Neighbourhood Maps: Decentralised Ranking in Small-World P2P Networks

Matteo Dell'Amico[*]

Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova
E-mail: dellamico@disi.unige.it

Abstract

Reputation in P2P networks is an important tool to encourage cooperation among peers. It is based on ranking of peers according to their past behaviour.

In large-scale real world networks, a global centralised knowledge about all nodes is neither affordable nor practical. For this reason, reputation ranking is often based on local history knowledge available on the evaluating node. This criterion is not optimal, since it ignores useful data about interactions with other peers.

*We propose a simple, scalable and decentralised method, called "*neighbourhood maps*", that approximates rankings calculated using link-analysis techniques, exploiting the short-distance characteristics of small-world networks.*

We test our algorithms using data from the OpenPGP web-of-trust, a real-world network of trust relationships.

1. Introduction

Trust management is an interesting problem which naturally arises in many P2P applications.

In traditional client-server applications, trust (and, consequently, access to resources) is usually provided through authentication on a server. The server is assumed to know which the trusted users are.

Most P2P applications have instead a different approach: there is no central authority, and peers have to resort to other criteria in order to evaluate trust.

In those applications where such a problem is ignored, and each peer receives the same level of trust, there is an incentive for selfish behaviour (i.e., consuming the resources of other nodes, while not giving anything in return). An increase in free riding [4] then appears, in which the common resources are exploited at the expense of the whole peer community.

[*]Financially supported by the Italian MIUR under the framework of the FIRB 2001 action, WEB-MiNDS Project. http://web-minds.consorzio-cini.it/

The notion of reputation – a measure of how good the past behaviour of a node was – is introduced to solve this problem. Each node ranks peers according to an evaluation of the past interactions it had with them (in the following, we will refer to this kind of locally evaluated reputation values as *LRV*s).

When nodes receive requests from other peers, they adopt a more cooperative approach towards ones with a good reputation. An incentive towards a "good behaviour" thus emerges, since free riders receive a poor quality of service. This idea of reciprocative behaviour is inspired by Axelrod [6], and has been implemented in the BitTorrent [11] and EDonkey2000 [1] file-sharing networks, with great success[1].

Using this technique, peers can only have information about nodes that previously gave service to them. This approach, however, is not effective whenever there is an asymmetry of interest [13]: when node A wants service from B, it is possible that B does not want anything from A. Even in the lucky cases where both interacting nodes can provide service to each other[2], there is an initial period in which nodes need to build up reciprocal trust, and only afterwards they can enjoy mutual cooperation. Thus, reciprocative behaviour loses effectiveness when networks are large, interest is asymmetrical, and/or interactions have a short duration.

The notion of *indirect* reciprocative techniques is introduced to solve the aforementioned shortcomings. Whenever a node has no direct knowledge about a peer, it uses information collected by the other nodes. This allows nodes to calculate reputation for all peers in the network, enabling reciprocation even when direct approaches are not successful.

In this work we assume a *LRV graph* is given, having peers as nodes and LRV values as edges. Given an eval-

[1]The traffic generated by these two applications was estimated to be more than 80% of the total P2P traffic, and about 50% of total Internet traffic at end of 2004 [9].

[2]This is true in some important cases (e.g., nodes downloading a big file and sharing the already completed parts).

uating x and an evaluated y node, a *distributed reputation value* $DRV(x, y)$ is calculated, depending on the data in the LRV graph. The basic idea is that $DRV(x, y)$ depends on the paths from x to y in the LRV graph: given the assumption that a node which is trusted by a trusted node deserves some trust itself, it becomes apparent that paths on the network can be used to represent trust relationships, with shorter paths representing a more direct, and possibly more significant, relationship.

Our basic idea is that each node creates a "local view" of reputation by collecting information about the closest nodes – a *neighbourhood map*. These maps are constructed by repeatedly contacting directly-connected nodes and combining data received from them. The size of the map is parametric, and the concept of closeness itself depends on the metrics being evaluated.

When peer x wants to rank peer y, y's reputation is calculated by evaluating data based on the nodes known in both x's and y's neighbourhood maps. Since in small-world networks the average distance between couples of nodes is low, if neighbourhood maps are big enough, with high probability there will be an intersection between the two neighbourhood maps. Intersections represent middle points in paths from x to y, and data related to them is used to compute the desired metrics.

