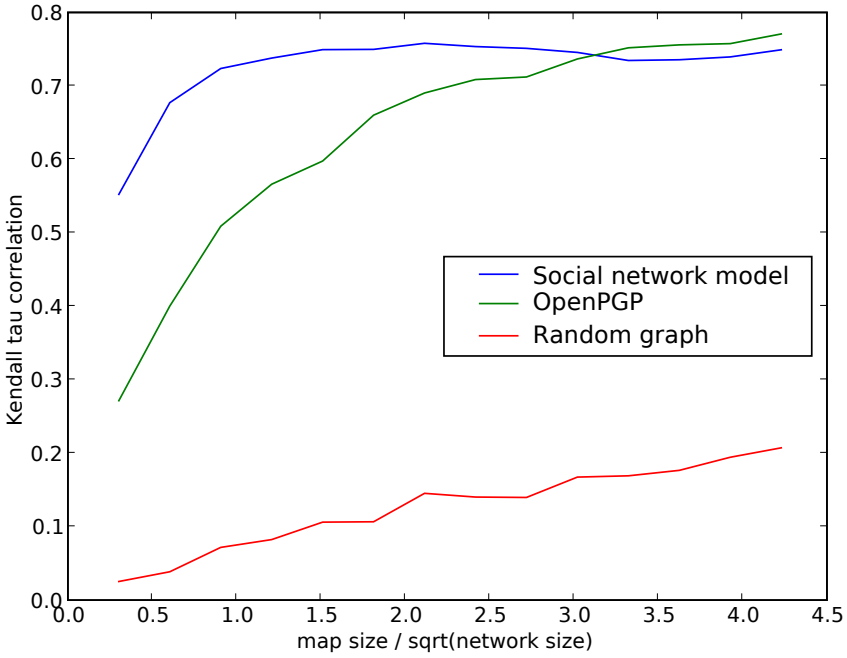


Figure 4.6: Kendall's τ correlation between Personalized PageRank and our approximation versus map size for the networks of figure 4.4. In each run of the experiments, a random node ranks 100 random peers; results are averaged over 100 runs.

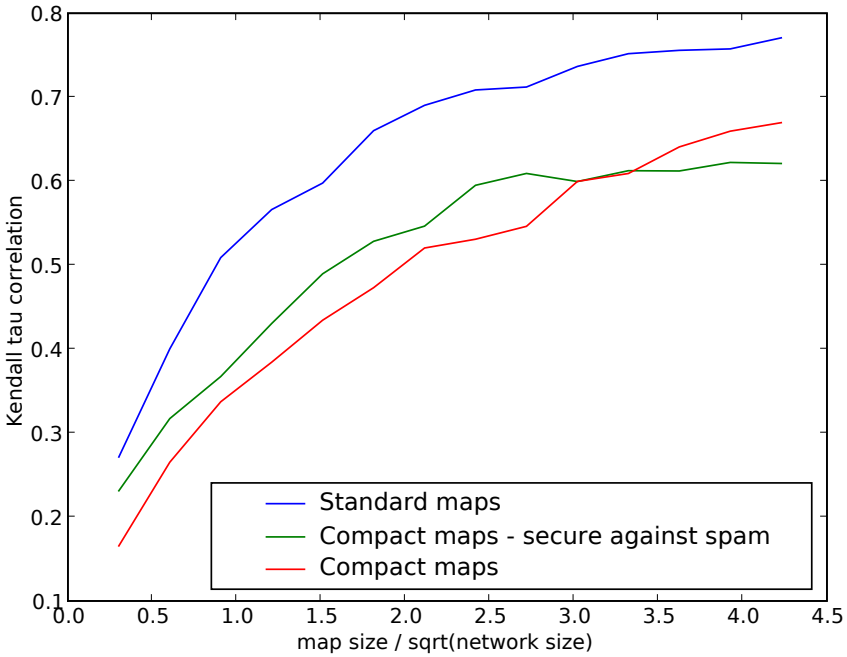


Figure 4.7: Kendall's τ correlation between PPR and our approximation using standard and compact maps. The "secure" version uses the priority values defined in equation 4.8 to defend against flooding on the web of trust.

much easier to compute in power-law networks than in regular random networks. We justify this result by noting that the PageRank values of a scale-free network follow a power law as well [PRU02]; a random network has instead a much narrower distribution of ranking values, so finding which peers have higher ranking is harder.

When maps are small, PageRank appears easier to approximate in our synthetic network model than in the OpenPGP web of trust. We motivate this with the fact that our model creates undirected networks, while the OpenPGP web of trust is a directed graph. This latter case makes small maps less efficient, since nodes that can be reached via a visit of a network (short outgoing paths) are not necessarily those that can rapidly reach the destination (short incoming paths).

Compact Maps Compact maps require less communication overhead since they require a single real value for each node instead of an adjacency list, and the approximated PageRank value can be computed more rapidly simply by computing the sum defined in equation 4.6 at page 55. The tradeoff is that some information gets thrown away, and the PPR

approximation becomes noticeably less precise; for this reason, it appears advisable to use compact maps only when the computational resources are severely limited.

In figure 4.7, we compare the precision using "standard" maps that represent the full subgraph and compact maps. The tradeoff is clearly visible; in the absence of malicious attack, the quality of approximation is not strongly affected by the use of priority values defined in equation 4.8 at page 56 against flooding on the web of trust.

4.5.4 File-Sharing Simulation

In order to show how neighborhood maps encourage cooperation, we developed a simulation, mimicking the one that Feldman *et al.* used to validate their MaxFlow algorithm [FLSC04]. It is meant to simply simulate what happens in a file-sharing application, associating a degree of satisfaction (*payoff*) for each node to the quality of service that they receive from the network.

Each node has an associated pattern of behavior (*strategy*) that specifies how it will respond to service requests from other nodes. During time, nodes are able to *learn*, by collecting statistics about which strategies obtain better payoffs, and switch strategies accordingly. The best performing strategy is found by evaluating the aggregate running average payoff for all peers according to their current strategies.

Time is divided into rounds. During each round, each node generates a fixed number m of requests that will be forwarded to m random peers; it decides, according to its strategy, whether to satisfy at most one requesting peer; with probability p_l it *learns*, by switching to the best performing strategy; with probability p_t there is a *turnover*: it leaves the network and a new peer, with the same strategy and a fresh history, joins it.

The history of the network is expressed in a web of trust where the weight w_{ab} for an edge from peer a to b represents the number of requests from b that were satisfied by a. The payoff for a peer a is defined as

$$\sum_n (7 \cdot w_{na} - w_{an}) .$$

This reflects the fact that the benefits for the nodes behaving as clients far outweighs the costs for the "servers".

The available strategies are the following:

Cooperator. Gives service at each turn to a random requesting peer.

Defector. Never satisfies any request.

Reciprocator. Ranks nodes according to the payoff change due to them (i.e., node a's ranking of b is $7 \cdot w_{ba} - w_{ab}$), and then satisfies the request coming from the highest rated peer.

Neighborhood Map Reciprocator: uses the PageRank approximation to rank peers; the highest rated peer is served at each step. For efficiency, we chose to use compact maps.

Neighborhood maps for nodes using the last strategy are constructed gradually: at the end of each round, an iterative step of algorithms 2 and 3 is performed. When a peer does not have an available neighborhood map (this happens because it is adopting a different strategy), the minimal starting map assigning value $1 - \alpha$ to itself is taken by default.

In agreement with Feldman *et al.*, we observed that the Reciprocator strategy is successful when the network is small enough; as the network grows, the probability of meeting many times the same node becomes small, and direct reciprocation becomes unfeasible. The Neighborhood Map user, instead, uses information propagated from other nodes in order to build its own view of the network. During the first iterations, the Defector strategy performs better, since the reciprocators do not have yet enough information to discover which nodes have an altruistic behavior. Anyway, when they obtain enough information, they become able to effectively reciprocate.

An instance of a situation where neighborhood maps are decisive in ensuring reciprocation is shown in figure 4.8. The parameters are $n = 500$ nodes, $m = 3$ requests per round, learning probability $p_l = 0.05$, and turnover probability $p_t = 0.0001$. Neighborhood maps are built with PageRank parameter $\alpha = 0.85$, and size $k = 67 \simeq 3\sqrt{n}$. On the left, the available strategies are Cooperator, Defector, and Reciprocator. On the right, Reciprocator is substituted by Neighborhood Map Reciprocator. The direct reciprocative strategy is not successful enough given these parameters, because the probability of meeting an unknown node is too high. In this case, Defector is the most successful strategy. On the right, the neighborhood maps approach does not work well for the first interactions, when knowledge of past node behavior is not sufficient, but in the following leverages on the information in neighborhood maps in order to build successful reciprocative patterns.

As shown by these results, neighborhood maps effectively encourage cooperation in situations where direct reciprocation is not sustainable, similarly to the approaches that require global knowledge [FLSC04]. However, approaches like neighborhood maps are more scalable, since they only require a small fraction of the resources needed to obtain global knowledge.

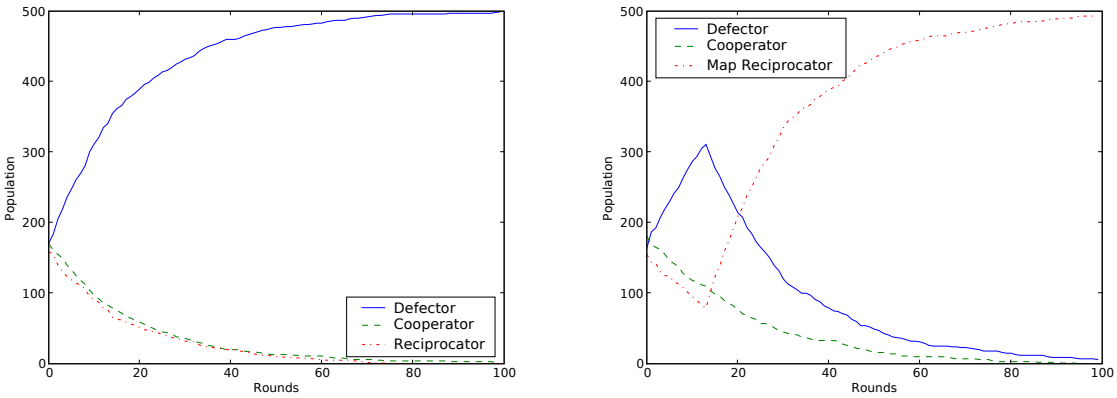


Figure 4.8: File-sharing simulation: evolution of population for each strategy.

Chapter 5

Mapping Small Worlds

In this chapter[1] we approach the question of navigability of social networks already briefly introduced in section 2.1.2. We start from the intuition used in chapter 3 to model social networks: the highly clustered structure of social networks is due to an underlying topology that reflects affinity between nodes. Our goal, here, is building a "network map" that reflects as closely as possible that topology.

The network map is an assignment of d coordinates to each node of the network, placing the most affine nodes closest. The map can be used to "navigate" through the network in the same way that the lattice coordinates are used in the Kleinberg model [Kle00a], finding short paths that connect pairs of nodes: given the coordinate of the destination, navigation proceeds from node to node by choosing at each step the neighbor closest to the destination, in terms of Euclidean distance on the map.

We provide an algorithm which is able to build, in a completely decentralized way (i.e., each node exchanges information only with its neighbors), a network map which is consistently helpful for the navigation of social networks.

Navigability is useful for computing reputation since it makes it easy to find the trust paths discussed in section 2.3.1; in chapter 7, we will introduce current work on integration between network maps and neighborhood maps to discover relevant subsets of webs of trust, in order to approximate reputation metrics even more efficiently.

Network maps can also be used to implement routing on social networks that can be used in "darknets", that is, anonymous networks where nodes establish connections only if they are mutually trusted. In the Freenet anonymous network, an analogous algorithm is used to perform routing on the social network [San06]; our experimental results show that, in

[1] This chapter is based on a paper presented at the seventh IEEE Conference on Peer-to-Peer Computing [Del07].

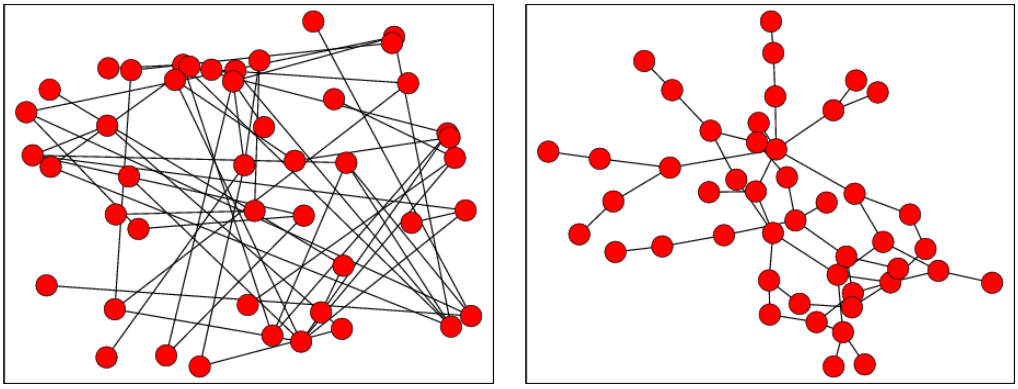


Figure 5.1: The same network, represented by positioning nodes randomly (left) and using a graph drawing algorithm (right).

the OpenPGP web of trust, the median length for shortest paths between pairs of random nodes is 4, and the existing Freenet algorithm is able to discover paths of median length 57. Our algorithm is able to discover paths of median length 6, and using a simple neighbor-of-neighbor optimization the path length becomes even closer to the optimal value: the median, in fact, falls to 4.

5.1 Network Maps

As we have already discussed in the previous chapters, many peer-to-peer applications are deeply related with social networks. Besides webs of trust, which are the primary focus of this thesis, other examples are the "data-sharing" [IRF04, RSG06] networks connecting consumers of the same content in file-sharing, or the social network in anonymous networks, where peers forward their requests to trusted nodes in order to decrease their traceability.

The intuition behind network maps comes from the idea that a good layout carries information about the structure of the network. Compare the two images in figure 5.1: they represent the same network, but paths between nodes are much more easily found by looking at the more "untangled" layout on the right. A good map is a tool that makes navigation easier.

Creating a network map means assigning a set of d coordinates to each node of a network, placing them in a d-dimensional space. As benchmark to evaluate the validity of the layout, we evaluate its effectiveness in making the network navigable through greedy routing, using the approach of Kleinberg. In our context, we consider "better" maps those that allow users to find shorter paths.

Network maps, however, are not only useful for "routing": we believe that the layout

we obtain reflects information about the intimate structure of the network; as such, this information could be used to analyze the network itself. Euclidean distance can be used as a metric to represent affinity between node. In addition to the use of network maps to discover paths in webs of trust, we believe that network maps can be used in various other fields.

- In "darknets" (also known as "friend-to-friend" networks) such as the one present in Freenet [CMH+02]: in order to enhance privacy, nodes connect only to trusted parties, and messages are routed on the social network reflecting mutual trust between peers. Our algorithm obtains considerably better results than the existing Freenet algorithm [San06].
- In networks that associate favorite items or products to users, the map can be used to recommend items of interest. Web search results could be personalized by assigning a higher ranking to nodes that are deemed as "close" to the interests of the requester. In chapter 7, we will discuss the application of network maps to the problem of information filtering.
- Network maps could also be used in "tag clouds" [CLP07], being applied to the network that connects tags if they have been used for the same item. Such an analysis could help in understanding how tags are related (for example, when tags can be considered as synonyms, and whether "subset" relationships exist).

To assign reasonable layouts, we take inspiration from algorithms for graph drawing, and adapt them to our case. The contribution introduced in this chapter can be summarized in three points:

1. We calculate network maps using Koren's graph drawing algorithm [Kor03] based on spectral graph theory, and show that this solution provides efficient routing.
2. We modify the algorithm to obtain scalability and decentralization, with no relevant loss in routing efficiency.
3. We analyze the improvement obtained by giving to each node information about the position of neighbors of their neighbors, and planning routing accordingly.

5.2 Related Work

This work creates a connection between two fields that, according to the knowledge of the author, have not been connected by previous work: the study of routing in complex

networks, and algorithms for automated graph drawing. This section will briefly introduce related work in both fields.

5.2.1 Small Worlds and Navigability

After Jon Kleinberg (see section 2.1.2), showed that small-world networks are not necessarily *navigable* networks, a significant amount of work has been devoted to the analysis of the navigability property.

