Distributed Routing in Small-World Networks

Oskar Sandberg [*]

December 22, 2005

Abstract

So called small-world networks – clustered networks with small diameters – are thought to be prevalent in nature, especially appearing in people's social interactions. Many models exist for this phenomenon, with some of the most recent explaining how it is possible to find short routes between nodes in such networks. Searching for such routes, however, always depends on nodes knowing what their and their neighbors positions are relative to the destination. In real applications where one may wish to search a small-world network, such as peer-to-peer computer networks, this cannot always be assumed to be true.

We propose and explore a method of routing that does not depend on such knowledge, and which can be implemented in a completely distributed way without any global elements. The Markov Chain Monte-Carlo based algorithm takes only a graph as input, and requires no further information about the nodes themselves. The proposed method is tested against simulated and real world data.

1 Introduction

The modern view of the so called "small-world phenomenon" can be dated back to the famous experiments by Stanley Milgram in the 1960s [15]. Milgram experimented with people's ability to find routes to a destination within the social network of the American population. He concluded that people were remarkably efficient at finding such routes, even towards a destination on the other side of the country. More recent studies using the Internet have come to the same conclusion, see [6].

Models to explain why graphs develop a small diameter ([19], [4], [17]), have been around for some times. Generally, these models specify the mixing of a structured base graph, such a as grid, and random "shortcuts" edges between nodes. However, it was not until Jon Kleinberg's work in 2000 [11] that a mathematical model was developed for how efficient routing can take place in such networks. Kleinberg showed that the possibility of efficient routing depends on a balance between the proportion of shortcut edges of different lengths with respect to coordinates in the base grid. Under a specific distribution, where the frequency of edges of different lengths decreases inverse proportionally to the length, simple greedy routing (always walking towards the destination) can find routes in $O(\log^2(n))$ steps on average, where n is the size of the graph.

1.1 Motivation Kleinberg's result is sharp in the sense that graphs where edges are chosen from a different distribution are shown not to allow for efficient searching. However, the small-world experiments seem to show that greedy-like routing is efficient in the world's social network. This indicates that some element of Kleinberg's model is present in the real world. In [12] and [18] this is motivated by reason of people's group memberships[1]. Several dynamic processes by which networks can evolve to achieve a similar edge distribution have also been proposed recently, for example, in [5], as well as in forthcoming work by this author [16].

However, in Kleinberg's search algorithm, the individual nodes are assumed to be aware of their own coordinates as well as those of their neighbors and the destination node. In the case of real world data, it may be difficult to identify what these coordinates are. In fact the participant nodes may be unaware of anything but their immediate neighborhood and thus oblivious of the global structure of the graph, and, importantly for this work, of geographic (or other) coordinates. For example, in peer-to-peer overlay networks on the Internet, one may wish to automatically find routes without relying on information about the local user,

[*]The Department of Mathematical Sciences, Chalmers Technical University and Gothenburg University. ossa@math.chalmers.se. I thank my advisors Olle Häggström and Devdatt Dubhashi for their assistance in reading drafts of this paper, as well as the anonymous referees for their constructive comments.

[1]Roughly: When a group is twice as large, people in it are half as likely to know each other.

let alone his neighbors or the routes target. In such a situation, how can we search for short paths from one node to another?

1.2 Contribution With this in mind, this paper attempts to return to Milgram's original problem of finding paths between people in social networks. Starting from an unmarked shortcut graph and no other information on the coordinates, we attempt to fit it against Kleinberg's model so as to make efficient searches possible. Taking as hypothesis that the graph was generated by applying Kleinberg's distribution model to a base graph with co-ordinate information, we attempt to recover the *embedding*. We approach this as a statistical estimation problem, with the configuration of positions in the grid assigned to each node as a (multi–dimensional) unknown parameter. With a good estimate for this embedding, it is possible to make greedy routing work without knowing the original positions of the nodes when the graph was generated. We employ a Markov Chain Monte-Carlo (MCMC) technique for fitting the positions.