LRV graphs can be created using data representing any kind of recommendation between entities. Some examples could be nodes on the EDonkey2000 network and the credits between them, scientific papers and their citations, WWW pages and their links, and the PGP web of trust that will be described in Section 3.

Paper Structure The remainder of this paper is structured as follows.

In Section 2 we will have a glance at past work regarding reputation in P2P networks.

Section 3 will introduce the PGP web of trust, the real-world graph we used for our experiments.

Section 4 will introduce the most important issues in designing indirect reciprocative techniques in P2P systems.

Sections 5 and 6 will describe how to use neighbourhood maps in order to approximate respectively the shortest path between nodes and PageRank on a graph; experimental Monte-Carlo results will show the level of accuracy for our method.

In Section 7 we will outline our ideas for further work and experimentation using neighbourhood maps.

2. Related Work

DRVs are calculated by analysing the structure of the network. Our technique is inspired by the "link analysis" algorithms [8] used in web-search engine ranking, such as PageRank [21] or HITS [18].

In the P2P area, EigenTrust [17] is a distributed implementation of PageRank. A distributed structure for storing data (e.g., a distributed hash table such as Chord [23]) is needed. A weakness of this approach is that a set of nodes needs to be pre-trusted by all the nodes, thus reducing the degree of decentralisation of the network.

[13] advocates a criterion in which the DRV depends on the node evaluating it, and proposes the calculation of the maximal flow between the evaluating and the evaluated node to be used as DRV. Unfortunately, the proposed algorithm requires a complete knowledge of the network and no scalable solution for large networks is proposed.

Under the assumption that the preferences of the nodes can be different depending on the characteristics of the network, nodes can evaluate similarity in preferences and give a higher reputation to "similarly minded" nodes, as seen in [19, 20].

A different approach, suitable for unstructured P2P networks such as Gnutella [2] is a rewiring approach [3]: nodes constantly change their connections in the network whenever they are unsatisfied with their current neighbours, in order to get connected to better peers.

3. 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, since it is a significantly sized network of real-world trust relationships.

OpenPGP [10] is an open protocol for privacy and authentication, using asymmetric key cryptography. In OpenPGP, there is no central certification authority binding persons to their respective public keys. Conversely, 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 is called a *web of trust*, and can be seen as a 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 [16] on May 9, 2005, a graph having 27398 nodes and 246355 edges (with an average of 8.99 outgoing edges per node).

In figure 1 on the following page, the in-degree and out-degree distributions are plotted on a logarithmic scale, and they are shown to be approximated with a power-law distribution. The graph is also a small-world network, having a mean shortest distance (MSD) between nodes of 5.96. These characteristics of low average distance and power-law degree distribution are common to many kinds of real-

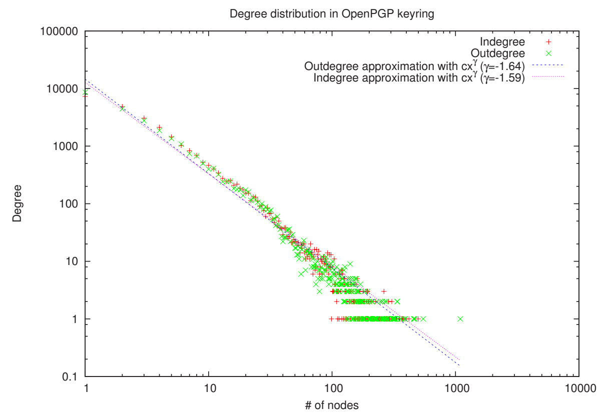

Figure 1. OpenPGP degree distribution

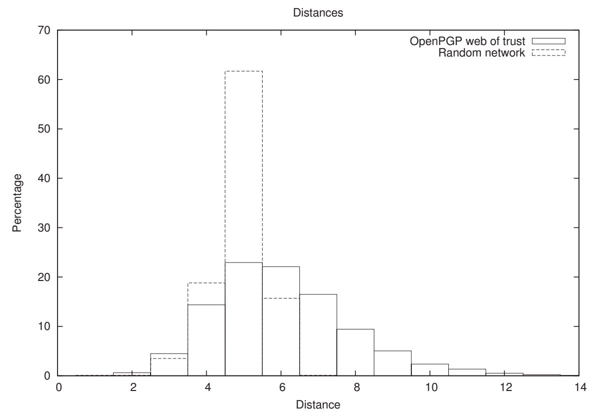

Figure 2. OpenPGP distance distribution

world networks [7], such as the Internet, the WWW, or the graph representing co-authorship of scientific papers.