Navigability of social networks has been explained by the fact that people are more likely to be connected to "similar" persons, that is, people who live in the same area or do the same work [WDN02, DMW03]. Since people in the experiment of John Milgram were forwarding the letter to nodes that were more and more affine to the destination, the probability that those nodes knew the target was growing higher.

Kleinberg proposed a simple geographical interpretation for the position of nodes in his model; while geographic position has been demonstrated to be the most useful piece of information for routing on social networks [LNNT05], it can be argued that other information can be used to achieve even better navigability. Moreover, in P2P applications deployed on the Internet, the influence of geographic distance in discouraging interactions is low. While in many cases geographic clustering probably happens anyway, it can be a byproduct of other relevant factors, such as language, culture or personal interests.

Models such as the one developed by Kleinberg start from a layout that reflects affinity between nodes and create plausible social networks. We focus on the reverse problem: we start from a real social network and create a plausible layout reflecting node similarity.

Oskar Sandberg [San06] developed an algorithm that performs routing in the Freenet darknet. Based on Kleinberg's work, a Metropolis-Hastings Markov chain algorithm places nodes in a grid, with the goal of minimizing the edge length. The algorithm works by iteratively selecting pairs of random nodes and considering whether to switch the position of the two nodes; a stress function based on edge lengths is calculated, and the nodes are switched positions according to the outcome of that stress function. Experimental results are obtained by applying the algorithm on the OpenPGP web of trust, which we already introduced in section 4.5.1.

In this chapter, we will apply our algorithm on the same network and compare the results of our algorithm with the ones obtained by using Sandberg's algorithm.

5.2.2 Graph Visualization

Given a social network, the ideal layout would place nodes in such a way that the number of hops that separates any two nodes is directly proportional to their distance on the layout. In other words, nodes at distance 1 are always connected, nodes at distance 2 always have a common neighbor, and so on. Unfortunately, such a layout could need $n - 1$ dimensions for a network with n nodes; this would be obviously untreatable for large-scale dynamic networks.

Our goal is to obtain a layout with a more manageable number of dimensions, where most edges connect nodes at a short distance: in other words, the probability that two nodes are linked decreases as the distance between those nodes increases.

While this approach gives no warranty of any relation between distance in the layout and distance in terms of number of hops, the experimental results show this provides a very useful heuristics.

The algorithm we propose is borrowed from graph drawing techniques. Many graph drawing algorithms try to obtain a visually pleasing layout by positioning connected nodes at short distance, while separating as much as possible the others. These objectives are perfectly in line with the desired properties of our layout.

Many well-known algorithms for graph visualization are based on a "spring model" [KK89]: repulsive forces pull nodes apart, and attractive forces bind linked nodes together. These algorithms are unfortunately difficult to implement on P2P networks, since each iteration of the algorithm is $O(n^2)$, and requires each node to exchange information about its position with each other node in the network.

More efficient algorithms for creating layouts are based on spectral properties of graphs. The algorithms that employ this kind of technique work on matrices that are related to the adjacency matrix of the graph, and use data from the eigenvalues of such matrices. It is possible to leverage on the sparseness of such matrices and on power iteration methods in order to obtain efficient implementations. We will adapt an algorithm of this class, which is described in the next section , to our requirements.

5.2.3 Koren's Spectral Layout Algorithm

This section will briefly introduce the degree-normalized algorithm for spectral graph drawing devised by Koren. For a more thorough explanation, see [Kor03]. While the algorithm works also on weighted graphs, here for simplicity we will refer to the unweighted case.

The one-dimensional problem is formalized as a linear optimization problem for a graph

$G = (V, E)$, where the objective is to obtain a vector x that minimizes a stress function

$$E(x) = \sum_{\langle i,j \rangle \in E} (x_i - x_j)^2$$

given the constraints

$$\sum_{i \in V} k_i x_i = 0, \qquad \sum_{i \in V} k_i \left(x_i\right)^2 = 1,$$

where k_i is the degree of node i. The coordinate attributed to node i for its layout on the x axis will then be x_i.

Minimization of $E(x)$ ensures that nodes connected by an edge will not be placed far apart in the layout; the constraints warrant that the layout is centered around 0 and that the layout does not collapse on a single point.

To obtain a bi-dimensional layout, we need to make sure that the x and y vectors are not correlated, thus providing as much information as possible; this is achieved by imposing an additional constraint

$$\sum_{i \in V} k_i x_i y_i = 0$$

when calculating the y vector. This can be seen as a "weighted orthogonality" requirement, and the procedure to ensure it is satisfied will be called "degree-normalized orthogonalization", or "D-orthogonalization". For multi-dimensional layouts, such constraint must hold for each vector against all previously computed vectors.

Koren shows that the solution to the problem can be obtained by using power iteration to find the eigenvectors u_i that satisfy

$$\frac{1}{2}\left(I + D^{-1}\right) A u_i = \lambda_i u_i$$

where I is the identity matrix, D is the diagonal degree matrix where $d_{ii} = k_i$ and $d_{ij} = 0$ if $i \neq j$, and A is the graph's adjacency matrix.

We label the $\langle \lambda_i, u_i \rangle$ eigenpairs so that $\lambda_1 > \lambda_2 \ldots$; the first eigenpair is always $\langle \lambda_1 = 1, u_i = 1^n \rangle$. The optimal solution to the d-dimensional problem is obtained via the $u_2, \ldots, u_{d+1}$ eigenvectors: the d coordinates that should be attributed to node i are $u_2(i), \ldots, u_{d+1}(i)$.

The pseudo-code for this algorithm is shown in algorithm 4. The inner loop of the algorithm (lines 13-30) has a surprisingly simple intuitive explanation:

Orthogonalization: avoids converging towards already computed values of relevant eigenvectors;

Multiplication: each node converges towards the center of its neighbors, in order to minimize distances between nodes connected by edges;

Normalization: the layout is "exploded" to ensure it does not collapse to a single point.

As with all power iteration algorithms, the convergence is geometric, with ratio $\frac{\lambda_i}{\lambda_{i+1}}$ for each u_i eigenvector. The convergence speed thus depends on the spectral characteristics of the social network.

The u_i vectors also satisfy the equation $Lu_i = \mu_i D u_i$, with L being the Laplacian matrix $D - A$, and $\mu_i = 2\,(1 - \lambda_i)$. Vectors satisfying such equation are very well known in spectral graph theory. In eigenpairs $\langle \mu_i, u_i \rangle$ with i small and greater than 1, the value of $u_i(j)$ is close to $u_i(k)$ if the nodes j and k belong to the same cluster [SM00]. Nodes belonging to the same community will thus be placed close in our layout.

5.3 Decentralized Implementation

The algorithm by Koren is projected to be executed on centralized systems and for reasonably small networks; we need to implement algorithm 4 so that it can be executed efficiently on a decentralized system, by requiring that each node needs to communicate only with its neighbors on the web of trust.

Notice that the core step of matrix multiplication (lines 23-26) is already decentralized: each node needs to know only the position of its neighbors in order to compute the multiplication step. Hence, the two parts that need to be modified are normalization (line 28) and D-orthogonalization (lines 15-21), as discussed in the next two sections.

5.3.1 D-Orthogonalization

The orthogonalization step is executed to make sure that the computation does not converge towards the trivial solution $u_1 = 1^n$, or towards the already computed $u_2, \ldots, u_{i-1}$ vectors when computing the u_i vector.

The power iteration method is based on the property that, starting with a vector that can be expressed as some linear combination $x_1 u_1 + \ldots + x_n u_n$ of the eigenvectors, the result of applying t iterative steps will be

$$\lambda_1^t x_1 u_1 + \ldots + \lambda_n^t x_n u_n.$$

Since $\lambda_1 > \lambda_2 > \ldots > \lambda_n$, the computation will converge in direction towards u_1 if $x_1 \neq 0$, towards u_2 if $x_1 = 0$ and $x_2 \neq 0$, and so on. For this reason, orthogonalization

Algorithm 4 Koren's layout algorithm.

Parameters: a graph $G = (V, E)$, the desired number of dimensions d.
Return value: a d-ple $u_2 \dots u_{d+1}$ where u_i represents the coordinates of each node for the $i-1$-th dimension.

$u_1 \leftarrow 1^n$ /* *First eigenvector* */

for $i \in 2 \dots d+1$ **do**
  /* *Calculating the* u_i *eigenvector* */

  for all $v \in V$ **do**
    $u_i \leftarrow$ random /* *random initialization* */
  end for

  while algorithm has not converged **do**

    for $j \in 1 \dots i-1$ **do**
      /* *Orthogonalize* u_i *against* u_j */
      $c \leftarrow \frac{\sum_{v \in V} k_v u_i(v) u_j(v)}{\sum_{v \in V} k_v (u_j(v))^2}$
      for all $v \in V$ **do**
        $u_i(v) \leftarrow u_i(v) - c \cdot u_j(v)$
      end for
    end for

    for all $v \in V$ **do**
      /* *Multiply by* $\frac{1}{2}(I + D^{-1}A)$ */
      $u'_i(v) \leftarrow \frac{1}{2}\left(u_i(v) + \frac{\sum_{(v,v') \in E} u_i(v')}{k_v}\right)$
    end for

    $u_i \leftarrow \frac{u'_i}{\|u'_i\|}$ /* *normalization* */

  end while
end for

return $u_2 \dots u_{d+1}$

is only really necessary at the first step of iteration, and is used afterwards only in order to improve numerical stability. We found, using double-precision floating point numbers, that the orthogonalization step was not necessary in our experiments.

It is reasonably easy to obtain a starting vector that is D-orthogonal to $u_1 = 1^n$: this happens if

$$\sum_{v \in V} k_n u_i(v) = 0.$$

We can obtain this by enforcing this rule: each node v_1 tries to find another (yet unpaired) random node v_2. The

$$k_1 u_i(v_1) + k_2 u_i(v_2) = 0$$

property can be ensured by negotiating a random pair $(r, -r)$ and having

$$u_i(v_1) = \frac{r}{k_1}, u_i(v_2) = \frac{r}{k_2}.$$

If a node is unable to find a random pair, it can start with a value of 0. If more efficient means are not available (e.g., a random look-up on a DHT), a reasonable way to find a random node is performing a long enough random walk on the social network. It appears that social networks are fast mixing [YKGF06], so a walk having length of $\Theta(\log n)$ should be enough.

The computation of the first eigenvalues using algorithm 4 on the OpenPGP dataset leads to eigenvalues μ_i that are very close to zero: executing 200 steps of the power iteration, the computed approximations of the first 10 eigenvalues are all between 0.007 and 0.012 (the respective λ_i values are thus between 0.994 and 0.9965).

The fact that many eigenvalues are close to 0 can be interpreted in two ways. Firstly, by analogy with the graph-drawing algorithm this seems to suggest that the network is inherently multi-dimensional. In fact, two individuals can be connected for many reasons: for example, they could live close, they could listen to similar music or they could do the same work. Each of those characteristics can be expressed by a measure of closeness that is independent from the others, and thus needs another dimension.

Another interpretation surges from the fact that social networks are composed of many different clustered sub-communities. For instance, it has been shown that users having email addresses corresponding to certain domains such as `.it` or `.debian.org` are far more likely to be linked [Ced04]. The presence of clusters in a network yields eigenvalues close to zero in the Laplacian matrix [Chu97].

A consequence of having the μ_i values close to zero is that the convergence using the power iteration method is very slow. This means that, after a reasonable number of iterations (200 in our experiment), the result, instead of converging fast towards the desired value, becomes a linear combination of the most relevant eigenvectors. This approximate solution,

however, turns out to be a reasonable solution for our problems; moreover, different starting vectors will result in very different results.

Since the goal for orthogonalization in the first place was to obtain different solutions for the computed vectors, that means we can just drop the step from the decentralized algorithm. Since the orthogonalization step is quadratic with respect to the number of dimensions, while the rest of the algorithm is linear, this means we gain the possibility of obtaining a higher dimensionality.

Should the orthogonalization step be needed anyway, it would be possible to compute D-covariance by computing the weighted averages of $k_j u_i(v_j) u_k(v_j)$ and $k_n u_i(v_j)^2$, as discussed in section 5.3.2.

5.3.2 Normalization

The normalization step is used for two reasons:

1. During computation, the size of the vector being computed can expand (in case the eigenvalue $\lambda_i > 1$) or shrink ($\lambda_i < 1$) to a size not in the range of the floating point representation (in our case, the λ_i are guaranteed to be less than 1).
2. At the end of computation, normalization is used in order to make the eigenvectors of the same length. In our case, if a vector u_i is much longer than any of the others, the distances on that vector would prevail over distances on others, hiding the benefits of using a multi-dimensional representation.

Since the λ_i values in our experiments were very close to 1, the values in the vector were well within the boundaries for floating point numbers, and therefore we only compute normalization at the end of the algorithm to obtain eigenvectors of the same length.

We will focus on weighted normalization, that is, for each vector u_i we want a constant weighted variance

$$\frac{\sum_{v_j \in V} k_j (u_i(v_j) - M_i)^2}{\sum_{v_j \in V} k_j} = 1,$$

where M_i is the weighted mean

$$\frac{\sum_{v_j \in V} k_j u(v_j)}{\sum_{v_j \in V} k_j}.$$

D-normalization against vector u_1 warrants that $M_i = 0$, so we just need to compute the weighted average t for the $u_i(v_j)^2$, and divide each $u_i(v_j)$ by $\sqrt{t}$.

We can notice that iterating the step in line 14 of algorithm 4, starting with a given $u(i)$ value for each node v_i will converge to the weighed average of the u vector on each node. In fact, the total sum

$$S = \sum_{v_i \in V} k_i u\left(v_i\right)$$

does not change: consider a single pair of connected nodes v_a and v_b. After each interaction, v_a will be displaced towards v_b by a distance of

$$\frac{u\left(v_a\right) - u\left(v_b\right)}{k_a},$$

and the displacement caused to v_b will be

$$\frac{u\left(v_b\right) - u\left(v_a\right)}{k_b}.$$

The link between v_a and v_b does not change S, and the same can be said for any other linked pair.

As the result of the discussion in this section, algorithm 5 performs a distributed normalization of all u_j vectors.

5.3.3 The Decentralized Algorithm

All modifications discussed so far yield to the distributed version shown in algorithm 6. The relationships between the number of iterations N, the quality of the resulting layout and the characteristics of the network are unfortunately not completely clear yet. For this reason, we recommend to use simulation to determine a suitable value of N for a given network.

The communication overhead of the algorithm is due to the need of passing, at each iteration, the coordinates of a given node to each of its neighbors. A node with degree k will thus send and receive Nk messages containing d floating point numbers each, where N is the number of iterations and d the number of dimensions.

A variant for directed networks could take into account only outgoing edges from node v_x in step 13, so that untrusted nodes can not influence the position of the node.

5.4 Experimental Validation

Once the layout is obtained, the routing algorithm is extremely simple, analogous to the "greedy" routing used in DHTs such as Chord [SMK+01]: the route progresses from node

Algorithm 5 Distributed normalization algorithm.

Parameters: on each node v_x, the values $u_2(v_x) \dots u_{d+1}(v_x)$ and the list of neighbors so that $(v_x, v_y) \in E$.

On each node v_x, execute the following code:

$N \leftarrow 200$ /* *number of iterations* */

for $j \in 2 \dots d+1$ **do**
  /* *Initialization* */
  $m_{j,x} \leftarrow u_j(v_x)^2$
end for

for $i \in 1 \dots N$ **do**
  /* *Approximate* $\frac{\sum_{v_j \in V} k_j \cdot u_i(v_j)^2}{\sum_{v_j \in V} k_j}$ *in* $m_{j,x}$ */
  for $j \in 2 \dots d+1$ **do**
    $m_{j,x} \leftarrow \frac{1}{2}\left(m_{j,x} + \frac{\sum_{(v_x,v_y) \in E} u_j(m_{j,y})}{k_x}\right)$
  end for
end for

for $j \in 2 \dots d+1$ **do**
  /* *Divide* $u_j(v_x)$ *by* $m_{j,x}$ */
  $u_j(v_x) \leftarrow \frac{u_j(v_x)}{\sqrt{m_{j,x}}}$
end for

Algorithm 6 Distributed layout algorithm.

Parameters: on each node v_x, the list of neighbors so that $(v_x, v_y) \in E$.
Return value: on each node v_x, its layout $u_2(v_x) \dots u_{d+1}(v_x)$.