We summarize our contributions as follows:

1. We give an MCMC algorithm to generate an embedding of a given graph into a one or two dimensional (toric) grid which is tuned to the distributions of Kleinberg's model.

2. This method is tested using artificially generated and controlled data: graphs generated according to the ideal model in one and two dimensions. The method is demonstrated to work quite well.

3. It is then applied to real social network data, taken from the "web of trust" of the users of an email cryptography program.

4. Finally, it is observed that the method used can be fully distributed, working only with local knowledge at each vertex. This suggests an application to routing in decentralized networks of peers that only connect directly to their own trusted friends in the network. Such networks, known as Friend-to-Friend networks of Darknets, have so far been limited to communication only in small cliques, and may become much more useful if global routing is made possible.

5. Our algorithm can thus be viewed also as a general purpose routing algorithm on arbitrary networks. It is tailored to "small world" networks, but appears to also work quite well for a more general class of graphs.

1.3 Previous Work Different methods of searching social networks and similar graphs have been discussed in previous work. In [3] a method is proposed for searching so called "power-law networks", either by a random walk or by targeting searches at nodes with high degree. Because such graphs have a highly skewed degree distribution, where a small set of nodes are connected to almost everyone, the methods are found to work well. The first author of that paper and a co-author recently investigated the problem of searching social networks in [2]. There they found that power-law methods did not work well, and instead attempted to use Kleinberg's model by trying to identify people's positions in some base graph based on their characteristics (where they live, work, etc). This was found to work well on a network with a canonical, highly structured base graph (employees of Hewlett Packard) but less well on the social network of students at Stanford University. Similarly Liben-Nowell et. al. [13] performed greedy searches using the town names as locations in the network of writers on the website "LiveJournal". They claim positive results, but consider searches successful when the same town as the desired target is reached: a considerably easier task than routing all the way.

In [20] the authors attempt to find methods to search a network of references between scientific authors. They mention Kleinberg's model, but state:

> "The topology of referral networks is similar to a two-dimensional lattice, but in our settings there is no global information about the position of the target, and hence it is not possible to determine whether a move is toward or away from the target".

It is the necessity of having such information that we attempt to overcome here.

2 Kleinberg's Model

Kleinberg's small-world model, like that of Watts and Strogatz [19] which preceded it, starts with a base graph of local connections, onto which a random graph of shortcut edges (long range contacts) is added. In its most basic form, one starts with a k-dimensional square lattice as the base network, and then adds q directed random edges at each node, selected so that each such shortcut edge from x points to y with probability:

$$\ell(x, y) = \frac{1}{d(x, y)^k H_k(n)}$$

where d denotes lattice distance in the base graph, n the size of the network, and H_k is a normalizing constant.

Kleinberg showed that in this case so-called *greedy routing* finds a path from any point to any other in, on average, $O(\log^2(n))$ steps. Greedy routing means always picking the neighbor (either through a shortcut or the base graph) which is closest to the destination, in terms of the lattice distance d, as the next step. Since routing within the base graph is permitted, the path strictly approaches the destination, and the same point cannot be visited twice.

In order to make the model more applicable to the real world, it is desirable to use the base graph only as a distance function between nodes, and thus only use the shortcut edges when routing. The necessity of a strictly approaching path existing then disappears, and we are left with the possibility of coming to a dead-end node which has no neighbor closer to the destination than itself. Kleinberg himself dealt with this issue in [12], working on non-geographical models, and there used q (node degree) equal to $\kappa \log^2(n)$ for a constant κ. In this case it is rather easy to see that κ can be chosen so as to make the probability that any node in the network is dead-end for a given query is arbitrarily small for all sizes n.

Actually, it suffices to keep the probability that a dead-end is encountered in any given route small. By approximate calculations one can see that this should hold if $q = \Theta(\log(n) \log\log(n))^2$. In practice we find that scaling the number of links with $\log(n)$ preserves the number of paths that do not encounter a dead end for all Kleinberg model graphs we have simulated.