In figure 2, the distribution of distances of this network is plotted against the one of a random network having the same number of nodes and edges (MSD 4.89). The random one has a much narrower shape, due to the absence of nodes with very high degree (hubs) and peripheral nodes in the random network.

The web of trust we are studying also exhibits quite a high clustering[3] value of 0.371. This feature, which is common among this kind of networks, poses us some problems, because this means that "close" pairs of nodes will have more intersections in the neighbourhood maps, and "far" ones will have fewer, resulting in a lower probability of finding intersections in more difficult cases.

[3]Clustering for a node n is defined, with k_n being the number of nodes connected to n via a an edge, as ratio between the numbers of edges between those k_n nodes and the maximal possible count of $k_n\,(k_n - 1)$. Clustering for a network is the average clustering for all nodes in that network.

4. Security Issues

While indirect reciprocative approaches are clearly more powerful than direct ones, adoption in P2P systems presents some important issues.

No Global Knowledge In networks having large size and/or a great number of interactions, it is unfeasible for peers to keep updated data about every interaction in the system. Our method is based on local knowledge. Another possibility is the use of decentralised data structures, like DHTs (distributed hash tables) such as Chord [23].

Cheap Identities In many cases, it is possible for new nodes to easily obtain new identities, effectively erasing their past history (*whitewashing*). Obviously, this can be used by nodes that had a malicious past behaviour. As [14] points out, whitewashing "…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".

Collusive Attacks A number of malicious nodes could introduce erroneous information in their history, in order to create attacks that aim to maliciously boost or decrease reputation of some nodes. Systems have to be designed in order to be resilient to this kind of attacks, giving more weight to data introduced by more reputable nodes.

Collusive attacks carried out by a great number of spurious identities, thanks to cheap identities, are known as *Sybil attacks* [12]. By focusing on paths connecting evaluating and evaluated nodes, we can design systems that are resilient to these kind of attacks [5, 13].

5. Estimation of Shortest Path

Graph distance is a simple way of evaluating trust for another node. 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. The concept of shortest path is indeed used as a reputation evaluation means in the PGP web of trust in [15, 16], and is resilient to Sybil attacks [5].

The construction of a neighbourhood map for distance, as shown in the following, can be done just by contacting neighbours in the network. Given a fixed size delimiter k, each peer stores the closest k nodes and their distances with regard both to incoming and outgoing paths. Finding an

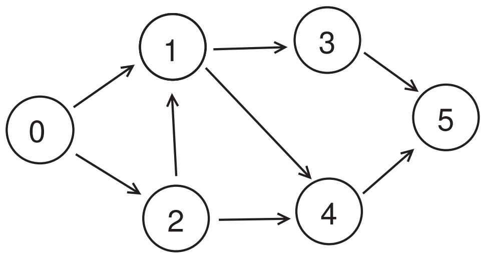

Figure 3. Example graph

intersection between node x's map for outgoing links and node y's map for incoming ones means that a path from x to y has been found.

Map Construction When edges are labelled with positive integer values, we can express a neighbourhood map as an array that contains, at index i, the set of nodes that are exactly at distance i. For instance, in the simple graph in figure 3 (where all edges are labelled with 1), the map for node 0 given a map size $k = 5$ would be $[\{0\}, \{1, 2\}, \{3, 4\}]$.

Algorithm 1 calculates the map for outgoing paths on a node n. Inputs are the size k of the neighbourhood map; a function $maps(\overline{n}, d)$, which returns the nodes that $\overline{n}$ has found at distance d; and a list, called $conn$, of $(node, distance)$ pairs representing n's outgoing edges.