On each node v_x, execute the following code:

$N \leftarrow 200$ /* *number of iterations* */

for $i \in 2 \dots d+1$ **do**
  /* *D-orthogonalize* u_i *against* $u_1 = 1^n$ */
  if a random paired node v_y is found **then**
    negotiate a random number r, giving $-r$ to v_y
    $u_i(v_x) \leftarrow \frac{r}{k_x}$
  else
    $u_i(n) \leftarrow 0$
  end if
end for

for $i \in 1 \dots N$ **do**
  /* *Multiplication* */
  for $j \in 2 \dots d+1$ **do**
    $u_j(v_x) \leftarrow \frac{1}{2}\left(u_j(v_x) + \frac{\sum_{(v_x,v_y)\in E} u_j(v_y)}{k_x}\right)$
  end for
end for

Perform normalization as shown in algorithm 5

return $u_2(v_x) \dots u_{d+1}(v_x)$

to node by selecting the neighbor that does not appear in the route and is closest to the destination according to the layout. When there is no possible route starting from the current node, it gets removed from the route and the next-to-last node in the route continues by choosing another neighbor. The route length is defined to take into account all visited nodes, thus counting also nodes where the route research failed.

An extremely simple optimization has been adopted: nodes with degree 1 that are not the destination are obviously useless as forwarders, and are thus excluded from routes.

The dataset used is the OpenPGP web of trust introduced in section 4.5.1. To obtain a connected network with undirected edges reflecting mutual trust, we put edges between nodes that reciprocally signed their public keys and extract the largest connected component. The resulting network contains 26,350 nodes with an average degree of 7.98; the average clustering is 0.3047.

In our experiments, a series of 10,000 pairs of nodes was generated. In order to reflect the fact that nodes that are more active in the web of trust are probably more active on the network, a node gets picked with a probability that is proportional to its degree.

There are no theoretical bounds on the convergence time of Sandberg's algorithm, and, as done in his work, we executed $6,000 \cdot n$ iterations of the algorithm.

Based on the success of strategies that perform search in unstructured networks by directing the search towards neighbors with higher degrees [ALPH01], routing strategies that took in consideration degree as well as distance on the layout were taken into consideration. We chose to discount the distance of each node i from the destination with a factor k_i^α, where k_i is the degree and α is a tunable parameter. Since the simpler layout-based routing constantly outperformed these strategies in the cases of 10 and more dimensions, they have been left out of this analysis.

5.4.1 Comparison Between Layout Algorithms

Figure 5.2 shows the results of running the above simple routing algorithm using the layout-defining algorithms developed by Sandberg and Koren and the same sequence of pairs. The graph shows the probability that the path length is lower than x steps. To give a clear view of the performance of the various algorithms both for long and short paths, logarithmic scale is used on the horizontal axis. The position of keys in the legend reflects the position of lines in the diagrams, from top (best) to bottom (worst).

Percentiles are given for path lengths: for example, 50% of all the searches are concluded in 57 steps or less when using the layout given by Sandberg's algorithm.

In order to give a reference against a best and and a worst case, the length of the shortest

paths is shown; a "random" routing algorithm has been also implemented. It proceeds by following edges at random, until the destination node is one of the neighbors of the current node. The execution of the random algorithm has been cut when the route length reached the size of the network.

Since a relevant percentage of routes "get lost" and need to visit a very high number of nodes before reaching the destination, the distributions are highly skewed and thus the average values are much higher than the medians (the average for Sandberg's layout is 1,452.55 steps, for Koren's layout is 552.28 and 114.56 for 1 and 10 dimensions respectively). We believe these values are not significant, since any realistic approach would choose a reasonable maximum number of hops; the performance due to the layout can be evaluated by counting the percentage of routes that terminate before that number of hops is reached.

While Sandberg's algorithm does not gain significant advantages by increasing dimensionality, it appears clear that Koren's layout obtains a great benefit from layouts with a high number of dimensions. Our evidence proves that the spectral layout algorithm provides a significant advantage over the algorithm proposed by Sandberg, especially when dimensionality is increased.

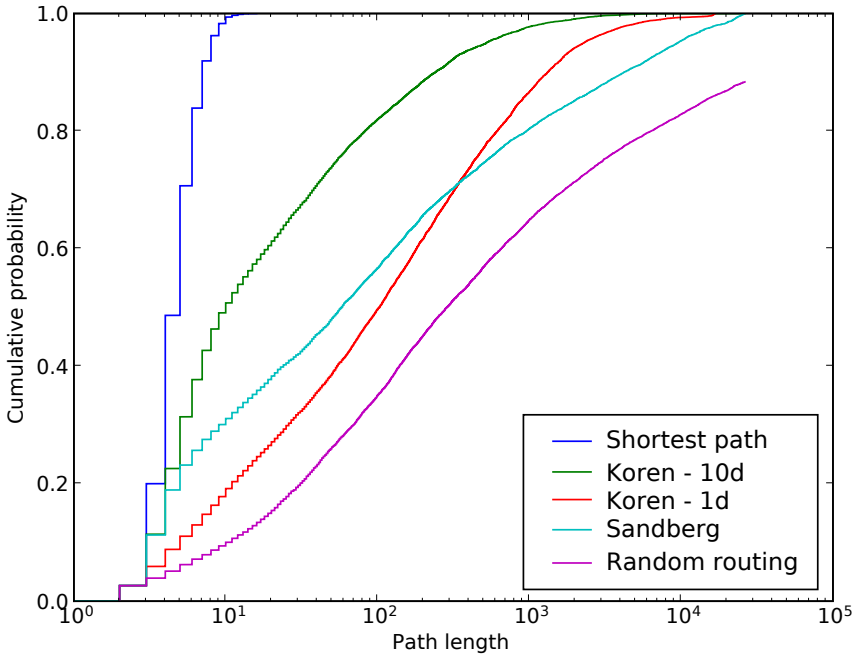
5.4.2 Decentralized Algorithm

Figure 5.3 compares the experimental results of using Koren's algorithm and its decentralized approximation of algorithm 6. Since the normalization step has quadratic complexity with respect to the number of dimensions (each vector needs to be orthogonalized against all previous eigenvectors), it limits the number of dimensions for which Koren's algorithm can be applied. On the other hand, removing this step allows us to use a much higher dimensionality. As experimental results show, the distributed implementation has results that are comparable to Koren's algorithm. Moreover, increasing dimensionality yields notably better results.

5.4.3 Neighbor-of-Neighbor Routing

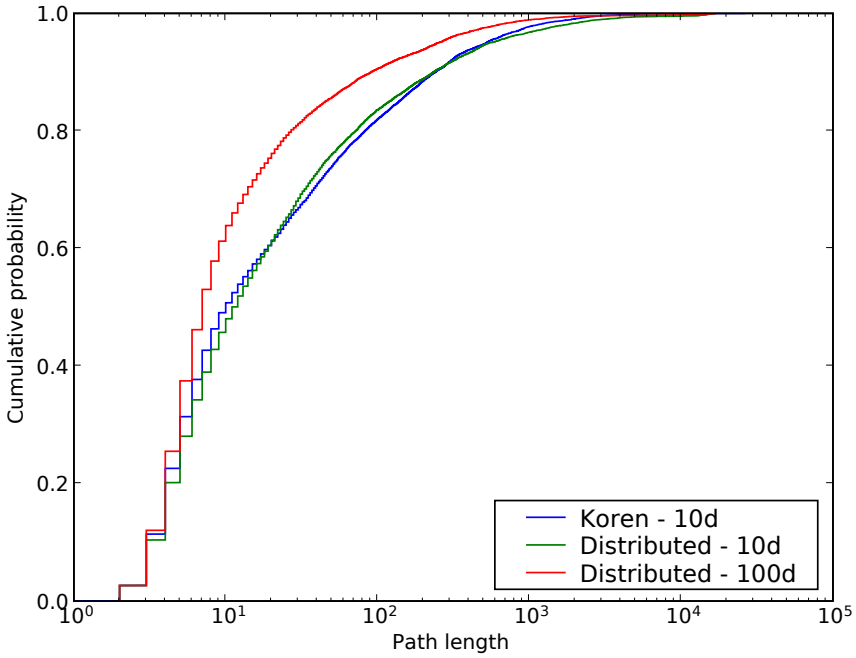
An obvious optimization for our routing algorithm, without changing the layout we produce, could be based on information about neighbors of neighbors (NoN): this method assumes that each nodes shares information about the positions on the layout of its neighbors to all of them.

In our case, this information can be exchanged at the end of algorithm 6. The routing algorithm is modified so that the route progresses towards the node that is closest to the destination among the neighbors of neighbors. The distributed implementation is natural, since the information about neighbors can be exchanged by just using the connections



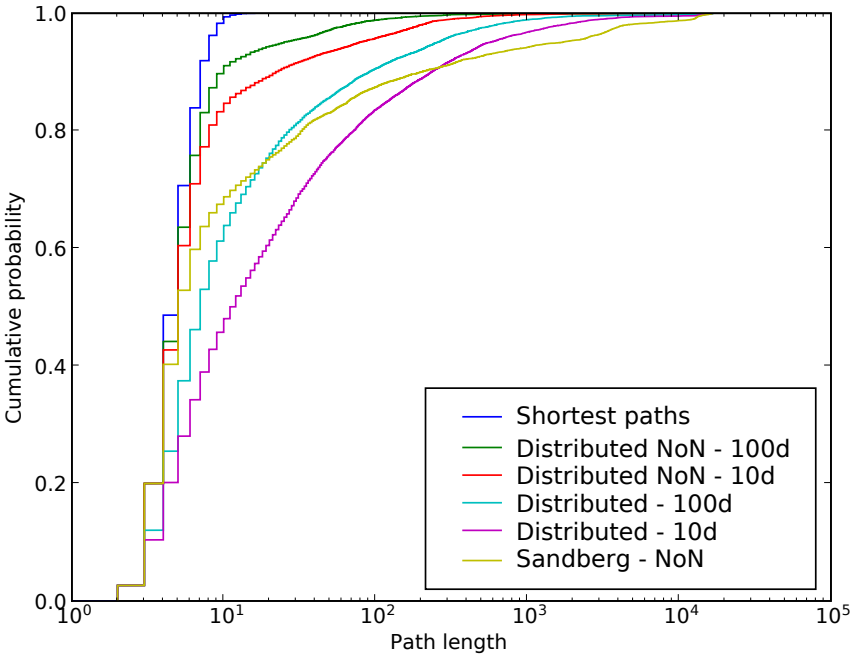
Percentile	25	50	75	90	99
Shortest path	3	4	5	6	9
Koren - 10d	4	9	53	241	1,912
Koren - 1d	16	102	433	1,320	7,475
Sandberg	5	57	523	4,202	22,297
Random	45	286	3,063	-	-

Figure 5.2: Comparison of path lengths using the layout algorithms developed by Koren and Sandberg.



Percentile	25	50	75	90	99
Shortest path	3	4	5	6	9
Decentralized - 100d	3	6	18	90	1,135
Decentralized - 10d	4	10	45	233	3,177
Koren - 10d	4	9	53	241	1,912

Figure 5.3: Experimental results for the decentralized algorithm shown in algorithm 6.



Percentile	25	50	75	90	99
Shortest path	3	4	5	6	9
Decentralized NoN - 100d	3	4	5	9	114
Decentralized NoN - 10d	3	4	6	21	320
Decentralized - 100d	3	6	18	90	1,135
Decentralized - 10d	4	10	45	233	3,177
Sandberg - NoN	3	4	18	199	12,492

Figure 5.4: Experimental results for NoN routing.

that are present on the web of trust. If anonymity is an issue, it could be possible to "obfuscate" the information on neighbors by sharing a slightly inaccurate information about the position of neighbors, as done in Neblo [Cia06].

Experimental results in figure 5.4 show that the NoN approach is very effective, at the cost of an increase in the amount of local information each node has to manage. The numeric results for path lengths suggest that network maps are already good enough to perform efficient routing in social networks: in the OpenPGP network, in 90% of cases, the algorithm is able to discover paths of length 9 or less.

5.5 Ideas for Future Work

In this thesis, we only adopt network maps as methods to find paths in social networks; however, they give a decentralized method to analyze network and discover similar items; applications using this approach such as the ones mentioned at the beginning of this chapter can be studied.

From the point of view of the security of the approach in "friend-to-friend" networks, the resilience to attacks on the layout algorithm needs to be investigated. A possible means could be based on limiting the influence of any single node in the placement of its neighbors.

To obtain better routing with a controlled increase in the number sent messages, the technique of probabilistic broadcasting [SBR04] could be introduced, introducing bifurcations in search paths when the layout routing appears to fail.

Chapter 6

Social Filtering

This work, so far, has regarded ways of computing reputation efficiently in a web of trust, with the assumption that reputation can be used in conjunction with reciprocative behavior to provide incentives towards cooperation. In this chapter[1], we focus on another application of webs of trust and reputation, namely their inclusion in information filtering to make obtain resistance to attack.

As discussed in section 1.2, we weave reputation and collaborative filtering (CF) to create the novel approach of *social filtering*. In CF, users' trustworthiness has been measured according to one of the following two criteria: *taste similarity* (i.e., "I trust those who agree with me"), or *social ties* (i.e., "I trust my friends, and the people that my friends trust"). The former criteria aims at identifying *competent* users, but is subject to abuse by malicious behaviors. The latter aims at detecting *well-intentioned* users, but fails to capture the natural subjectivity of tastes. We argue that, in order to be trusted, users must be *both* well-intentioned and competent.

In this chapter we describe SOFIA, an algorithm that realizes the social filtering philosophy, and validate its performance on two real large-scale datasets. Experimental results show that the recommendations produced by SOFIA are both accurate and attack resilient.

6.1 Philosophy of the Approach

In chapter 1, we sketched the reasons why the "long tail" phenomenon creates a need for personalized recommendations: market forces drive traditional retailers to sell only those items that have the best chance to sell, thus losing less popular ones. Internet-based

[1]This chapter is based on joint work with Licia Capra [DC07].

applications, on the contrary, can afford to offer a much wider choice of items. The "market share" of the long-tail items is impressive: even though the quantities for each of them may be small, the numbers add up to large volumes overall, given the length of the tail itself.

Long-tail contents have a popularity which is much lower than the top items, but this does not necessarily imply a lower quality: their appeal may be high, but restricted to a small section of the overall population. This raises the issue of *connecting supply and demand*: helping users finding the particular niche content they would enjoy. This is precisely the goal that filtering techniques try to satisfy.

The most popular technique to realise this connection is collaborative filtering (CF) [HKBR99]. Most of the work on collaborative filtering has been focusing on identifying users with similar preferences, and then recommending items that people with similar tastes have approved. The adoption of collaborative filtering has arguably been a key factor in the success of Amazon.com: readers are invited to send reviews of books and to rate them on a five-star scale; other readers can comment on how useful the reviews were. By so doing, customers with similar tastes are detected and their past purchases used to create recommendations for similar users for what to buy next (i.e., thus connecting supply and demand). Traditional collaborative filtering techniques have worked quite well for the mass market and under the assumption of collaborative behaviors. However, these techniques have been subject to abuse by malicious behaviors [LFR06]: for example, malicious users could copy honest users' reviews, to gain high similarity scores with them; they could subsequently inject inflated reviews in the system, to trick those users into buying an item or, viceversa, to disrupt an item's sales. Episodes of this kind have been documented on Amazon.com [Ols02, Har04].

We argue that *accurate* and *robust* filtering techniques can be devised by exploiting information from webs of trust. We call this approach *social filtering*. The core idea is to give higher weight to recommendations received from *trusted* users. To be trusted, a user must be both *well intentioned* and *competent*. Traditional collaborative filtering techniques focus only on competence (i.e., the ability to give useful - in a subjective way - recommendations), without considering the fact that competent users may indeed be malicious. Rather than relying on all recommendations from similar (i.e., competent) users, our approach specifically looks for well-intentioned users (i.e., users who are willing to provide honest recommendations) in the web of trust. Social ties are a warranty against malicious behaviors: if the trust inference algorithm is robust, it would be very costly for an attacker to build enough friendships with 'honest' users to effectively subvert the system. By so doing, we can prove that social filtering is more robust than traditional CF. Moreover, even in the absence of malign behavior, we demonstrate that social filtering yields to more accurate recommendations, in situations where social ties intrinsically reveal information about users' similarity (e.g., two friends are more likely to share hobbies and tastes than

two complete strangers); this is especially important when dealing with items that strongly appeal only to a small/niche community.

