Mapping Small Worlds

Matteo Dell'Amico
Dipartimento di Informatica e Scienze dell'Informazione
Università di Genova
Via Dodecaneso 35, 16146 Genova, Italy
dellamico@disi.unige.it

Abstract

Social networks are usually navigable small worlds: individuals are able to find short chains of acquaintances connecting pairs of unrelated nodes. This property can be explained by the fact that nodes are characterized by a series of properties, such as geographical position, work or educational background; the navigation proceeds towards the node that is "most similar" to the destination. Since nodes are likely to be linked with similar individuals, this strategy permits to quickly reach the destination.

We approach the problem of creating the information that makes a network navigable. Starting from a given network, and without any other information, we show how nodes can reconstruct, with a scalable and decentralized algorithm, a "network map": a d-dimensional layout that places nodes in a way that reflects the network structure, so that navigability is achieved. Euclidean distance on the layout is used as a measure for node similarity, and efficient routing can be simply achieved by iteratively jumping towards the neighbor that is closest to the destination.

The network map provides a means for implementing routing on social networks that can be used in "darknets", that is, anonymous networks where nodes establish connections only if they are mutually trusted. Moreover, the distance between nodes on the network map can be used as a measure of node affinity, and may help in various types of network analysis, for instance to help evaluate reputation in webs of trust, or in order to perform "personalized" ranking.

1. Introduction

Many peer-to-peer applications are deeply related with social networks. Examples can be the "data-sharing" [13] 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.

It appears obvious that tools that can analyze the properties of those social networks can be very helpful in modeling and improving those applications. Our goal, here, is to express a metric that models "affinity" between nodes, where the likelihood that two nodes are linked increases with their affinity. This is done here by building a layout mapping nodes to positions in a d-dimensional space. We will use Euclidean distance as affinity measure, and we will call such layout a "network map".

The affinity between nodes can be used as a heuristics for "routing" between nodes in social network: paths progress from the current node towards the neighbor that is closest to the destination in the network map.

The contribution of this paper can be summarized in three points:

1. We calculate network maps using Koren's graph drawing algorithm [16] 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.

The problem of routing in social networks arises in "darknets" such as the one implemented in Freenet [8]: 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 [20].

We believe that the layout we obtain reflects information about the intimate structure of the network; as such, this information could be used in order to analyze the network itself.

For instance, in networks that associate favorite items or products to users, the map could be used to recommend interesting items. Web search results could be personalized by assigning a higher ranking to nodes that are deemed as "close" to the interests of the requester. The layout could also be used in order to optimize reputation evaluation, in the case of "subjective" ranking of the nodes, when the ranking depends on the paths that connect the evaluating and the evaluated nodes [12]: information on the network map could be used to discover paths connecting the two nodes. Network maps could also be used in "tag clouds" [3], 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).

2. Related work

2.1. Small worlds and navigability

The "small world" phenomenon has been discovered by John Milgram in 1967 [18]: people could send a letter to an unknown other person across the USA using only direct acquaintances, with an average number of six links. A network is considered a small world if it presents high clustering and short chains of links connecting unrelated nodes; these properties are ubiquitous among social networks.

Clustering can be defined [19] as the ratio between the number of existing edges between neighbors of a given node and their maximum number. In other words, it can be seen as the probability that "two friends of someone are friends between themselves". Networks with high clustering tend to have tightly connected groups (*clusters*) where each node is connected to many other nodes in the same group. While intuition may suggest that most paths would not escape from the starting clusters, in small world networks the number of nodes that can be reached grows exponentially with the number of steps, yielding logarithmic diameter with respect to network size.

Indeed, Milgram's experiment also proved that the social network was *navigable*: people were able to actually find short paths in the graph, using only local information about their contacts and some information about the target, such as their address or their work.

Navigability has been explained by the fact that nodes are more likely to connect to "similar" nodes, that is, people that live in the same area or do the same work [24, 11]. Since people were forwarding the letter to nodes that were more and more affine to the destination, the probability of those nodes to know the target was growing higher.

Models for small-worlds networks often implicitly embrace the concept of affinity between nodes: they are placed in a d-dimensional layout (a ring, a lattice, or any other topology), and close nodes in that layout are linked with a higher probability [25, 9]. In these cases, distance on the network can be seen as a measure of the affinity between nodes.

Kleinberg [15] introduced a family of models for navigable small world networks: n nodes[1] are placed on a d-dimensional grid, and each node has q directed long-range connections to other nodes, where the probability of a node being linked is inversely proportional to its distance elevated to d. It is shown that routing based on the layout is efficient (poly-logarithmic with respect to n).

Kleinberg gave a simple geographical interpretation for the position of nodes in the grid; while geographic position has been demonstrated to be the most useful piece of information for routing on social networks [17], it can be argued that other information can be used to obtain 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 [20] 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 [5].