For simplicity, we suppose the algorithm will be evaluated synchronously by all nodes in the network (i.e., all nodes simultaneously evaluate the same step of the outer **for** cycle). In a dynamically changing network, nodes would need to execute continuously this algorithm in order to keep maps up to date.

An object-oriented notation is adopted, with arrays having an $append$ method and indexing starting with 0.

Algorithm 1 Neighbourhood maps for distance

```
$my\_maps \leftarrow [\{my\_id\}]$
**for all** $dist \in 1 \ldots \infty$ **do**
  $m \leftarrow \{n' \mid (n, d) \in conn \wedge n' \in maps(n, dist - d)\}$
  $m \leftarrow m \setminus \left(\bigcup_{i=0}^{dist-1} my\_maps[i]\right)$
  $my\_maps.append(m)$
  $size \leftarrow$total number of elements in $my\_maps$
  **if** $size \geq k$ **then**
    arbitrarily select $size - k$ elements from $my\_maps[dist]$ and remove them
    **return** $my\_maps$ (exiting from loop)
  **end if**
**end for**
```

O_a	
Dist.	Nodes
0	$\{a\}$
1	$\{x\}$
2	$\{y\}$
3	$\{w, z\}$

I_b	
Dist.	Nodes
0	$\{b\}$
1	$\{t\}$
2	$\{w\}$
3	$\{y, z\}$

I_c	
Dist.	Nodes
0	$\{c\}$
1	$\{l\}$
2	$\{m\}$
3	$\{n, o\}$

Figure 4. Example for algorithm 2

The algorithm recursively finds the set of nodes exactly at distance $dist$ from a node n by iterating through all nodes n' that are connected to n with outgoing edges. For each n', all previously undiscovered nodes at distance $dist - d$ are added, with d being the label of the edge connecting n to n'.

Algorithm 1 must also be executed to calculate distances for incoming paths. In order to do this, $conn$ needs to contain data about incoming edges instead of outgoing ones.

Evaluating Distance Once the neighbourhood maps have been built, estimating distance means finding the minimal sum of distances for nodes that appear in both maps, i.e. returning the shortest path found.

If we do not find a path, we have to guess. Since we are working with small-world graphs, we can guess that the real distance is not much larger than the greatest distance we could have possibly found. For this reason, we just use the sum of the maximum distances indexed by the two neighbourhood maps.

Given node x's outgoing map O_x and node y's incoming map I_y, the shortest path from x to y is thus calculated according to algorithm 2.

Algorithm 2 Distance evaluation

```
$path\_lengths \leftarrow \{i + j \mid \exists n.n \in (O_x[i] \cap I_y[j])\}$
**if** $path\_lengths \neq \emptyset$ **then**
  **return** $\min(path\_lengths)$
**else**
  **return** $(length\,(m_o) - 1) + (length\,(m_i) - 1)$
**end if**
```

As an example, consider the neighbourhood maps from figure 4. Evaluating distance from a to b, the nodes appearing in both maps are y, w and z. Distances related to paths passing through them are respectively 5, 5 and 4. The minimum distance is thus 4, which will be the outcome of the algorithm. In order to evaluate distance from a to c, the sum of the maximal indexes in the two maps will be used, since no intersection is found. The result is thus $3 + 3 = 6$.

The distance estimation algorithm has been tested on the OpenPGP web of trust and a random network having the same number of nodes and edges, as discussed in sections

5.1 and 5.2.

5.1. Non-empty Intersections

Having a non-empty intersection between the two neighbourhood maps means a path has been found, and thus we have an upper bound on the distance between the nodes. The way in which maps are constructed does not, anyway, guarantee the shortest path has been found[4]. In this section, we will analyse how the probability of getting a non-empty intersection between neighbourhood maps varies in relation to the map sizes; section 5.2 will give results about the precision level reached by our approximation.

Expectation in Random Maps In order for neighbourhood maps to convey useful information, they need to have a non-empty intersections with high probability. In order to do this, we are going to study how this probability behaves relating to the network size, under the assumption that the neighbourhood maps are a random selection of nodes in the network[5].