Note that the network of social interactions (i.e., the "word of mouth" phenomenon) has played an important role in the successful commercialization of many items; on the Internet, it can be diffused via personal recommendations, instant messaging, or blogs. Empirical evidence shows that users navigate their social networks to find content on Flickr [LJ07]. The probability that a user recommends some content to another user depends on both the closeness of their social relationship and their similarity in tastes; it is precisely this phenomenon that our social filtering approach tries to capture and automate.

The algorithms described in this chapter assume complete knowledge of the web of trust and all judgments the users expressed. Current work on a decentralized approximation for SOFIA will be introduced in chapter 7.

Social filtering relies on the identification of *trusted* recommenders. In the scope of this work, we call trusted a recommender that is both well-intentioned and competent. The three questions we are thus trying to answer are: (1) how to evaluate intention (section 6.1.1); (2) how to evaluate competence (section 6.1.2); and (3) how to combine this information to find trusted recommenders (section 6.1.3).

6.1.1 Intent – Trust over Users

We define intent as the the *willingness* of a user to provide honest judgments[2], differentiating "spammers" from people who are legitimately using the application. Note that a judgment given with good intent is not necessarily useful, since users may have different tastes and preferences; section 6.1.2 will illustrate how to find competent users among well-intentioned ones.

Webs of trust are perfectly adequate to represent intent of users; (subjective) trust on intent can be thus calculated by using reputation metrics based on transitive trust propagation, as described in section 2.3.1. One caveat on the creation of the web of trust has to be observed: the security of the approach is based on the assumption that it is difficult, for malicious nodes, to obtain endorsements from honest ones. For this reason, we strongly discourage the creation of the web of trust via automated matching purely based on users' similarity.

[2]In the following, we will use the more general term 'judgments', instead of 'recommendations', as our approach is equally applicable to recommendations (i.e., endorsements of products or content) as to 'negative' or purely informative judgments (e.g., "avoid that restaurant" or "this is relaxing music").

6.1.2 Competence – Trust over Judgements

Together with intent, competence is a key component in evaluating the trustworthiness of recommenders. In this work, we define *competent* those users who are able to make correct judgments; since the definition of "correct" judgments is inherently subjective, competence is a subjective matter as well.

Again referring to section 2.3.1 for the discussion, the co-citation pattern that propagates trust to users that expressed concordant judgments appears adequate to capture this subjective issue.

However, users' competence is not sufficient to warrant trust to their judgments. For instance, let us consider a malicious user Mallory, wishing to trick Alice in believing a dishonest judgment Z stating that "Mallory's Greasy Restaurant offers very good food". In order to do so, Mallory could simply copy Alice's judgments; using the co-citation trust propagation pattern, Alice would deem Mallory a very competent evaluator, and would consequently believe/trust judgment Z too.

We argue that competence should thus be combined with intent to identify trustworthy recommenders, that is, recommenders who are willing to provide us with honest judgments and that we are likely to find useful.

6.1.3 The Combined Approach

As discussed above, using the *transitivity* trust propagation pattern alone is not enough, as subjectivity of tastes, which is an intrinsic characteristic of judgments, is lost. On the other hand, using the *co-citation* trust propagation pattern alone is subject to abuse by malicious users.

We propose a novel approach that combines the strengths of the two patterns, while circumventing their individual weaknesses: we exploit the transitivity trust propagation pattern on the web of trust to determine well intentioned users, and the co-citation trust propagation pattern on the network of judgments to evaluate their competence. By so doing, we are capable of inferring trust over judgments, in a way that is both accurate and robust. The underpinning idea is that, in order to be trusted, a judgment must have been expressed by a user who is both *willing* (intent) and *able* (competence) to give useful judgments. We call the new approach *social filtering*. Based on the interpretation of trust propagation over intent and competence we gave in the previous two sections, A can infer trust for a judgment Y expressed by a user D (figure 6.1) if:

1. there exists a directed path from A to D in the web of trust (e.g., $A \rightarrow B \rightarrow C \rightarrow D$);
2. A and D both expressed at least one common judgment (e.g., X).

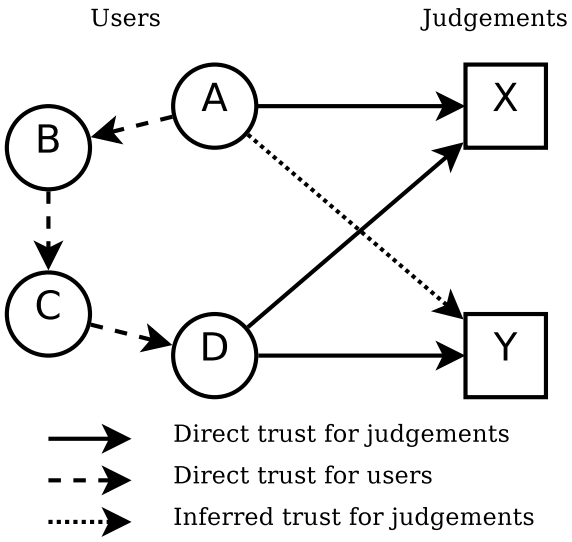


Figure 6.1: Combined trust-propagation approach.

This is the first approach we are aware of that aims at increasing the utility of recommendations, by exploiting information coming from the social network *and* from individual's preferences at the same time. We are aware of only two other works where the transitivity and co-citation trust propagation patterns have been used together, but with rather different goals and following a different philosophy: in [GKRT04], trust is propagated using *either* co-citation or transitivity in a social network where links represent similarity in preferences; in [MA07a], the transitive trust propagation pattern is used as an *alternative to* the co-citation pattern, in order to bootstrap trust when traditional user similarity cannot be computed, again because of lack of information. These approaches work well in those scenarios where there is a strong correlation between social ties and individual preferences. On the contrary, our approach is best suited in those scenarios where the social network is not just a surrogate of users' preferences. As we shall demonstrate in Section 6.4, when separate information is available about the web of trust and judgments, an approach that reasons about intent and competence *at the same time* can yield the biggest increase in the utility of recommendations, even in the absence of malign behavior. Before doing so, we discuss how we have realised social filtering in practice.

6.2 Realization of the Approach

In the previous section, we have introduced social filtering from a philosophical viewpoint, highlighting the advantages of propagating trust over both intent and competence, in order to give users *trusted judgments*. To be of practical use, an implementation of social filtering

would need to attribute a numeric value to the *amount* of trust a judgment deserves. This would ultimately allow users to rank judgments and/or to filter out unreliable ones. In this section, we first describe how the transitive and co-citation trust propagation patterns have been separately implemented in literature; we then motivate the choices we have adopted, and how they have been uniquely combined in SOFIA, our own implementation of social filtering. In describing our implementation, we will refer to the general case of weighted social networks, with weights expressing the strength of social ties. The user-judgment edges can be weighted as well, representing the level of confidence of a user towards a given judgment. The unweighted case is just a specific instance of the more general one, with all instances of trust relationships and/or judgments having the same weight.

6.2.1 Evaluating Intent

SOFIA works in two phases: in the first phase, the web of trust is examined and intent values are computed for all nodes; in the second phase, those values are used as an input to weigh the authoritativeness of each node.

To estimate intent for all the nodes, a reputation metric is needed. In case of a Sybil attack, we are interested in limiting the *overall* trust gained by the set of Sybil nodes, since SOFIA considers the opinions of many of them simultaneously. Personalized PageRank (PPR), as discussed in section 2.3.4, has this property and thus appears as a reasonable choice for our case.

We recall that PPR simulates a random walk starting at the evaluating node and stopping at each step with probability α. This parameter influences the resilience to attack since a Sybil attack cannot increase the reputation of the Sybil region by a factor higher than $\frac{1}{1-\alpha}$: the lower the value of α the better the robustness. Low values of α also increase subjectivity, as they reward short paths over long ones, while when α approaches 1 the outcome of the algorithm becomes more and more similar, regardless of the initiator node. Finally, the lower the value of α the faster the convergence speed of the algorithm (with $\alpha = 0.5$, more than 99.9% of the overall ranking weight comes from paths of length up to 10). Note, however, that low values of α may cause honest nodes who are "socially far-away" not to be considered, thus discarding potentially useful information. This may affect the accuracy of our algorithm, with respect to traditional collaborative filtering techniques where the full dataset is considered instead. We will analyse optimal choices of α with respect to accuracy vs. robustness in section 6.4.

In our realisation of social filtering, we have chosen to deploy Personalised PageRank to quantify the transitive trust propagation over the social network, as it combines our requirements of subjectivity and robustness. A pseudo-code description of Personalised PageRank can be found in algorithm 7. The computational complexity of the algorithm

Algorithm 7 Personalised PageRank.

Parameters: a social network $G = (V, E)$; an evaluating node $A \in V$; weights such that w_{ij} is the weight of edge (i, j); a $0 < \alpha < 1$ parameter.
Return value: a vector r where r_i is the score of node i.

$n \leftarrow$ size of V; $r \leftarrow 0^n$; $r_A \leftarrow 1$

while algorithm has not converged **do**
  $\hat{r} \leftarrow 0^n$; $\hat{r}_A \leftarrow 1 - \alpha$
  for all $i \in V$ **do**
    $d \leftarrow \sum_{j \in V} w_{ij}$
    if $d > 0$ **then**
      for all j such that $(i, j) \in E$ **do**
        $\hat{r}_j \leftarrow \hat{r}_j + \alpha \frac{w_{ij} r_i}{d}$
      end for
    else
      $\hat{r}_A \leftarrow \hat{r}_A + \alpha r_i$ /* *If i is a sink we restart from A* */
    end if
  end for
  $r \leftarrow \hat{r}$
end while

return r

is proportional to the number of edges in the web of trust times the number of iterations needed to make the algorithm converge.

6.2.2 Evaluating Competence

The co-citation trust propagation pattern has been widely studied and applied to the problem of ranking Web pages. One of the most famous algorithms realising this pattern is HITS [Kle99]. HITS conceptually divides pages in two subsets: authorities (i.e., pages whose content satisfy the query), and hubs (i.e., pages that link to relevant documents, that is, to authorities). Using an iterative process, HITS traverses the linkage structure of Web documents, and computes both a hub weight and an authority weight for each visited page at every step, so that:

1. Forward Step (from hubs to authorities): the weight given to an authority is proportional to the sum of the weights of those hubs linking to it;

2. Backward Step (from authorities to hubs): the weight given to a hub is proportional to the sum of the weights of those authorities being linked by it.

If weights expressing confidence are present in the network of judgments, they can be used as a multiplicative factor (i.e., a link with weight 2 acts as two separate links, each with weight 1). The process continues (renormalizing scores at every iteration) until it converges, and the top ranking pages, according to their authority scores, are then returned.

The principle behind HITS is that good hubs link good authorities, and good authorities are linked by good hubs, in a mutually reinforcing way. We argue that the same principle holds in our scenario, where we can expect competent users to give valuable judgments, and valuable judgments to be given by competent users. If we map users to hubs and judgments to authorities, we can run an HITS-like iterative algorithm to rank judgments, which is our ultimate goal. This would not realise our social filtering method though, as the following caveats must be addressed first.

Solving the TKC problem (and rewarding for niche judgments). It has been demonstrated that the HITS algorithm suffers from the "Tightly Knit Community" (TKC) syndrome [LM01]: if a community of users all gave the same (or very similar) judgments (thus resulting in a highly connected bipartite graph), the competence weight of the community would disproportionately increase, with the judgments they express being excessively high-ranked, even if they are not authoritative. A set of malicious users could thus artificially create a TKC in order to artificially boost their ranking.

To solve this problem, we adopt the solution proposed in SALSA [LM01]: we divide the weight that each hub transfers at each forward step by its outdegree (the sum of weights on outgoing edges), and we do the same for authorities and their indegree at each backward step. After a forward step, the total weight transferred from a single hub to its linked authorities is thus equal to the weight on that hub; viceversa, after a backward step, the total weight that is redistributed from a single authority to the set of hubs linking to it equals the weight gained by the authority. Thus, the sum of weights remains constant at every step, removing the need for normalization.

A very desirable side-effect of this alteration is that users who express non-mainstream judgments are rewarded: the fewer the users who have expressed some judgments, the higher the weight that is transferred back on a per-user basis. In so doing, we express the fact that "niche" judgments are more significant than mainstream (redundant) ones: if a user, in her top 10 listenings, has both The Beatles and an unheard-of rock band, it is likely that the unknown rock band is more indicative of her musical preferences. This is analogous, in principle, to the TF-IDF approach used in Information Retrieval [BYRN99], where the significance of a term in a document is determined by multiplying the number of occurrences in the document (Term Frequency) with

a logarithm of the *inverse* of the overall frequency of that term (Inverse Document Frequency).

Subjectivity of ranking. HITS-like algorithms provide non-subjective results, as they are independent of the user A starting the search. To cater for the subjectivity required by our scenario, we initialize the algorithm so that the only hub (user) with a non-zero weight is the reference node A itself (instead of assigning an equal weight to any hub in the network). In so doing, the first forward step of the algorithm only considers the judgments given by the reference node, thus tailoring the ranking results to his/her tastes. To limit the propagation of trust to judgments that are too dissimilar from the tastes of A, after each backward step, the weights associated to each user are multiplied by a parameter $\beta \in (0, 1)$, and the trust given to A is increased by $1 - \beta$. These two changes are similar, in spirit, to the modifications already suggested for PageRank, where we forced the random walk to start from the very same node; the β parameter plays the same role that α plays in PageRank, ensuring the convergence of the algorithm, with lower values of β impling faster convergence and higher subjectivity.

Catering for Well-Intentioned Users. As discussed in section 6.1.2, trust propagation over competence alone is susceptible to attacks. We propose to add robustness to HITS-like algorithms, by incorporating users' intent assessment as follows. To begin with, Personalised PageRank (Algorithm 7) is run on the social network, thus obtaining a vector with nodes' reputation, as seen by the reference node A. We then run the subjective HITS-like algorithm, so that, at every backward step, trust is redistributed from judgments to users in a way that is *proportional to users' intent*, as measured by Personalised PageRank. In other words, *reputation becomes a multiplicative factor for backward trust propagation.* As discussed in section 6.2.1, a Sybil coalition can obtain only a limited amount of trust from the social network, so the amount of trust that can be transferred to malicious nodes is limited too.

We call the algorithm that results from modifying the HITS-like approach in the three ways described above SOFIA, that is, SOcial FIltering Algorithm. The resulting pseudocode is shown in algorithm 8.

The result of running SOFIA is a pair of vectors $\langle t, \hat{t} \rangle$ containing *trust* numeric values for respectively users in U and judgments in J, computed considering both the intent and the competence, as seen by the reference node A. The normalization parameters ($\sum_{k \in J} w_{uk}$, $\sum_{v \in U} w_{vj} r_v$) can be calculated outside the loops, so the computational cost of the algorithm is, similarly to PageRank, proportional to the number of edges in E times the number of iterations of the algorithm.

Algorithm 8 SOFIA.

Parameters: a judgment bipartite network $G = (V, E)$, where V is the union of the set of users U and the set of judgments J; an evaluating node $A \in U$; weights such that w_{uj} is the weight of edge (u, j); an intent ranking vector r computed using Personalised PageRank over the web of trust, so that r_u is the intent ranking of user u; a $0 < \beta < 1$ parameter.
Return value: a pair of vectors $\langle t, \hat{t} \rangle$ such that t_u is the trust ranking of user u, and $\hat{t}_j$ is the trust ranking of judgment j.

$n \leftarrow$ size of U
$m \leftarrow$ size of J
$t \leftarrow 0^n$
$t_A \leftarrow 1$

while algorithm has not converged **do**

/ Forward Step: from users to judgments */*
$\hat{t} \leftarrow 0^m$
for all $(u, j) \in E$ **do**
$\hat{t}_j \leftarrow \hat{t}_j + \frac{w_{uj}}{\sum_{k \in J} w_{uk}} t_u$
end for

/ Backward Step: from judgments to users */*
$t \leftarrow 0^n; t_A \leftarrow 1 - \beta$
for all $(u, j) \in E$ **do**
$t_u \leftarrow t_u + \beta \frac{w_{uj} r_u}{\sum_{v \in U} w_{vj} r_v} \hat{t}_j$
end for

end while

return $\langle t, \hat{t} \rangle$

6.2.3 Example

To see how SOFIA works, let us consider the sample web of trust and judgment network depicted in figure 6.2. The web of trust represents "friendship" relationships on a social networking site between users Alice (A), Bob (B), Carol (C) and Dave (D) (e.g., Alice listed Bob and Carol as friends; Bob and Dave mutually listed themselves as friends, etc.).