3 The Problem

The problem we are faced with here is this: given a network, presumed to be generated as the shortcuts in Kleinberg's model (in some number of dimensions), but without any information on the position of the nodes, can we find a good way to embed the network into a base grid so as to make the routing between them possible? This may be viewed as a parametric statistical estimation problem. The embedding is thus seen as the model's parameter, and the data set is a single realization of the model.

Seen from another perspective, we are attempting to find an algorithmic approach to answering the fundamental question of greedy routing: which of my neighbors is closest to the destination? In Kleinberg's model this is given, since each node has a prescribed position, but where graphs of this type occur in real life, that is not necessarily the case. The appeal of the approach described below is that we can attempt to answer the question using no data other than the graph of long connections itself, meaning that we use the clustering of the graph to answer the question of who belongs near whom.

Our approach is as follows: we assign positions to the nodes according to the a-posteriori distribution of the positions, given that the edges present had been assigned according to Kleinberg's model. Since long edges occur with a small probability in the model, this will tend to favor positions so that there are few long edges, and many short ones.

4 Statement

Let V be a set of nodes. Let ϕ be a function from V onto G, a finite (and possibly toric) square lattice in k dimensions[3]. ϕ is the configuration of positions assigned ! to the nodes in a base graph G. Let d denote graph distance in G. Thus for $x, y \in V$, $d(\phi(x), \phi(y))$ denotes the distance between respective positions in the lattice.

Let E denote a set of edges between points in V, and let them be numbered $1, \ldots, m$. If we assume that the edges are chosen according to the Kleinberg's model, with one end fixed to a particular node and the other chosen randomly, then the probability of a particular E depends on the distance its edges cover with respect to ϕ and G. In particular, if we let x_j and y_j denote the start and end point, respectively, of edge j, then:

$$\Pr(E|\phi) = \prod_{i=1}^{m} \frac{1}{d(\phi(x_i), \phi(y_i))^k H_G} \tag{4.1}$$

where H_G is a normalizing constant.

When seen as a function of ϕ, (4.1) is the likelihood function of a certain configuration having been used to generate the graph. The most straightforward manner in which to estimate ϕ from a given realization E is to choose the maximum likelihood estimate, that is the configuration $\hat{\phi}$ which maximizes (4.1). Clearly, this is the same as configuration which minimizes the product (or, equivalently, log sum) of the edge distances. Explicitly finding $\hat{\phi}$ is clearly a difficult problem: in one dimension it has been proven to be NP-complete [7], and there is little reason to believe that higher

[2]Roughly: The probability that a link will not be dead-end to a query decreases with $(\log n)^{-1}$. With $c \log(n) \log\log(n)$ links per node, the probability that a given node is a dead-end is thus bounded by $(\log n)^{\theta}$. θ can be made large by choosing a large c, thus making the probability of encountering a node in the $O(\log n)^2$ nodes encountered in a walk small.

[3]In our experiments below, we focus mostly on the one dimensional case, with some two dimensional results provided for comparisson purposes.

dimensions will be easier. There may be hope in turning to stochastic optimization techniques.

Another option, which we have chosen to explore here, is to use a Bayesian approach. If we see ϕ as a random quantity chosen with some probability distribution from the set of all possible such configurations (in other words, as a parameter in the Bayesian tradition), we can write:

$$\Pr(\phi|E) = \frac{\Pr(E|\phi)\Pr(\phi)}{\Pr(E)} \tag{4.2}$$

which is the a-posteriori distribution of the node positions, having observed a particular set of edges E. Instead of estimating the maximum likelihood configuration, we will try to assign configurations according to this distribution.

4.1 Metropolis-Hastings Algorithm The Metropolis-Hastings algorithm is a remarkable algorithm used in the field of Markov Chain Monte-Carlo. It allows one, given a certain distribution π on a set S, to construct a Markov chain on S with π as its stationary distribution. While simulating a known distribution might not seem extraordinary, Metropolis-Hastings has many properties that make it useful in broad range of applications.