In this paper, 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.

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. All edges would have unitary length.

[1] In the following, we will use n as shorthand notation for the size of the network under discussion.

Unfortunately, such a layout would need $n-1$ dimensions; 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 putting 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" [14]: 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 $O\left(n^2\right)$, and require 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 such an algorithm, described in the following section, to our requirements.

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 through explanation, see [16]. While the algorithm works also on weighted graphs, here we will refer for simplicity 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 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 result is Algorithm 1. The iterative step of the algorithm (lines 7-16) 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 avoid convergence in 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 $\mu_i = 2\left(1 - \lambda_i\right)$. Vectors satisfying such equation are very well known in spectral graph theory. In eigenpairs $\langle \mu_i, u_i \rangle$ with i small and not 1, the value of $u_i(j)$ is close to $u_i(k)$ if the nodes j and k belong to the same cluster [22]. Nodes belonging to the same community will thus be placed close in our layout.

Algorithm 1 Koren's layout algorithm.

1: $u_1 \leftarrow 1^n$ {First eigenvector}
2: **for** $i = 2$ to $d+1$ **do** {calculating the u_i eigenvector}
3: **for all** nodes $v \in V$ **do**
4: $u_i(v) \leftarrow$random {random initialization}
5: **end for**
6: **while** algorithm has not converged **do**
7: **for** $j = 1$ to $i-1$ **do** {orthogonalize against u_j}
8: $c \leftarrow \frac{\sum_{v \in V} k_v u_i(v) u_j(v)}{\sum_{v \in V} k_v (u_j(v))^2}$
9: **for all** nodes $v \in V$ **do**
10: $u_i(v) \leftarrow u_i(v) - c \cdot u_j(v)$
11: **end for**
12: **end for**
13: **for all** nodes v **do** {Multiply by $\frac{1}{2}\left(I + D^{-1}A\right)$}
14: $u'_i(v) \leftarrow \frac{1}{2}\left(u_i(v) + \frac{\sum_{(v,v') \in E} u_i(v')}{k_v}\right)$
15: **end for**
16: $u_i \leftarrow \frac{u'_i}{\|u'_i\|}$ {normalization}
17: **end while**
18: **end for**

3. The OpenPGP web of trust

The OpenPGP web of trust will be used in order to validate our experimental results. It is a social network where nodes know and trust each other, and has been already used by Sandberg [20] for testing his routing on social network algorithms.

OpenPGP [2] 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.

In order 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 26350 nodes with an average degree of 7.98; the average clustering is 0.3047.

The computation of the first eigenvalues using algorithm 1 leads to eigenvalues μ_i that are very near 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). With these values, thus, the convergence speed of the algorithm is slow.

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 subcommunities. 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 [4]. The presence of clusters in a network yields eigenvalues close to zero in the Laplacian matrix [6].

4. Decentralized implementation

We need to implement Algorithm 1 so that it can be executed on a decentralized system. We will do this by requiring that each node needs to communicate only with its neighbors on the web of trust.

The core step of matrix multiplication (lines 13-15) is already decentralized: each node needs to know only the position of its neighbors in order to compute the multiplication step. The two parts that need to be modified are normalization (line 16) and D-orthogonalization (lines 7-12).

4.1. D-Orthogonalization

The orthogonalization step is carried out in order 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 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 experiment.

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 can be performing a long enough random walk on the social network. It appears that social networks are fast mixing [26], so a walk having length of $\Theta(\log n)$ should be enough.

A consequence of the fact that the eigenvalues $\lambda_2 \simeq \lambda_3 \simeq \ldots$ are similar in value (as seen in section 3) 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 pretty good 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 4.2.

4.2. Normalization

The normalization step is used for two reasons: 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). 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 were very close to 1, the values were well within the boundaries for floating point numbers, so we only compute normalization at the end of the algorithm.

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

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

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 1, 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 $S = \sum_{v_i \in V} k_i u(v_i)$ 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 $\frac{u(v_a) - u(v_b)}{k_a}$, and the displacement caused to v_b will be $\frac{u(v_b) - u(v_a)}{k_b}$. The link between v_a and v_b does not change S, and the same can be said for any other linked pair.

Algorithm 2 performs a distributed normalization of all u_j vectors.

Algorithm 2 Distributed normalization algorithm.