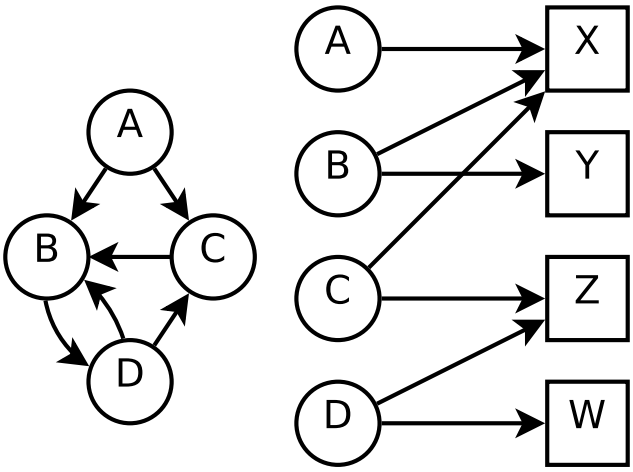


Figure 6.2: Example web of trust (left) and network of judgments (right).

Judgments represent recommendations of their favorite music bands: the Xenons (X), Yodels (Y), Zaars (Z) and Whistlers (W) (e.g., Alice, Bob and Carol all endorse the Xenons; Bob also likes Yodels, etc.).

We now want to give A, our reference user, recommendations about music bands she may like. We thus run SOFIA over the graphs displayed in figure 6.2, where we assume all edges to have unitary weight. We also set the parameters $\alpha = \beta = 0.5$.

To begin with, Personalised PageRank is executed over the web of trust. The algorithm yields a normalised intent ranking of 0.50 for A, 0.231 for B, 0.154 for C and 0.115 for D. Note that B obtains a higher ranking than C, as he is trusted by all other users in the web of trust (despite both being at the same distance from A); D is penalized instead for being farthest from A. The penalty for long paths can be adjusted by varying the α parameter.

After having computed user intent, we then run the core of the SOFIA algorithm, to rank judgments. The algorithm converges to vector $\hat{t} = \{0.909, 0.051, 0.035, 0.005\}$, listing the authority weights for X, Y, Z and W respectively. Note that Y obtains a higher ranking than Z: in fact, although they follow the same propagation pattern (one-step propagation from A via X), the intent ranking used during the backward step from X is higher for B than for C. For the opposite reason, W obtains the lowest ranking, as its recommendation is obtained via two steps of transitivity (via Z from C, and up via X from A), and it is endorsed by D, who has the lowest intent ranking.

6.3 Attack Model

In order to validate our social filtering algorithm, we have conducted a variety of experiments on two very large real datasets. While ideal to measure accuracy, real datasets are unsuitable to test the robustness of the algorithm while varying threat intensity. To demonstrate the robustness of SOFIA, we thus have to manually inject attacks on top of real datasets, and run experiments under different configuration settings. In this section, we analyse threat strategies, leaving their enactment and corresponding experimental validation to section 6.4.

In the scenario we are considering, the most plausible goal of an attacker would be to alter the rating of a certain judgment X. It may do so either to trick a single user A, or more extensively to deviate the judgments of all users, in favour of (or against) X. Let us analyse how an attacker could achieve such goal. In the first case, since the attacker wants to be rated by A as a very competent user, it could first copy the judgments that A expressed, and then add a new judgment X. In the second case, there is no single set of judgments the attacker can copy, as each user would have expressed different ones: copying popular judgments would yield to very little reward, as a consequence of our strategy to reward users who gave niche judgments more (see section 6.2.2 point (1)); on the contrary, copying 'niche' judgments would yield to very high appeal, but to rather few users. We will thus model this attack as we modeled the targeted attack, that is, by copying the judgments of a randomly chosen node A and adding the judgment for X; however, rather than studying the impact of the attack on A, we will study the 'collateral damage' that the attack has on other users.

The attack strategies described above model the behavior of one attacker only. However, to increase the impact of the attack itself (i.e., to increase the ranking of judgment X), we must also consider the case of an attacker who has the ability to create an unlimited number of Sybil identities, all endorsing X. We assume that each Sybil can create any number of outgoing edges in the web of trust, from the Sybil node to any other user. They can also create any number of incoming edges, originating within the Sybil coalition. However, what they cannot do is create incoming edges from honest nodes at will, since obtaining trust from well-intentioned peers is costly. It is thus reasonable to expect a low cut between the "honest" and the "Sybil" region [YKGF06]. In our experiments, we will thus create Sybil regions that are highly interconnected internally; we will then set the amount of incoming links from honest nodes as a parameter, and analyse the robustness of SOFIA (i.e., how highly ranked can X become) against it.

6.4 Experimental Validation

We have evaluated SOFIA along two dimensions: accuracy and robustness against Sybil attacks. The results are reported in section 6.4.2 and 6.4.3 respectively. Both experiments were conducted using data from two real datasets: the Citeseer online scientific digital library, and the Last.fm music and social networking website. The key characteristics of these datasets are briefly summarised below.

6.4.1 The Datasets

Citeseer (`http://citeseer.ist.psu.edu/oai.html`) is an online scientific literature digital library, containing over 750,000 documents. From this repository, we have extracted a social network based on the co-authorship relation: if A and B have co-authored n papers together, then an edge between the two will be added to the social network, with weight n. The judgment network is built from the citations instead: if a paper X authored by A cites paper Y, then an (unweighted) edge from A to Y is added to the judgment network; the rationale is that, by citing Y, the authors of X have expressed the judgment "Y is relevant with respect to the topic discussed in X". To obtain a more manageable subset of the whole network, we isolated a highly-clustered subset of 10,000 authors, and took in consideration only the papers that had them as authors. The result is a set of 182,675 different papers; 48,998 of them received at least one citation by one of the others. The average degree of the web of trust is 64.87.

Last.fm (`http://last.fm/`) is a "social music" website that creates profiles of the musical taste of its users, by tracking which songs they listen more often to in their digital music players. As in other social networking websites, users can explicitly create an (unweighted) social network by adding other users to their friend-list. We gathered our social network with a breadth-first crawl of 10,000 users using the Audioscrobbler Web Services (`http://www.audioscrobbler.net/data/webservices/`). We then considered the 50 most listened artists of each user, and ended up with a total of 51,654 different artists. The judgment network was finally created by linking users to their most listened artists (thus representing the judgment "user A likes to listen to songs by X"), and by weighting each judgment edge with the number of times the user listened to songs by that artist. The average degree of the web of trust is 7.25.

Algorithm	Iterations	Ranking percentiles						
		5	10	25	50	75	90	95
PPR ($\alpha = 0.3$)	3	**1**	**2**	8	32	161	4,293	–
	5	**1**	**2**	8	30	**115**	**1,709**	**11,609**
	10	**1**	**2**	**7**	**29**	141	3,341	20,287
N-SOFIA ($\beta = 0.05$)	3	**1**	**1**	**3**	12	67	1,060	–
	5	**1**	**1**	**3**	12	**63**	1,136	–
	10	**1**	**1**	**3**	**11**	72	**1,020**	–

Table 6.1: Hidden judgment ranking of PPR and N-SOFIA (best results in bold) with different numbers of iterations on the Citeseer dataset.

6.4.2 Accuracy

In Collaborative Filtering, datasets are usually composed of judgments on a positive/negative scale such as, e.g., a zero to five stars ranking. CF algorithms assemble those rankings and try to predict how a user would rank a given object such as a book or a film. In that case, evaluating accuracy is straightforward: judgments are removed from the dataset and then the CF algorithm is used to predict the missing judgments. In our datasets, though, only one kind of judgment can be present for any given object: judgments such as "I don't like to listen to artist X" or "paper X is irrelevant with regard to this topic" are simply not represented; trying to predict the judgment given to an object would thus be pointless.

To assess the accuracy of SOFIA in giving recommendations, we instead performed the following experiment on both datasets: we "hid" one random edge $A \rightarrow X$ from the judgment network, run SOFIA on the modified network, and used its output (i.e., a vector of weights) to rank all judgments from A's viewpoint; this is equivalent to producing recommendations, tailored to A, based on the computed ranking of judgments. Since X is a judgment that A expressed (before we hid it), A obviously approves of it, so a good recommendation engine should return X at a very high ranking. Thus, the highest the position of X in the ranked list of judgments, the better the accuracy of the ranking algorithm. In the Citeseer dataset, the experiment is equivalent to guessing a missing citation from a paper; in Last.fm, it means finding the missing artist in the top-50 chart of a user. In the following, all the results shown (for a given algorithm and set of parameter) were computed from 1,000 individual instances of the experiment.

The first set of experiments aimed at analysing the impact that the two different trust propagation patterns (transitivity and co-citation) individually had on prediction accuracy; at the same time, we wanted to quantify the effect that different choices of parameters had on it (namely α, β and the number of iterations). We thus separated the two "halves" of SOFIA into:

Personalised PageRank-Based Recommendations (PPRR). Each user u is first ranked using PPR; the ranking r_u is then simply divided between all the judgments u has expressed (proportionally to the edge weight). PPRR thus enables us to measure the impact of trust transitivity, while disregarding the network of judgments;

Non-SOcial FIltering Algorithm (N-SOFIA). All nodes in the web of trust are given equal intent ranking, instead of relying on the Personalised PageRank output. N-SOFIA thus enables us to study the impact of the co-citation pattern while disregarding the social network.

We chose to use this algorithms as opposed to those available in literature in order to specifically capture the relevance of the co-citation and transitivity patterns when comparing them with SOFIA, as opposed to the differences in ranking due to the particularities of the algorithms.

The first parameter we have studied is the *number of iterations* needed to obtain satisfying results. Table 6.1 shows the percentiles of the ranking of the "hidden" judgments, when running both PPRR and N-SOFIA on the Citeseer dataset, with α and β parameters chosen to optimize the results. As the table shows, a rather small number of iterations is enough to obtain very good results: for instance, after 10 iterations, 10% of the hidden judgments can be found in the top 2 returned results (i.e., recommendations) of PPRR, and at the very top for N-SOFIA; half of the hidden judgments (50th percentile) were returned within the top 29 recommendations made by PPRR, and in the top 11 by N-SOFIA, and so on[3]. Neither algorithms benefit from increasing the number of iterations after a small threshold. Rather, a higher number of iterations results in slightly worse results; our interpretation of this phenomenon is that, even if the weights on long paths are very low, they still introduce some "noise" since recommendations coming from longer paths are less reliable. In the following, the number of iterations for both parts of the algorithm has been set to 5.

We then studied the impact that parameters α and β had on the accuracy of PPRR and N-SOFIA on the specific datasets at hand[4]. Table 6.2 and 6.3 report the results for different values of α on PPRR, and of β on N-SOFIA, respectively. The key observation obtained from these numbers is that, on both datasets, N-SOFIA performs better than PageRank, suggesting that the information on tastes is more valuable than the information that can be inferred from the social network. On both datasets, the optimal value for β is much lower than the optimal value for α, suggesting that taste similarity propagates effectively on

[3]Note that the judgments returned with ranking higher than of X are not mistakes: they are simply other recommendations that these algorithms compute but, given that such judgments were never made by A (unlike X), we have no way of measuring how accurate those are.

[4]Note that a single optimal choice of these parameters does not exist, as they intrinsically depend on the characteristics of the dataset (in terms of "level of transitivity").

Dataset	α	Ranking percentiles					
		5	10	25	50	75	90
Citeseer	0.2	**1**	**2**	**8**	33	132	3,076
	0.3	**1**	**2**	**8**	**30**	**115**	**1,709**
	0.85	2	4	11	48	242	3,473
Last.fm	0.3	**5**	14	75	361	**2,107**	**15,064**
	0.5	**5**	**12**	**66**	**344**	2,188	16,025
	0.85	**5**	14	71	367	2,289	15,648

Table 6.2: Impact of α on hidden judgment ranking with Personalised PageRank.

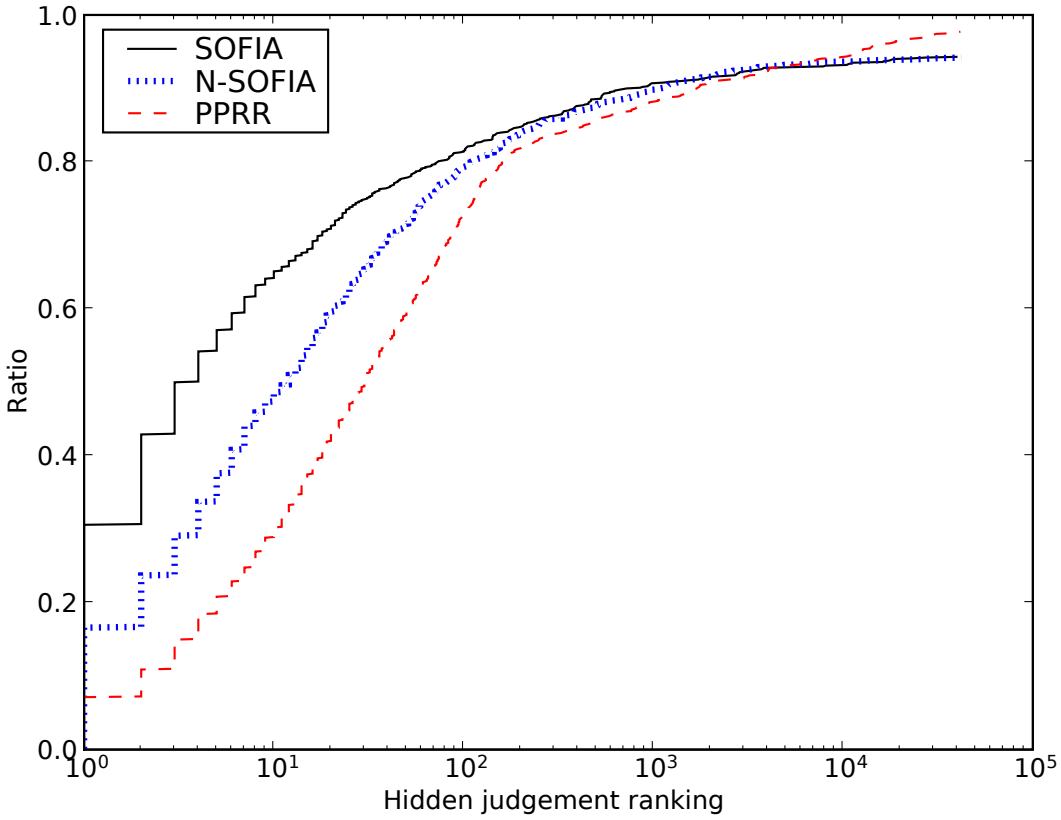
Dataset	β	Ranking percentiles					
		5	10	25	50	75	90
Citeseer	0.02	**1**	**1**	**3**	14	87	2,820
	0.05	**1**	**1**	**3**	**12**	**63**	**1,136**
	0.3	**1**	**1**	4	17	93	1,603
Citeseer (CF)	–	**1**	**1**	**3**	15	88	–
Last.fm	0.01	**2**	**6**	**32**	**157**	**822**	**3,954**
	0.1	5	13	58	269	1,305	10,599
	0.3	8	20	89	404	1,742	9,878
Last.fm (CF)	–	3	8	36	204	1,061	7,735

Table 6.3: Impact of β on hidden judgment ranking with N-SOFIA.

short paths only. Also, the optimal values for α are remarkably lower in our experiments than the "traditional" recommended $\alpha = 0.85$ for PageRank, reflecting the fact these datasets reward higher subjectivity. We have also compared the accuracy of N-SOFIA with traditional Collaborative Filtering techniques (in particular, using the cosine-based similarity measure): given that N-SOFIA produces recommendations based only on the network of judgments, while discarding social relations, we expect N-SOFIA and traditional CF to exhibit similar accuracy. As table 6.3 illustrates (rows labeled CF), the accuracy is indeed comparable on both datasets. Note that attacks have not been considered yet: once introduced (section 6.4.3), results will change dramatically, with approaches based on competence only (i.e., Collaborative Filtering-like techniques) suffering the most.