If a value for the distance exists (i.e., the two nodes are part of the same connected component), the probability of having one or more intersections in the neighbourhood maps, when the network is composed by n nodes and maps have size k is at least

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

Studying the limit for $n \to \infty$ of the right member of equation (1):

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

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

Equation (2) tells us that the asymptotic value of equation (1) depends on $\lim_{n\to\infty}\frac{k^2}{n}$: if it goes to infinity (that is, k grows faster than $\sqrt{n}$), then the success probability expressed in equation (1) goes to 1. Otherwise, if it goes to 0, success probability goes to 0.

[4]In graphs where all labels are less or equal to x, the optimal distance can be shown to be overestimated at most by x. Since in the OpenPGP web of trust all edges are labelled with 1, the overestimation is thus at most 1.

[5]This assumption is not true, even for random networks: in fact, nodes with high degree have higher probability of appearing in a map than those which do not. Anyway, the graph in figure 5 shows this provides a useful approximation for random graphs.

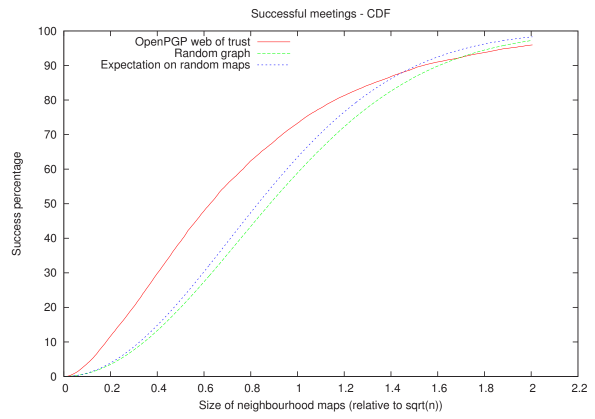

Figure 5. Non-empty intersection probability

With $k = t\sqrt{n}$, we get $1-e^{-t^2}$. This means we can get an arbitrarily low probability of an empty intersection, keeping map size proportional to $\sqrt{n}$.

Monte Carlo Results The graph in figure 5 shows the evolution of non-empty intersection probability versus the neighbourhood maps sizes. We contrasted the behaviour of the algorithm for the OpenPGP web of trust, our theoretical predictions for random maps, and the behaviour in a random network having the same number of nodes and edges as the web of trust graph. The neighbourhood map sizes vary from 0 to $2\sqrt{n}$, and the experimental data have been produced by evaluating intersection on 10000 random node pairs. The map sizes on the x axis have been divided by $\sqrt{n}$, for easier referencing against equation (2).

While the curve for the random network and the theoretical prediction for random maps have a similar shape, the curve for the PGP web of trust significantly deviates from the other two; we infer this could be motivated by two reasons:

- for low-size maps, having hubs helps us, because it is more likely that a "famous" node is at a short distance between source and target, and thus we are more likely to find a path;
- for larger maps, we are paying for the high clustering of our network: information in neighbour maps is more redundant than in other cases, since maps of close nodes will be very similar and thus will convey less information.

5.2. Approximation

In order to discuss the accuracy of our method to evaluate distances, we compared the results of our approximation with the exact distance.

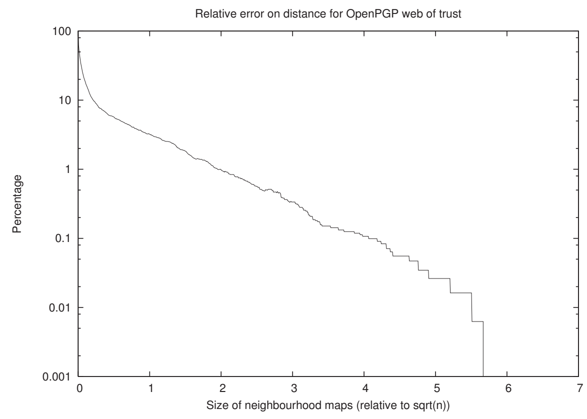

Figure 6. Relative error plot

The graph in figure 6 shows (with logarithmic scale on the y axis) how the relative average error is, compared to the map size. The test graph is, as usual, the OpenPGP web of trust. The behaviour of error seems to suggest that the approximation given by our method increases exponentially with the map size.