1: on each node v_x do
2: $N \leftarrow 200$ {number of iterations}
3: **for** $j = 2$ to d **do** {initialization}
4: $\quad m_{j,x} \leftarrow u_j(v_x)^2$
5: **end for**
6: **for** $i = 1$ to N **do**
7: $\quad$ **for** $j = 2$ to d **do**
8: $\quad\quad m_{j,x} \leftarrow \frac{1}{2}\left(m_{j,x} + \frac{\sum_{(v_x, v_y) \in E} u_j(m_{j,y})}{k_x}\right)$
9: $\quad$ **end for**
10: **end for**
11: **for** $j = 2$ to d **do** {initialization}
12: $\quad u_j(v_x) \leftarrow \frac{u_j(v_x)}{\sqrt{m_{j,x}}}$
13: **end for**

4.3. The decentralized algorithm

All modifications discussed above yield to the distributed version shown in Algorithm 3. The relationships between the number of iterations N, the quality of the resulting layout and the characteristics of the network are unfortunately not clear yet. For this reason, we recommend using simulation to determine a suitable value of N for a given network.

The communication overhead due to the algorithm consists of passing, at each iteration, the coordinates that a node has 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 on the node.

Algorithm 3 Distributed layout algorithm.

on each node v_x do
$N \leftarrow 200$ {number of iterations}
for $j = 2$ to d **do** {D-orthogonalize against $u_1 = 1^n$}
  if a random pair v_y is found **then**
    negotiate a random number r, giving $-r$ to v_y
    $u_j(v_x) \leftarrow \frac{r}{k_x}$
  else
    $u_i(n) \leftarrow 0$
  end if
end for
for $i = 1$ to N **do** {multiplication}
  for $j = 2$ to d **do**
    $u_j(v_x) \leftarrow \frac{1}{2}\left(u_j(v_x) + \frac{\sum_{(v_x,v_y)\in E} u_j(v_y)}{k_n}\right)$
  end for
end for
Do normalization as shown in Algorithm 2.

5. Experimental validation

Once the layout is obtained, the routing algorithm is extremely simple, analogous to the "greedy" routing used in DHTs such as Chord [23]: the route progresses from node 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.

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 $6000n$ 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 [1], routing strategies that took in consideration degree as well as distance on the layout were taken into consideration. These strategies forward the route towards the neighbor with a higher value of $\frac{k_i^\alpha}{d_i}$, where k_i is the degree, d_i is the distance from the destination, 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.

5.1. Comparison between layout algorithms

Figure 1 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.

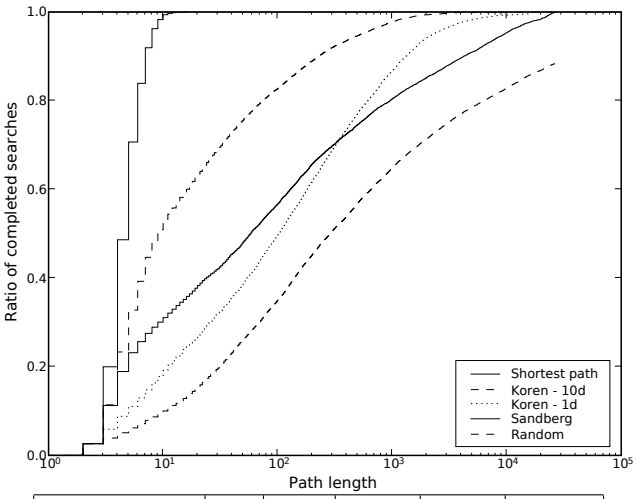
In order to give a reference against a best and and a worst case, the length of the shortest paths is shown; a "random" routing algorithm has been also implemented. It proceeds by following edges at random, until the destination node is one of the neighbors of the current node. The execution of the random 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 1452.55 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.

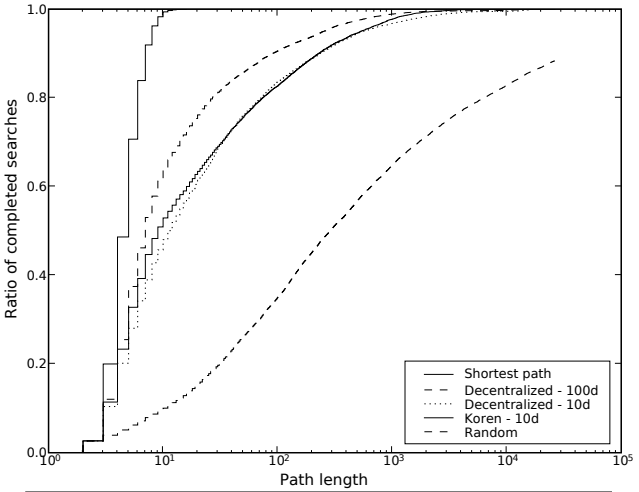
5.2. Decentralized algorithm

Figure 2 compares the experimental results of using Koren's algorithm and its decentralized approximation of Algorithm 3. 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