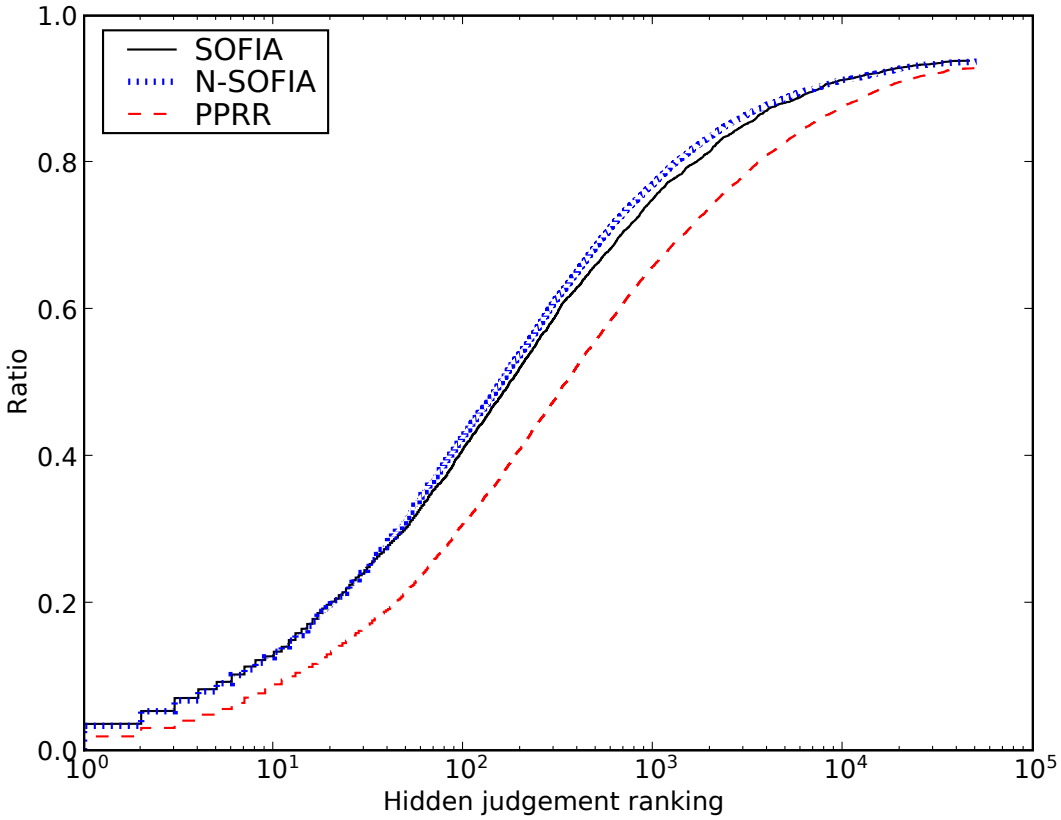
As a final set of experiments, we have compared the accuracy of PPRR and N-SOFIA with SOFIA, under the best choice of parameters for both datasets. Results are shown in Figures 6.3 and 6.4, for Citeseer and Last.fm respectively (the graphs plot the cumulative distribution function for the ranking of hidden judgments). Using the Citeseer dataset, SOFIA outperforms both algorithms, with 50% of the hidden judgments being ranked in the top 4 positions, against 12 for N-SOFIA and 30 for PPRR. The accuracy gain of SOFIA is

perhaps more striking when considering up to 75% of the hidden judgments: using SOFIA, a user would find the hidden judgment in the the top-30 list of recommended papers, while using PPRR the top-115 would have to be investigated. Of particular relevance is the observation that, even now that malicious attacks are *not* considered, SOFIA outperforms N-SOFIA (and traditional CF), despite the fact that SOFIA throws away (potentially useful) information coming from (honest) socially far-away nodes. This means that SOFIA



Algorithm	Ranking percentiles						
	5	10	25	50	75	90	95
SOFIA	**1**	**1**	**1**	**4**	**31**	**855**	–
N-SOFIA	**1**	**1**	3	12	63	1,136	–
PPRR	**1**	2	8	30	115	1,709	**11,609**

Figure 6.3: Hidden judgment ranking comparison on the Citeseer dataset. The α and β parameters were tuned for best performance ($\alpha = 0.5$, $\beta = 0.3$ for SOFIA, $\beta = 0.05$ for N-SOFIA, $\alpha = 0.3$ for PPRR).



Algorithm	Ranking percentiles						
	5	10	25	50	75	90	95
SOFIA	**2**	**6**	**32**	174	992	7,429	–
SOFIA (2)	3	8	46	240	1,347	11,919	–
N-SOFIA	**2**	**6**	**32**	**157**	**822**	**6,954**	–
PPRR	5	12	66	344	2,188	16,025	–

Figure 6.4: Hidden judgment ranking comparison on the Last.fm dataset ($\alpha = 0.9$ and $\beta = 0.05$ for SOFIA, $\alpha = 0.5$ and $\beta = 0.1$ for SOFIA (2), $\beta = 0.01$ for N-SOFIA, $\alpha = 0.5$ for PPRR).

effectively exploits knowledge gathered from the social network to counter-balance this loss of data, and the gain is higher than the cost for datasets that, like Citeseer, exhibit the intrinsic property of having "socially close" nodes more likely to share tastes.

The performance gain of SOFIA on the Last.fm dataset is less striking. As Figure 6.4 demonstrates, SOFIA still outperforms PPRR by a factor of 2. However, the performance

of SOFIA and N-SOFIA are almost indistinguishable: with this dataset, the two aforementioned effects – the loss of data that SOFIA suffers from not considering far away nodes and the added knowledge it gathers from the social network – balance each other out. However, even in these circumstances, we argue that running the whole SOFIA, instead of N-SOFIA alone, pays off: as we shall demonstrate in the next section, once attacks are in place, SOFIA outperforms N-SOFIA by far, thus yielding the best results overall in terms of accuracy *and* robustness. Note that the table in Figure 6.4 also reports the results of running SOFIA on an additional set of parameters, in particular, with a lower value of α; while accuracy becomes worse, we shall demonstrate, in the next section, that robustness to attacks becomes better, as shorter paths are considered, thus reducing the chance of traversing an attack region.

6.4.3 Robustness

As discussed in section 6.3, we are interested in evaluating how much an attacker, with the ability of creating an unlimited number of fake nodes, can raise the ranking of a given judgment X. We assume that, while it is relatively cheap to create a fully connected Sybil sub-network, it is costly for any Sybil node to enter the social network of an honest node (i.e., to be directly trusted by an honest user). We have thus designed our experiments as follows: we have created a completely connected Sybil sub-network of 100 nodes, and attached it to the honest part of the web of trust with a parametric number k of *attack edges*; each attack edge is given a weight of 1, and the honest node to which it connects is chosen at random. All Sybil nodes copy all the judgments given by a random "victim" V, and then create another edge towards a malicious judgment X (in Last.fm, where judgments are weighted, the weight is set as the maximum between the judgments of the victim). We then study how the ranking of X changes, before and after the attack, both on V and on other random nodes in the network, *for different values of* k. Once again, for each algorithm and set of parameters, the results have been obtained with 1,000 instances of the experiment. Note that the strength of an attack on traditional CF techniques has usually been measured in terms of the proportion of malicious nodes in the network [MBBW07]; however, the number of Sybil nodes is not relevant for PPR- and SOFIA-like algorithms, where the impact of the attacker is limited by the total ranking of the Sybil region. We have thus fixed the number of Sybils to 100 in all experiments, while varying parameter k, which does influence the ranking of the Sybil region instead.

Tables 6.4 and 6.5 shows how the ranking of malicious judgment X varies, with respect to parameter k, when enacting the attack respectively on the Last.fm and Citeseer datasets. The α and β parameters were the same as those used for the experiments shown in figures 6.4 and 6.3. The first row of the table shows the ranking of X when no attack is in place.

Algorithm	k	Role	Percentiles						
			5	10	25	50	75	90	95
Any		no attack	2,583	5,165	12,914	25,827	38,741	46,489	49,071
N-SOFIA		victim	1	1	1	1	1	1	1
		other	34	85	348	1,185	3,132	5,875	7,482
CF		victim	1	1	1	1	1	1	1
		other	25	54	214	1,522	27,367	–	–
PPRR	1		2,297	4,459	10,730	20,493	33,322	–	–
	10		1,285	2,353	4,759	8,757	13,371	19,648	26,846
	100		334	559	1,092	2,012	3,101	4,434	5,290
SOFIA	1	victim	679	1,386	3,406	11,182	31,765	–	–
		other	2,264	4,409	9,599	19,186	33,064	–	–
	10	victim	41	132	469	1,311	2,815	7,039	34,725
		other	1,082	2,126	4,612	8,779	14,718	22,254	26,959
	100	victim	1	2	13	74	197	377	564
		other	215	391	1,040	2,649	5,571	8,395	10,179
SOFIA (2)	100	victim	15	46	138	353	697	1,042	1,234
		other	448	705	1,578	3,106	5,128	7,447	9,187

Table 6.4: Ranking of the "malicious judgment" after a Sybil attack on the Last.fm dataset ($\beta = 0.01$ for N-SOFIA, $\alpha = 0.05$ for PPRR, $\alpha = 0.9$ and $\beta = 0.05$ for SOFIA, $\alpha = 0.5$ and $\beta = 0.1$ for SOFIA (2)).

No value is reported in the tables when the ranking for the malicious judgment is 0. As it can be seen, this happens quite often especially for the Citeseer dataset where most papers in the set did not receive a single citation before the attack.

Let us consider N-SOFIA and cosine-based collaborative filtering (CF) first. The main difference between the two algorithms is that that N-SOFIA propagates trust over many steps with co-citation trust propagation; CF, instead, just uses one step. This results in a lower coverage (items that can be rated), and the results confirm this, since often CF is unable to rank the malicious judgment. Apart from this phenomenon, the two algorithms behave quite similarly. Since they do not take into account the social network, the number of attack edges k is irrelevant in these cases. As shown, the malicious judgment X comes always at the very top of the recommendations made for the victim node V, even though, before the attack, such judgment was in position 2.5K or above! The ranking of X becomes high even for nodes who are not specifically under attack, thus confirming the fact that both N-SOFIA and traditional collaborative filtering techniques based on taste similarity only are highly vulnerable to Sybil attacks. Note also that, unlike PPRR and SOFIA, algorithms that do not limit the amount of trust that a Sybil region can gain, will suffer more by (cheaply) increasing the number of Sybil nodes.

Algorithm	k	Role	Percentiles						
			5	10	25	50	75	90	95
Any		no attack	9,134	18,268	45,669	91,338	137,006	164,408	173,541
N-SOFIA		victim	1	1	1	1	1	1	1
		other	1,218	2,614	7,213	17,407	30,267	42,019	–
CF		victim	1	1	1	1	1	1	1
		other	–	–	–	–	–	–	–
PPRR	1		6,799	12,343	29,613	–	–	–	–
	10		2,692	5,122	10,943	21,440	32,878	45,441	–
	100		546	933	1,933	3,888	6,624	9,990	12,168
SOFIA	1	victim	138	476	2,813	–	–	–	–
		other	6,352	12,465	28,796	–	–	–	–
	10	victim	2	13	85	328	1,074	16,579	–
		other	3,558	6,541	13,167	25,001	37,070	–	–
	100	victim	1	2	13	49	120	241	332
		other	1,762	3,002	7,056	14,630	26,459	41,969	–

Table 6.5: Ranking of the "malicious judgment" after a Sybil attack on the Citeseer dataset ($\beta = 0.05$ for N-SOFIA, $\alpha = 0.3$ for PPRR, $\alpha = 0.5$ and $\beta = 0.3$ for SOFIA).

On the contrary, the impact of the attack on Personalised PageRank is marginal. In this case, being a victim is indistinguishable from being any node in the network, given that individual opinions are not taken into consideration. Even when the Sybil region has conquered 100 attack edges, the ranking of the malicious judgment X is at position 2000 or above in 50% of the cases.

The robustness of SOFIA is comparable to that of PPRR when considering non-victim nodes. The victim node clearly suffers instead, but much less than when using N-SOFIA: for example, when the Sybil region has 10 attack edges to the honest part of the network, 50% of the times the malicious judgment X is ranked at around position 1300 or above by the victim node using SOFIA, instead of position 1 using N-SOFIA. The impact of the attack becomes non-negligible for victim nodes running SOFIA once the number of attack edges reaches $k = 100$. Note, however, that this is a rather costly attack: in fact, it requires tricking 1% of the 10,000-node network into trusting dishonest nodes, and all this effort just to change the ranking of judgment X by a single node V, with X only gaining marginally in other nodes' viewpoints. This result supports the claim we made at the end of the previous section, that is, that running SOFIA pays off, as its accuracy is at least as good as that of N-SOFIA, but its robustness to Sybil attacks is ways higher. Last but not least, it is worth observing the impact of different choices of parameters on the robustness of SOFIA; the last set of results shown in table 6.4 are obtained using the alternative set of parameters for SOFIA that were specified in figure 6.4: while the accuracy of the

recommendations using this second set of parameters was shown to be worse, the use of a lower α value makes the system more attack-resilient. As expected, there is a trade-off between accuracy and robustness, and the desired balance between the two features can be obtained by adjusting the parameters to the specific characteristics and requirements of the domain at hand.

Chapter 7

Current Work and Conclusions

The techniques presented in this work are all focused on the efficient exploitation of webs of trust. In chapters 4, 5, and 6, we introduced them separately. We are currently working on the natural evolution of this study: combined approaches that use these techniques in conjunction, to obtain even more significant results. In this chapter, we present preliminary results on enhancing the effectiveness of neighborhood maps by also using information on network maps (section 7.1) and on a decentralized implementation of SOFIA suitable for P2P systems, using neighborhood maps (section 7.2). In section 7.3, we conclude this thesis with a recapitulation of our contributions to the state of the art.

7.1 Combining Network and Neighborhood Maps

In chapter 4, we showed how neighborhood maps, local representations of the surroundings of a node in the web of trust, can be used to find trust paths and approximate reputation metrics. In chapter 5, instead, we focused on the network map, which is a distributed representation of the whole web of trust; it provides another method to discover trust paths efficiently and in a decentralized way.

It is quite natural to think about using the two techniques together: the network map, similarly to the map of a whole country, is helpful to discover the long-range routes that help us getting close to the destination; a neighborhood map, analogously to the map of a town, can then be used to efficiently reach the destination when we are close to it.

Following this intuition, we took in consideration the case where neighborhood maps are definitely smaller than the sizes (comparable to $\sqrt{n}$) used in chapter 5. When node A wants to compute PPR to estimate the trustworthiness of peer B, they are likely to have no common node in their neighborhood maps; however, both A and B can store the

network map coordinates of each node in their neighborhood maps, and use them to obtain additional information about the network. Nodes whose coordinates are "in the middle" between A and B are more promising, since they are more likely to be touched by short paths from A to B. Therefore, neighborhood maps can be retrieved from these nodes, and the desired metrics can be computed by combining the maps of A, B, and all these nodes between them.

Algorithm 9 returns a subset of the web of trust according to this idea. It starts with a sub-graph of the web of trust corresponding to the union of the neighborhood maps of A and B; iteratively, for a parametric number k of iterations, the node X in the current sub-graph that is closest to the halfway point from A to B in the network map gets contacted; the current subset of the WoT gets then updated with the neighborhood map of X. After the k iterations, the algorithm returns the computed graph, which can be used to approximate the desired reputation metric.

Algorithm 9 Return a subset of the WoT to approximate $R(A, B)$.

Parameters: two nodes A and B; the number k of neighborhood maps to be joined from other nodes.
Return value: a sub-graph S of the web of trust suitable to compute an approximation of $R(A, B)$.