The irregular behaviour on the right hand of the graph is motivated by the fact that the sample having an error when the map size is large is very low; in our 10000 sample pairs we could not find errors for a map size greater than $938 \cong 5.67\sqrt{n}$.

Our experimental results appear to confirm our theoretical prediction that maps having sizes equal to $k\sqrt{n}$ can be used in order to calculate our desired value. The k value can be adjusted depending on the desired precision level.

6. Approximating PageRank

As shown in the previous section, the shortest path length can be efficiently approximated using neighbourhood maps. The most significant weakness of this approach, though, is that only one path between a pair of node on the LRV network is taken into consideration; thus, the resulting ranking conveys little information.

Algorithms based on link analysis can lead to more significant results. PageRank [21] is probably the most widely known of them, being used to rank results in the Google web search engine.

6.1. About PageRank

In PageRank, a random walk on the LRV graph is performed, with a probability α of stopping at each step. The result of PageRank is the probability that each node has of being the end point of such a walk[6]. Nodes having a higher probability in such distribution get a higher ranking. In the final asymptotic result, all paths of all lengths from the starting nodes are taken into account, with a decreasing weight for longer paths.

When PageRank is used for web page ranking, and in EigenTrust [17], the starting point is a random distribution over many or all the nodes in the graph; in our approach, the starting point is the evaluating node. This results in a *subjective* (i.e., depending on the node evaluating it) reputation evaluation. As [13] argues, this provides security against collusive attacks, without requiring any kind of centralisation.

The α parameter is relevant to the speed of convergence of the algorithm, since all paths of length n have a total weight of $\alpha(1-\alpha)^n$. For the standard value $\alpha = 0.15$, this means that 83.3% of the total weight attributed by PageRank is constituted by paths up to 10 in length, 96.7% by paths of length up to 20, and 99.4% up to 30.

While this algorithm can be influenced by Sybil attacks, it can be easily shown that rankings can not be boosted by a factor exceeding $\frac{1}{\alpha}$[7]. Since PageRank values typically fit on a power law [22], we can assume such an attack would have only a limited impact on the resulting ranking.

6.2. Constructing Neighbourhood Maps

Let $p_{a,b}$ be the probability that a random walk starting from node a ends at node b[8]. If $N(n)$ is the set of neighbours (outgoing links) of n, and $\#N(n)$ its cardinality, the following equations hold:

$$p_{a,b} = \begin{cases} \alpha + \dfrac{(1-\alpha)}{\#N(a)} \displaystyle\sum_{n \in N(a)} p_{n,b} & \text{if } a = b \\ \dfrac{(1-\alpha)}{\#N(a)} \displaystyle\sum_{n \in N(a)} p_{n,b} & \text{otherwise} \end{cases} . \quad (3)$$

Equation (3) is used in order to create algorithm 3, which calculates all values of p (each node n calculates all values $p_{n,m}$ for all other nodes m).

We use this algorithm to create a neighbourhood map that contains the k highest ranked nodes by each node ("outgoing" map). In order to put a bound on computational requirements, at the end of each iteration we can reset to 0 all values $p_{a,b}$ where b is not in the k nodes that received the highest ranking from a, and iterate only on nodes that appear in neighbours' maps (i.e., nodes m such that it exists a

[6]This is equivalent, although more fit to our explanation, to the usual definition of a probability α of reverting back to a starting point, and evaluating the stationary distribution of such an infinite random walk.

[7]The attack consists in creating loops that never escape from the set of malicious nodes.

[8]In other words, $p_{a,b}$ is the ranking of b which is calculated by a.

Algorithm 3 Neighbourhood maps for PageRank

```
on each node $a$ do:
for all node $b$ do
    $p_{a,b} \leftarrow 0$
end for
while values have not converged do
    for all node $b$ do
        if $a = b$ then
            $p_{a,b} \leftarrow \alpha + \frac{1-\alpha}{\#N(a)} \sum_{n \in N(a)} p_{n,b}$
        else
            $p_{a,b} \leftarrow \frac{1-\alpha}{\#N(a)} \sum_{n \in N(a)} p_{n,b}$
        end if
    end for
end while
```

neighbour n having $p_{n,m} > 0$). We will denote the resulting maps as O_a, where $O_a[b]$ will denote our approximation for $p_{a,b}$. As a shortcut, we will write $b \in O_a$ to mean that O_a contains a value for b.