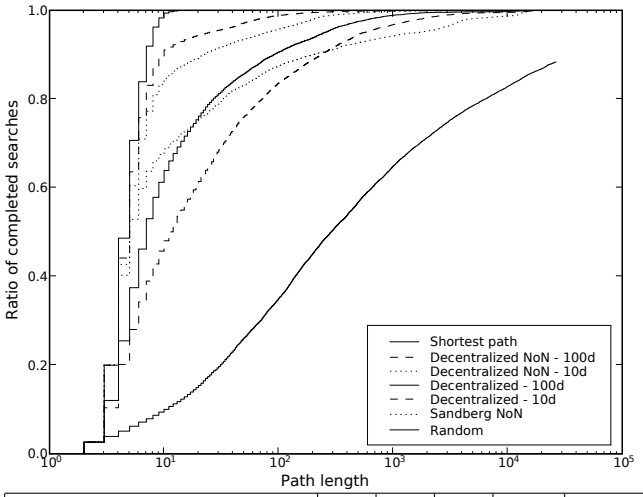
Percentile	25	50	75	90	99
Shortest path	3	4	5	6	9
Koren - 10d	4	9	53	241	1912
Koren - 1d	16	102	433	1320	7475
Sandberg	5	57	523	4202	22297
Random	45	286	3063	-	-

Figure 1. 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	1135
Decentralized - 10d	4	10	45	233	3177
Koren - 10d	4	9	53	241	1912

Figure 2. Experimental results for the decentralized algorithm shown in Algorithm 3.



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	1135
Decentralized - 10d	4	10	45	233	3177
Sandberg - NoN	3	4	18	199	12492

Figure 3. Experimental results for NoN routing.

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.

5.3. 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): it is reasonable to assume that a user has some information about the acquaintances of a friend.

In our case, we could assume that each node knows the position on the layout of the neighbors of each neighbor (this information can be exchanged at the end of Algorithm 3). 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 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 [7].

Experimental results in Figure 3 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.

6. Conclusions and future work

We presented a distributed decentralized algorithm that assigns a layout to nodes of a complex network in a decentralized way, based on algorithms for graph drawing using spectral techniques. It can be used to obtain reasonably efficient routing on the network, making the idea of a secure and anonymous "friend-to-friend" network viable.

As mentioned in the Introduction, this approach can be applied to various kinds of complex networks and to reputation evaluation, especially when a decentralized approach is needed.

The NoN approach seen in Section 5.3 can be extended using a more refined strategy, like obtaining some set of local information about a fixed number of close nodes, as seen in [10]. This can allow the most problematic nodes - that is, low-degree nodes which are not connected to high degree ones, to store an amount of local information that is hopefully enough to allow them to reach and be reached efficiently by other nodes.

To obtain better routing with a controlled increase in the number of sent messages, a technique similar to probabilistic broadcasting [21] could be introduced, introducing bifurcations in search paths when the layout routing appears to fail.

To use this approach in a darknet, resilience to attack on the layout algorithm by malicious nodes needs to be investigated. A possible means could be based on limiting the influence of any single node on the placement of its neighbors.

Acknowledgments The author thanks the many people who gave suggestions and help. Among them, Maurizio Filippone was especially helpful in introducing spectral graph theory.

References

[1] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physical Review E*, 64(4):046135+, 2001.

[2] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP message format. Internet RFC 2440, IETF, 1998.

[3] C. Cattuto, V. Loreto, and L. Pietronero. Semiotic dynamics and collaborative tagging. *PNAS*, 104(5):1461–1464, 30 Jan. 2007.

[4] J. Cederlöf. Dissecting the Leaf of Trust. http://www.lysator.liu.se/ jc/wotsap/leafoftrust.html.

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

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

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

[8] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[9] M. Dell'Amico. Highly Clustered Networks with Preferential Attachment to Close Nodes. In *Proceedings of ECCS 2006*, Nov. 2006.

[10] M. Dell'Amico. Neighbourhood Maps: Decentralised Ranking in Small-World P2P Networks. In *3rd International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P)*, Apr. 2006.

[11] P. S. Dodds, R. Muhamad, and D. J. Watts. An experimental study of search in global social networks. *Science*, 301:827–829, Aug. 2003.

[12] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *ACM conference on electronic commerce*, 2004.

[13] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file-sharing communities. In *INFOCOM*, Hong Kong, 2004.

[14] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 12 Apr. 1989.

[15] J. M. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *STOC*, pages 163–170, 2000.

[16] Y. Koren. On spectral graph drawing. In *COCOON: Annual International Conference on Computing and Combinatorics*, 2003.

[17] D. Liben-Nowell, J. Novak, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, 102(33):11623–1162, 2005.

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

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

[20] O. Sandberg. *Searching in a Small World*. PhD thesis, Chalmers University of Technology and Göteborg University, 2005.

[21] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury. Percolation search in power law networks: making unstructured peer-to-peer networks scalable. In *P2P 2004*, pages 2–9.

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

[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.

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

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

[26] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *ACM SIGCOMM 2006*, pages 267–278. ACM, 2006.