/ We assume the function $M(X)$ retrieves from node X and returns a graph (V_X, E_X) corresponding to the neighborhood map of X; the function $P(X)$ returns a mapping from V_X to $\mathbb{R}^d$ returning the d-dimensional coordinates of nodes in V_X in the network maps. */*

$S \leftarrow M(A) \cup M(B)$ */* In S we will build the return value */*
$L \leftarrow P(A) \cup P(B)$ */* L contains the network map positions of all nodes in S */*
$m \leftarrow \frac{L[A]+L[B]}{2}$ */* Middle point between A and B */*

for $i \in 1 \dots k$ **do**
  $N \leftarrow$ node in S that minimizes $|L(N) - m|$
  $S \leftarrow S \cup M(N)$
  $L \leftarrow L \cup P(N)$
end for

return S

Experimental Results We repeated the experiment on Personalized PageRank approximation of section 4.5.3, using the undirected version of the OpenPGP web of trust and its

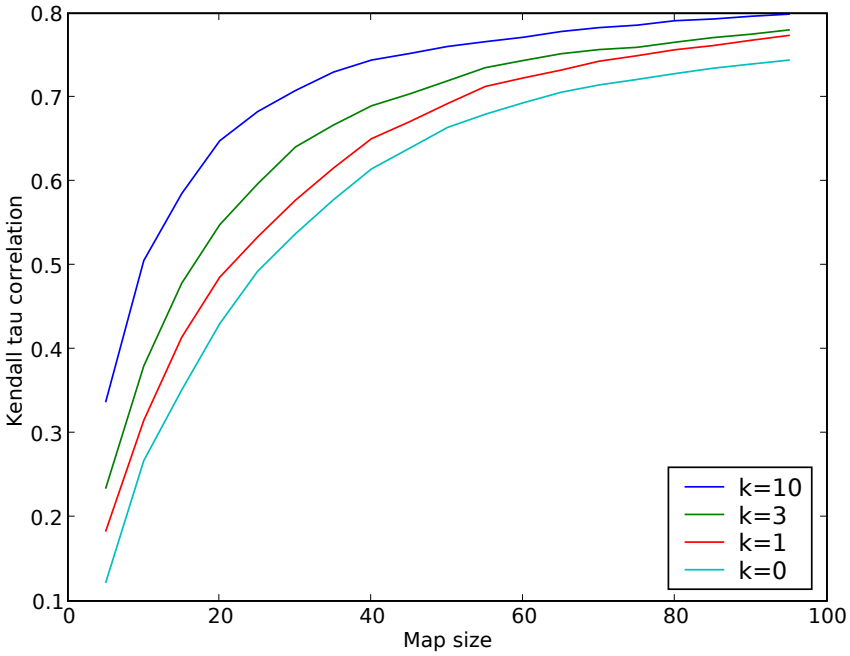


Figure 7.1: Correlation for PPR computed on the web of trust sub-graphs produced by algorithm 9, in function of the number of contacted nodes k. It is possible to obtain high correlation between PPR on the subsets and on the whole web of trust even when the map size is definitely smaller than $\sqrt{n} \simeq 162$.

layout computed for section 5.4. In figure 7.1, the results show that this technique allows to obtain high correlation with the correct ranking even when the neighborhood map size is much lower than $\sqrt{n}$.

Further work on larger networks is needed to understand how the size of neighborhood maps is related to correlation values and the size of the network (i.e., to obtain the same correlation when the network size grows, do maps need to grow faster or slower than $\sqrt{n}$?).

7.2 Decentralized SOFIA

In chapter 6, we introduced the SOFIA algorithm to give users good recommendations while retaining security against malicious attacks. The algorithm requires knowledge of the whole web of trust and judgements, and as such it is unreasonable to implement on a large-scale P2P network.

In this section, we will show a solution that uses neighborhood maps to approximate SOFIA in a scalable way. The SOFIA algorithm needs a Personalized PageRank approximation that can be computed using neighborhood maps.

Algorithm 10 Decentralized SOFIA.

Parameters: the map size s, the number of nodes m contacted at each iteration, the number of iterations k.
Return value: a pair of vectors $\langle t, \hat{t} \rangle$ such that t_u is the trust ranking of user u, and $\hat{t}_j$ is the trust ranking of judgment j.

On each node N, execute the following code:

/ We assume the PPR(X) function is given. It uses neighborhood maps to estimate the PPR values for node X (as seen by N). */*

$M \leftarrow$ the s nodes with highest PPR value in the PPR map of N
$J \leftarrow$ all judgments expressed by the nodes in M
$R \leftarrow$ mapping from X to PPR(X) for each X in M

for $i \in 1 \ldots k$ **do**
  $M' \leftarrow M; J' \leftarrow J$
  for all node X in the highest-ranked m nodes in M **do**
    Update M' and J' with the values of M and J in node X
    Update R with the PPR values of new nodes
  end for
  $\langle t, \hat{t} \rangle \leftarrow$ SOFIA (see algorithm 8 on page 96) on nodes M', judgments J' and reputation values R'
  $M \leftarrow$ the s nodes of M' with highest ranking (represented by t) in M'
  $J \leftarrow$ the judgements in J' expressed by users in M
end for

return $\langle t, \hat{t} \rangle$

In algorithm 10, we give a decentralized implementation of SOFIA. The algorithm assumes that neighborhood maps for Personalized PageRank are used; judgments expressed by nodes in the maps can be transmitted along with the maps themselves during the PPR map construction steps. Neighborhood maps for SOFIA (containing the information represented by the variables M and J in the algorithm) are initialized with the highest-ranked nodes in the PPR maps. Iteratively, the maps on the m highest-ranked nodes in the SOFIA map are downloaded; SOFIA is then computed using the information downloaded for all

of them. The s nodes that will remain in the map for the following iteration are the ones with the highest SOFIA user ranking.

This algorithm, in practice, explores the web of trust in order to discover nodes with higher SOFIA rankings at each step. While SOFIA is computed multiple times, the complexity of the algorithm is lower due to the fact that only a small number of user and judgments are present in the maps.

Experimental Results We compared the effectiveness of centralized SOFIA and this decentralized variant on the Last.fm dataset (see section 6.4.1). The parameters are those that we found to guarantee better results for SOFIA: $\alpha = 0.9, \beta = 0.1$. We implemented Personalized PageRank with the compact neighborhood maps described in section 4.3.2 (map size is $3\sqrt{n} = 300$). While more accurate results can be obtained with standard maps, we chose to use this solution to reduce the computing overhead of the simulation.

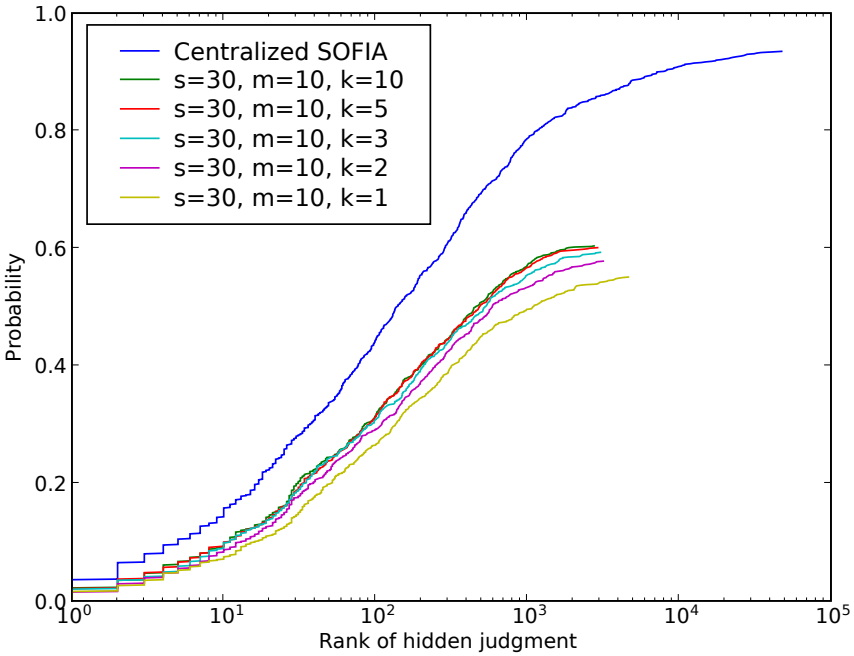
In the experiment, 1,000 random judgements have been hidden, and the ranking of hidden judgements is displayed. Differently from the experiments of section 6.4.2, where the judgements were hidden one-by-one, in this case they were hidden together.

In the following, we discuss our preliminary results relative to the impact on the ranking of hidden judgments for the parameters k, s, and m. While the results are noticeably worse than the ones obtained with centralized SOFIA, we must take in consideration the fact that the results are produced by working on amounts of data which are much smaller than the whole dataset. Smaller datasets also affect the coverage (percentage of ranked judgements); thus, using our decentralized version, the hidden judgements appeared in the ranking less often.

Impact of the number of iterations k. The number of iterations is a parameter that influences the length of the exploration: how many times new nodes need to be contacted before the SOFIA maps converge to the best nodes. Figure 7.2 shows quite clearly that the improvement of the quality of ranking is observed for the first few iterations; executing more than five iterations yields no noticeable improvement.

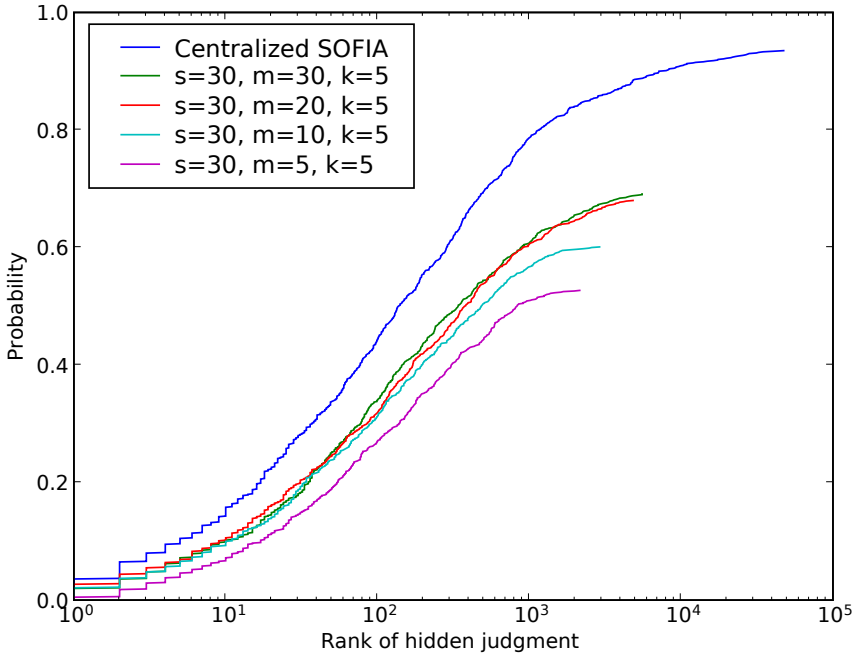
Impact of the number of contacted nodes m. The m parameter influences the number of nodes that get contacted at each step to build the new sub-graph for SOFIA: obviously, high m increases the network traffic due to the algorithm, but gives more information to the nodes. In figure 7.3, we notice that, for $s = 30$, results close to the optimum are obtained with $m = 20$ nodes. Further data is needed to provide sound advice on the choice of m when varying s.

Impact of the map size s. In figure 7.4, finally, we compare hidden judgement ranking with different values for the map size s. As could easily be expected, better results



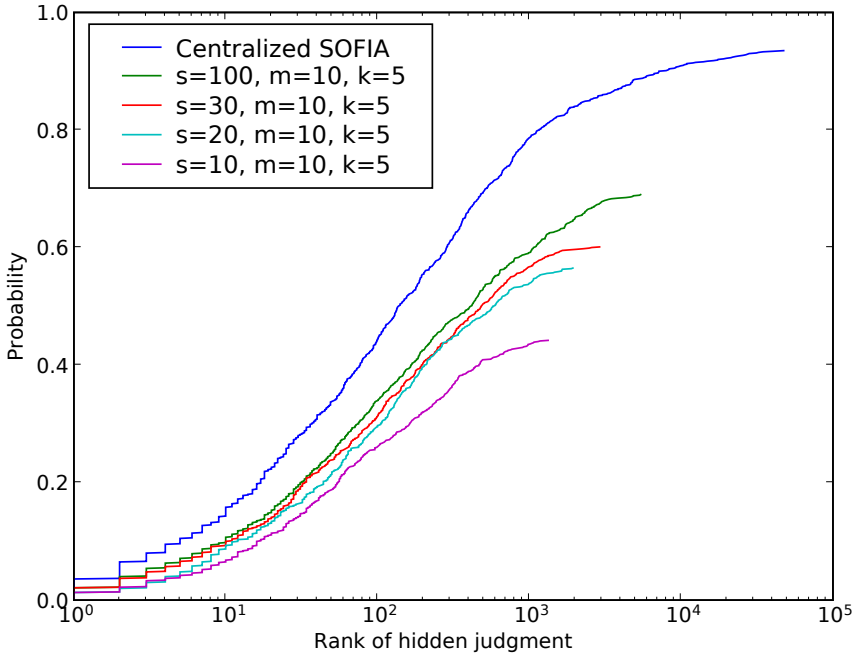
Parameters	Ranking percentiles			
	5	10	25	50
Centralized SOFIA	1	4	24	136
$s = 30,\ m = 10,\ k = 10$	3	10	55	455
$s = 30,\ m = 10,\ k = 5$	3	9	55	480
$s = 30,\ m = 10,\ k = 3$	4	10	55	525
$s = 30,\ m = 10,\ k = 2$	4	12	63	586
$s = 30,\ m = 10,\ k = 1$	4	15	82	1,083

Figure 7.2: Decentralized SOFIA and number of iterations k.



Parameters	Ranking percentiles			
	5	10	25	50
Centralized SOFIA	1	4	24	136
$s = 30,\ m = 30,\ k = 5$	3	9	48	338
$s = 30,\ m = 20,\ k = 5$	3	8	52	373
$s = 30,\ m = 10,\ k = 5$	3	9	55	480
$s = 30,\ m = 5,\ k = 5$	5	16	78	827

Figure 7.3: Decentralized SOFIA and number of contacted nodes m.



Parameters	Ranking percentiles			
	5	10	25	50
Centralized SOFIA	1	4	24	136
$s = 100,\ m = 10,\ k = 5$	2	9	49	409
$s = 30,\ m = 10,\ k = 5$	3	9	55	480
$s = 20,\ m = 10,\ k = 5$	5	11	63	574
$s = 10,\ m = 10,\ k = 5$	6	15	84	–

Figure 7.4: Decentralized SOFIA and map size s.

are returned with higher map sizes. However, we can notice that the increase on the quality gets lower and lower when s grows: for $s = 10$, the median rank for the hidden judgment is 409 when $s = 100$ and is 480 with $s = 30$.

Our results show that a decentralized implementation of subjective collaborative filtering technique seems reasonable, even if the loss in recommendation quality due to the smaller datasets appears noticeable. We are currently working on validating this technique with bigger map size s and other datasets.

7.3 Conclusions

In traditional systems, security is usually enforced in a centralized way, where administrators, or "super-users", have the right to decide which users can access (and modify) the resources. In completely decentralized P2P systems, an omnipotent user would not make sense: resources have different owners with very different goals; moreover, an authority which is responsible for security would be a vulnerability for the whole system. Webs of trust and reputation, on the other hand, introduce an inherently decentralized security paradigm where access privileges are dynamically earned by offering services to the peers.

In this thesis, we explored various issues and opportunities related to the use of webs of trust in decentralized systems. The main contributions can be summarized as follows.

- In chapter 3, we introduced a model for social networks that generalizes the well-known Barabási-Albert model. This model provides an explanation to common network properties such as being scale-free, small-world, and hierarchical.
- Chapter 4 tackles the problem of scalable and decentralized analysis of social networks. With neighborhood maps, nodes become able to approximate complex subjective reputation metrics, needing only a small part (with size proportional to the square root of the network size) of the information stored in the whole web of trust.
- Chapter 5 presents another approach to the problem introduced in the previous chapter. With our algorithm, nodes create a distributed representation of the network topology that makes it easy and scalable to perform "routing" on the web of trust: finding the short (and thus more significant) "trust paths" that are used to compute the reputation of nodes. With respect to the state of the art, our algorithm is able to find paths that are, in some cases, orders of magnitude shorter.
- In chapter 6, we introduced social filtering, an approach which uses webs of trust in collaborative filtering systems to provide security against attacks such as shilling,

even in the unfavorable case where attackers are able to create an unlimited number of fake identities.

- In this chapter, we outlined the direction taken in current research, combining the techniques of chapters 4, 5 and 6 to obtain more effective approximations and to implement social filtering in a decentralized way.

As future work, we plan to implement our techniques in a large-scale P2P network, in order to harness the benefits of reputation as an incentive to cooperation, and to use social filtering as an attack-resistant method to provide sound recommendations to users.

With this work, we presented techniques that deal with the issue of creating secure and useful applications that are decentralized both in the technological (P2P) and the cultural ("long tail") sense. We think that the "centrifugal force" that has been driving away from centralized structure many Internet applications is not over yet, and research in this field has the potential to harness inspiring results and promote many new successful applications.

Bibliography

[AB02] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.

[Abd07] Hervé Abdi. The kendall rank correlation coefficient. In Neil J. Salkind, editor, *Encyclopedia of Measurement and Statistics*. Sage, Thousand Oaks (CA), USA, 2007. Available from World Wide Web: `http://www.utdallas.edu/~herve/Abdi-KendallCorrelation2007-pretty.pdf`.