If $N^{-1}(n)$ is the set of incoming links to b, the following equation is also satisfied:

$$p_{a,b} = \begin{cases} \alpha + (1-\alpha) \displaystyle\sum_{n \in N^{-1}(b)} \frac{p_{a,n}}{\#N(n)} & \text{if } a = b \\ (1-\alpha) \displaystyle\sum_{n \in N^{-1}(b)} \frac{p_{a,n}}{\#N(n)} & \text{otherwise} \end{cases} . \tag{4}$$

This allows us to write an algorithm that uses the same approach as before in order to build a neighbourhood map ("incoming" map) of the k nodes that give the highest ranking to b. Using the same notation as before, we will call the resulting map I_b for each node b.

6.3. Using Neighbourhood Maps to Approximate PageRank

Let us say that peer a wants to evaluate peer b's PageRank value (i.e., $p_{a,b}$).

Let $X_{a,b}$ be the set of nodes in both a's outgoing map and b's incoming one, that is $x \in X_{a,b} \iff x \in O_a \wedge x \in I_b$. For each $x \in X_{a,b}$, we have then calculated a probability $O_a[x]$ that a random walk starting from a will stop at x, and a $I_b[x]$ probability that a walk starting from n will arrive at b. Given the fact that the probability that a walk stops is α, the probability of having a random walk getting from a to b passing through x is then

$$\frac{1-\alpha}{\alpha} O_a[x] \cdot I_b[x].$$

Since, as seen before, shorter paths are the most important ones in constructing rankings, and most of the shorter paths are found using neighbourhood maps, this means that using this approach we can hope to capture most of them. As an estimation of ranking, we then use the following formula, ignoring the $\frac{1-\alpha}{\alpha}$ constant which is not going to affect the ranking:

$$\sum_{x \in X_{a,b}} O_a[x] \cdot I_b[x]$$

An important point in evaluating the accuracy of this approach is that if a path passes through more than one elements of X, then it is likely that it is accounted more than once. This fact is going to introduce some noise in our calculation.

6.4. Avoiding "spam" attacks

An attacker could attack the "incoming" neighbourhood map for a node by creating a high number of fake nodes, and making them give a high ranking to the attacked node. In this way, that map would become full of references to insignificant nodes, and thus useless.

In order to avoid this kind of attack, nodes constructing the maps for incoming links can decide to give precedence to nodes that appear in their own "outgoing" map. In order to reach this goal, we added a priority value $P(a, b)$ defined for each pair of nodes (a, b) such that $b \in I_a$, which has this value:

$$P(a,b) = \begin{cases} O_b[a] \cdot I_b[a] & \text{if } a \in O_b \\ \min_{x \in O_b} (O_b[x]) \cdot I_a[b] & \text{otherwise} \end{cases} .$$

Each node constructs its I_x map putting the first k nodes according to this priority value.

6.5. Experimental Results

We evaluated the similarity of ranking between our method and the PageRank evaluation, using Kendall's tau distance as metrics. It consists in evaluating the probability that, given two rankings and a pair of random nodes, the two rankings agree on which one is ranked higher. The graph in figure 7 on the following page shows how the neighbourhood map size is related to this probability. The test graph is always the OpenPGP web-of-trust described in Section 3.

We can see that the "secure" method described in section 6.4 provides better results than the original one. This is probably due to the fact that, using that criterion, more relevant nodes are taken into account.

7. Further Work

Other kinds of metrics (such as, for instance, maximal flow) could be approximated using neighbourhood maps.