[AH00] Eytan Adar and Bernardo A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), 2 October 2000. Available from World Wide Web: `http://firstmonday.org/issues/issue5_10/adar/index.html`.

[AJB99] Reka Albert, Hawoong Jeong, and Albert-László Barabási. Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131, 1999.

[ALPH01] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in power-law networks. *Physical Review E*, 64(4):046135+, 2001.

[AMT05] Paolo Avesani, Paolo Massa, and Roberto Tiella. Moleskiing.it: a trust-aware recommender system for ski mountaineering. 2005. Available from World Wide Web: `http://www.i-journal.org.uk/Papers/TRECK/2005/AVESANI.pdf`.

[And06] Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.

[Ati02] Y. Atif. Building trust in e-commerce. *IEEE Internet Computing*, 6(1):18–24, 2002.

[Axe84] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[BA99] Albert-László Barabási and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[BKM+00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 33(1):309–320, June 2000.

[BKR05] P. Blanchard, T. Krueger, and A. Ruschhaupt. Small world graphs by iterated local edge formation. *Physical Review E*, 71(046139), 2005.

[Bol01] Béla Bollobás. *Random Graphs*. Cambridge University Press, January 2001.

[Bri00] Dan Bricklin. The cornucopia of the commons: How to get volunteer labor. `http://www.bricklin.com/cornucopia.htm`, August 2000.

[BRRT05] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. Link analysis ranking: Algorithms, theory, and experiments. *ACMTIT: ACM Transactions on Internet Technology*, 5, 2005.

[BW00] A. Barrat and M. Weigt. On the properties of small-world network models. *The European Physical Journal B - Condensed Matter*, 13(3):547–560, January 2000.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.

[Cac04] CacheLogic. The true picture of peer-to-peer filesharing. `http://www.cachelogic.com/research/p2p2004.php`, July 2004.

[Cac05] CacheLogic. Peer-to-Peer in 2005. `http://www.cachelogic.com/research/p2p2005.php`, 2005.

[CDFT98] Jon Callas, Lutz Donnerhacke, Hal Finney, and Rodney Thayer. OpenPGP message format. Internet RFC 2440, IETF, 1998. Available from World Wide Web: `http://www.ietf.org/rfc/rfc2440.txt`.

[Ced04] Jörgen Cederlöf. Dissecting the Leaf of Trust. `http://www.lysator.liu.se/~jc/wotsap/leafoftrust.html`, 2004.

[CF05] Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Proceedings of Third Workshop on Economics of Peer-to-Peer Systems (P2PECON 2005), Philadelphia, PA.*, 22 August 2005.

[Chu97] F. R. K. Chung. Spectral graph theory. *Regional Conference Series in Mathematics, AMS*, 92:1–212, 1997.

[Cia06] Giuseppe Ciaccio. Improving sender anonymity in a structured overlay with imprecise routing. In George Danezis and Philippe Golle, editors, *PET 2006 Revised Selected Papers*, volume 4258 of *LNCS*, pages 190–207. Springer, 2006.

[CLP07] Ciro Cattuto, Vittorio Loreto, and Luciano Pietronero. Semiotic dynamics and collaborative tagging. *PNAS*, 104(5):1461–1464, 30 January 2007.

[CMH+02] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[Coh03] Bram Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA*, 2003.

[DC07] Matteo Dell'Amico and Licia Capra. SOFIA: Social filtering for robust recommendations. Technical Report RN/07/20, University College London, 2007. Available from World Wide Web: `http://www.cs.ucl.ac.uk/research/researchnotes/documents/RN_07_20.pdf`.

[DDCdVP+02] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In Vijayalakshmi Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, DC, USA*, pages 207–216. ACM, November 2002.

[Del06a] Matteo Dell'Amico. Highly clustered networks with preferential attachment to close nodes. In *Proceedings of the Second European Conference on Complex Systems (ECCS 2006), Oxford, United Kingdom*, November 2006.

[Del06b] Matteo Dell'Amico. Highly clustered networks with preferential attachment to close nodes. Technical Report DISI-TR-06-06, DISI, Università di Genova, 6 April 2006. Available from World Wide Web: `http://www.disi.unige.it/research/expand-techrep?id_tr=30`.

[Del06c] Matteo Dell'Amico. Neighbourhood maps: Decentralised ranking in small-world P2P networks. In *Proceedings of 3rd International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P 2006), Rhodes, Greece*, April 2006.

[Del07] Matteo Dell'Amico. Mapping small worlds. In *Seventh IEEE International Conference on Peer-to-Peer Computing, 2007 (P2P 2007), Galway, Ireland*, pages 219–228, 2007.

[Del08] Matteo Dell'Amico. Neighbourhood maps: decentralized ranking in small-world P2P networks. *Concurrency and Computation: Practice and Experience*, 20(6):659–674, 25 April 2008. Available from World Wide Web: `http://dx.doi.org/10.1002/cpe.1245`.

[DM06] Alessandro Di Marco. Effects of the decay factor in the BFS algorithm. Technical Report DISI-TR-06-05, DISI, Università di Genova, April 2006. Available from World Wide Web: `http://www.disi.unige.it/research/expand-techrep?id_tr=29`.

[DMW03] Peter Sheridan Dodds, Roby Muhamad, and Duncan J. Watts. An experimental study of search in global social networks. *Science*, 301:827–829, August 2003.

[Dou02] John R. Douceur. The Sybil attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA*, March 2002.

[Dut99] Prajit K. Dutta. *Strategies and Games*. MIT Press, 1999.

[eMu03] eMule credit system. `http://www.emule-project.net/home/perl/help.cgi?l=1&rm=show_topic&topic_id=134`, 2003.

[FC05a] M. Feldman and J. Chuang. The evolution of cooperation under cheap pseudonyms. In *Proceedings of Seventh IEEE International Conference on E-Commerce Technology (CEC 2005)*, pages 284–291, 2005.

[FC05b] Michal Feldman and John Chuang. Overcoming free-riding behavior in peer-to-peer systems. *SIGecom Exch.*, 5(4):41–50, July 2005.

[FC06] Eric J. Friedman and Alice Cheng. Manipulability of PageRank under sybil strategies. In *Proceedings of the First Workshop on Networked Systems (NetEcon06), Ann Arbor, MI, USA*, June 2006.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 251–262, New York, NY, USA, 1999. ACM.

[FiCS03] Ramon Ferrer i Cancho and Ricard V. Solé. *Statistical Mechanics of Complex Networks*, chapter Optimization in Complex Networks, pages 114–126. Springer, 2003.

[FLSC04] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of ACM Conference on Electronic Commerce (EC'04), New York, NY, USA*, May 2004.

[FR01] Eric J. Friedman and Paul Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics & Management Strategy*, 10(2):173–199, August 17 2001.

[GGM05] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *Proceedings of First International Workshop on Adversarial Information Retrieval on the Web (AIRWEB 2005)*, 2005.

[GKRT04] R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web (WWW2004), New York, NY, USA*, pages 403–412, May 2004.

[Gol05] Jennifer Ann Golbeck. Computing and applying trust in web-based social networks, April 11 2005. Available from World Wide Web: `http://hdl.handle.net/1903/2384`.

[Har68] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.

[Har04] Amy Harmon. Amazon glitch unmasks war of reviewers. The New York Times, 14 February 2004. Available from World Wide Web: `http://query.nytimes.com/gst/fullpage.html?res=9C07E0DC1F3AF937A25751C0A9629C8B63&sec=&spon=&partner=permalink&exprod=permalink`.

[HCH07] Farookh K. Hussain, Elizabeth Chang, and Omar K. Hussain. State of the art review of the existing pagerank based algorithms for trust computation. In *Second International Conference on Systems and Networks Communications (ICSNC 2007)*, page 75, 2007.

[HK02] Petter Holme and Beom Jum Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65(026107), 2002.

[HKBR99] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International Conference on Research and De-*

velopment in Information Retrieval (SIGIR '99), pages 230–237, New York, NY, USA, 1999. ACM Press.

[Ipo06] Ipoque. P2P survey 2006. `http://www.ipoque.com/media/internet_studies/p2p-studie_2006`, October 2006.

[Ipo07] Ipoque. Internet study 2007. `http://www.ipoque.com/media/internet_studies/internet_study_2007`, 2007.

[IRF04] Adriana Iamnitchi, Matei Ripeanu, and Ian Foster. Small-world file-sharing communities. In *Proceedings of INFOCOM 2004, Hong Kong, China*, March 2004.

[JHP06] Audun Jösang, Ross Hayward, and Simon Pope. Trust network analysis with subjective logic. In *ACSC '06: Proceedings of the 29th Australasian Computer Science Conference*, pages 85–94, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[Jö96] Audun Jösang. The right type of trust for distributed systems. In *NSPW '96: Proceedings of the 1996 workshop on new security paradigms, New York, NY, USA*, pages 119–131. ACM Press, 1996.

[KE02] Konstantin Klemm and Víctor M. Eguíluz. Highly clustered scale-free networks. *Physical Review E*, 65(036123), 2002.

[Ken38] Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938.

[KK89] Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 12 April 1989.

[KKR+99] Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. The web as a graph: Measurements, models, and methods. In *Proceedings of Computing and Combinatorics: 5th Annual International Conference (COCOON'99)*. Springer, July 1999.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM: Journal of the ACM*, 46:604–632, September 1999.

[Kle00a] Jon M. Kleinberg. Navigation in a small world. *Nature*, 406(6798), August 2000.

[Kle00b] Jon M. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170, 2000.

[Kle02] Judith Kleinfeld. The small world problem. *Society*, 39(2):61–66, January 2002.

[Kor03] Yehuda Koren. On spectral graph drawing. In *Proceedings of COCOON 2003: Annual International Conference on Computing and Combinatorics*, 2003.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of WWW 2003*, pages 640–651, 2003.

[KSS97] Henry Kautz, Bart Selman, and Mehul Shah. Referral web: combining social networks and collaborative filtering. *Commun. ACM*, 40(3):63–65, March 1997. Available from World Wide Web: `http://portal.acm.org/citation.cfm?id=245123`.

[Lan03] Marc Langheinrich. When trust does not compute – the role of trust in ubiquitous computing. Workshop on Privacy at Ubicomp 2003, Seattle, Washington, October 2003.

[Lev00] R. Levien. Advogato's trust metric. `http://www.advogato.org/trust-metric.html`, 2000.

[LFR06] Shyong Lam, Dan Frankowski, and John Riedl. Do you trust your recommendations? An exploration of security and privacy issues in recommender systems. *Emerging Trends in Information and Communication Security*, pages 14–29, 2006.

[LJ07] Kristina Lerman and Laurie Jones. Social browsing on flickr. In *Proceedings of International Conference on Weblogs and Social Media (ICWSM 2007)*, March 2007. Available from World Wide Web: `http://www.icwsm.org/papers/paper27.html`.

[LM01] R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, April 2001.

[LNNT05] David Liben-Nowell, Jasmine Novak, and Andrew Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, 102(33):11623–1162, 2005.

[LR04] Shyong K. Lam and John Riedl. Shilling recommender systems for fun and profit. In *WWW '04: Proceedings of the 13th international conference on World Wide Web, New York, NY, USA*, pages 393–402. ACM Press, 2004.

[LSB03] S. Lee, R. Sherwood, and Bobby Bhattacharjee. Cooperative peer groups in nice. volume 2, pages 1272–1282 vol.2, 2003.

[MA07a] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of ACM Recommender Systems Conference, Minneapolis, MN, USA*, 2007. Available from World Wide Web: `http://www.gnuband.org/papers/trust-aware_recommender_systems/`.

[MA07b] Paolo Massa and Paolo Avesani. Trust metrics on controversial users: balancing between tyranny of the majority and echo chambers. *International Journal on Semantic Web and Information Systems*, 2007.

[Mas05] Paolo Massa. Attacking HITS (and not PageRank). `http://www.gnuband.org/2005/03/31/attacking_hits_and_not_pagerank/`, March 2005.

[Mas07] Paolo Massa. Reputation is in the eye of the beholder: on subjectivity and objectivity of trust statements. Invited position paper at ENISA Workshop "Security Issues in Reputation Systems" at the eema's "The European e-identity conference", 2007. Available from World Wide Web: `http://www.gnuband.org/2007/06/09/reputation_is_in_the_eye_of_the_beholder_on_subjectivity_and_objectivity_of_trust_statements/`.

[MBBW07] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans. Inter. Tech.*, 7(4):23, 2007.

[MG01] Nisha Mathias and Venkatesh Gopal. Small worlds: How and why. *Physical Review E*, 63(021117), 2001.

[MIF02] R. Matei, A. Iamnitchi, and P. Foster. Mapping the Gnutella network. *Internet Computing, IEEE*, 6(1):50–57, 2002.

[Mil67] S. Milgram. The small world problem. *Psychology Today*, 1:61, 1967.

[MS02] S. S. Manna and Parongama Sen. Modulated scale-free network in Euclidean space. *Physical Review E*, 66(066114), 2002.

[MSK02] Manfred Milinski, Dirk Semmann, and Hans-Jürgen Krambeck. Reputation helps solve the 'tragedy of the commons'. *Nature*, 415:424–426, January 2002.

[New01] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(025102), 2001.

[New03] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[NS98] M. A. Nowak and K. Sigmund. Evolution of indirect reciprocity by image scoring. *Nature*, 393(6685):573–577, 1998.

[Ols02] Stefanie Olsen. Amazon blushes over sex link gaffe. CNET News.com, 6 December 2002. Available from World Wide Web: `http://www.news.com/2100-1023-976435.html`.

[O'R05] Tim O'Reilly. What is Web 2.0. `http://www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`, September 2005.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. Available from World Wide Web: `http://dbpubs.stanford.edu/pub/1999-66`.

[Pri76] Derek J. Price. General theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27:292–306, 1976.

[PRU02] Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Using PageRank to characterize web structure. In *Proceedings of the 8th Annual International Conference on Computing and Combinatorics (COCOON 2002)*, pages 330–339, 2002.

[RB03] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(026112), 2003.

[RSG06] Giancarlo Ruffo, Rossano Schifanella, and Enrico Ghiringhello. A decentralized recommendation system based on self-organizing partnerships. In Fernando Boavida, Thomas Plagemann, Burkhard Stiller, Cédric Westphal, and Edmundo Monteiro, editors, *Proceedings of NETWORKING 2006 - Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems, 5th International IFIP-TC6 Networking Conference, Coimbra, Portugal, May 15-19, 2006*, volume 3976 of *Lecture Notes in Computer Science*, pages 618–629. Springer, 2006.

[RWd+03] J. Rosenberg, J. Weinberger, dynamicsoft, C. Huitema, Microsoft, R. Mahy, and Cisco. STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs). Internet Request

for Comment RFC 3489, Internet Engineering Task Force, March 2003. Available from World Wide Web: `http://www.ietf.org/rfc/rfc3489.txt`.

[San06] Oskar Sandberg. Distributed routing in small-world networks. In *Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX 2006), Miami, FL, USA*, January 2006.

[SBR04] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury. Percolation search in power law networks: making unstructured peer-to-peer networks scalable. In *Proceedings of IEEE P2P2004*, pages 2–9, 2004.

[Sch96] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition*. Wiley, October 1996.

[SGM04] Q. Sun and H. Garcia-Molina. SLIC: a selfish link-based incentive mechanism for unstructured peer-to-peer networks. In *Proceedings of 24th International Conference on Distributed Computing Systems, 2004*, pages 506–515, 2004.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

[SMK+01] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, pages 149–160, 2001.

[SP73] Maynard J. Smith and G. R. Price. The logic of animal conflict. *Nature*, 246(5427):15–18, November 1973.

[TM69] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[WDN02] D. J. Watts, P. S. Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296:1302–1305, 2002.

[WS98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):397–498, June 1998.

[XBS02] R. Xulvi-Brunet and I. M. Sokolov. Evolving networks with disadvantaged long-range connections. *Physical Review E*, 66(026118), 2002.

[YJB02] Soon-Hyung Yook, Hawoong Jeong, and Albert-László Barabási. Modeling the Internet's large-scale topology. *Proceedings of the National Academy of Sciences of the United States of America*, 99(21):13382–13386, 15 October 2002.

[YKGF06] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Proceedings of ACM SIGCOMM 2006*, pages 267–278, 2006.

[ZH07] Runfang Zhou and Kai Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *Transactions on Parallel and Distributed Systems*, 18(4):460–473, 2007.

[Zim95] Philip R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.

[ZL04] C. N. Ziegler and G. Lausen. Spreading activation models for trust propagation. In *Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04)*, pages 83–97, 2004.