The high clustering presented by this kind of real-world networks reduces the precision we can reach. A plausible

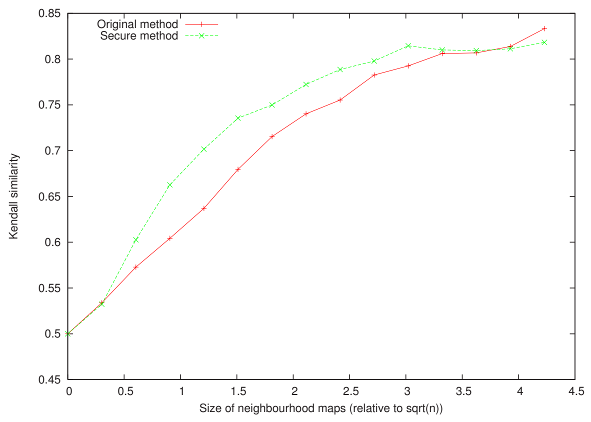

Figure 7. Kendall's tau distance for PageRank

way of increasing the effectiveness of this method is trying to differentiate neighbourhood maps of close nodes, giving a higher priority (as seen in Section 6.4) to nodes that appear to be outside the cluster a node belongs to.

In order to quantify the incentive to cooperation given to nodes, we plan to do an "evolutionary-game" analysis similar to the one used in [13], contrasting the payoffs received with selfish and cooperative behaviour.

8. Conclusion

In this paper we have shown a generic method for calculating metrics about relationships between nodes in a graph. We have shown how to use this method to give an estimation of graph distance and of PageRank. Our results imply that using on each node data structures needing a storage space proportional to $\sqrt{n}$, where n is the size of the network, can yield useful results.

We believe that our idea can be fruitfully exploited in large-scale peer-to-peer networks in order to create a scalable and efficient way of giving incentives to cooperation and thus helping solve the "free-riding" problem.

Acknowledgements The author wishes to thank Prof. Giovanni Chiola and Prof. Marina Ribaudo for their kind suggestions and support, and the anonymous reviewers for their insightful remarks.

References

[1] Edonkey2000. http://www.endonkey2000.com.

[2] Gnutella - A Protocol for a Revolution. http://rfc-gnutella.sourceforge.net.

[3] A. Marcozzi, D. Hales, G. Jesi, S. Arteconi, and O. Babaoglu. Tag-Based Cooperation in Peer-to-Peer Networks with Newscast. Technical Report UBLCS-2005-15, University of Bologna, Dept. of Computer Science, May 2005.

[4] E. Adar and B. A. Huberman. Free riding on gnutella. *First Monday*, 5(10), 2 Oct. 2000.

[5] Alice Cheng and Eric Friedman. Sybilproof Reputation Mechanisms. In *Third Workshop on Economics of Peer-to-Peer Systems, Philadelphia, PA.*, 22 Aug. 2005.

[6] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[7] A. L. Barabási. *Linked.* (Perseus, Cambridge, Massachusetts), 2002.

[8] Borodin, Roberts, Rosenthal, and Tsaparas. Link analysis ranking: Algorithms, theory, and experiments. *ACMTIT: ACM Transactions on Internet Technology*, 5, 2005.

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

[10] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP message format. Internet Request for Comment RFC 2440, Internet Engineering Task Force, Nov. 1998.

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

[12] J. R. Douceur. The sybil attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Mar. 2002.

[13] Feldman, Lai, Stoica, and Chuang. Robust incentive techniques for peer-to-peer networks. In *CECOMM: ACM Conference on Electronic Commerce*, 2004.

[14] E. J. Friedman and P. Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics & Management Strategy*, Aug. 17 2001.

[15] Henk P. Penning. PGP pathfinder and key statistics. http://www.cs.uu.nl/people/henkp/henkp/pgp/pathfinder/.

[16] Jörgen Cederlöf. Wotsap. http://www.lysator.liu.se/ jc/wotsap/index.html.

[17] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *WWW*, pages 640–651, 2003.

[18] J. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM: Journal of the ACM*, 46, 1999.

[19] Lik Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Massachusetts Institute of Technology, 2003.

[20] Njål T. Borch. Improving semantic routing efficiency. In *Second International Workshop on Hot Topics in Peer-to-Peer Systems*, 21 July 2005.

[21] Page and Lawrence. PageRank: Bringing order to the web. Stanford Digital Libraries Working Paper 1997-0072, Stanford University, 1997.

[22] G. Pandurangan, P. Raghavan, and E. Upfal. Using pagerank to characterize web structure. In *COCOON*, pages 330–339, 2002.

[23] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.