The algorithm starts with a selection kernel $\alpha : S \times S \mapsto [0,1]$. This assigns, for every state s, a distribution $\alpha(s,r)$ of states which may be selected next. The next state, r, is selected according to this distribution, and then accepted with a probability $\beta(s,r)$ given by a certain formula of α and π. If the state is accepted, it becomes the next value of the chain, otherwise the chain stays in s for another time-step. If r is the proposed state, then the formula is given by:

$$\beta(s,r) = \min\left(1, \frac{\pi(r)\alpha(r,s)}{\pi(s)\alpha(s,r)}\right).$$

The Markov chain thus defined, with transition Matrix $P(s,r) = \alpha(s,r)\beta(s,r)$ if $s \neq r$ (and the appropriate row normalizing value if $s = r$), is irreducible if α is, and can quite easily be shown to have π as its stationary distribution, see [9], [10]. The mixing properties of the Markov chain depend on α, but beyond that the selection kernel can be chosen as need be.

4.2 MCMC on the Positions Metropolis-Hastings can be applied to our present problem, with the aim of constructing a chain on the set of position functions, $S = G^V$, that has (4.2) as its stationary distribution [4]. Let α be a selection kernel on S, and ϕ_2 be chosen by α from ϕ_1. It follows that, if we let $\alpha(\phi_1,\phi_2) = \alpha(\phi_2,\phi_1)$, and assume a uniform a-priori distribution, then:

$$\begin{aligned}\beta(\phi_1,\phi_2) &= \min\left(1, \frac{\Pr(E|\phi_2)}{\Pr(E|\phi_1)}\right)\\ &= \min\left(1, \prod_{i=1}^{m} \frac{d(\phi_1(x_i),\phi_1(y_i))^k}{d(\phi_2(x_i),\phi_2(y_i))^k}\right)\end{aligned}$$

Let ϕ_2 be an x,y-switch of ϕ_1 if $\phi_1(x) = \phi_2(y)$, $\phi_1(y) = \phi_2(x)$, and $\phi_1(z) = \phi_2(z)$ for all $z \neq x,y$. In such cases, the above simplifies by cancellation to:

$$\beta(\phi_1,\phi_2) = \min\left(1, \prod_{i \in \mathrm{E}(x\vee y)} \frac{d(\phi_1(x_i),\phi_1(y_i))^k}{d(\phi_2(x_i),\phi_2(y_i))^k}\right) \tag{4.3}$$

where $\mathrm{E}(x \vee y)$ denotes the edges connected to x or y. This function depends only on edge information that is local to x and y.

We are now free to choose a symmetric selection kernel according to our wishes. The most direct choice is to choose x and y randomly and then to select ϕ_2 as the x,y-switch of ϕ_1. This is equivalent to the kernel:

$$\alpha(\phi_1,\phi_2) = \begin{cases} 1/(n + \binom{n}{2}) & \text{if } x,y\text{-switch} \\ 0 & \text{otherwise.}\end{cases} \tag{4.4}$$

The Markov chain on S with transition matrix

$$P(\phi_1,\phi_2) = \alpha(\phi_1,\phi_2)\beta(\phi_1,\phi_2)$$

with α and β given by (4.4) and (4.3) respectively, is thus the Metropolis-Hastings chain with (4.2) as its stationary distribution. Starting from any position function, it eventually converges to the sought a-posteriori distribution.

A problem with the uniform selection kernel is that we are attempting to find a completely distributed solution to our problem, but there is no distributed way of picking two nodes uniformly at random. In practice, we instead start a short random walk at x, and use as y the node where the walk terminates. This requires no central element. It is difficult to specify the kernel of selection technique explicitely, but we find it more or less equivalent to the one above. See Section 8 below.

[4]Another way of looking at this is as an example of *Simulated Annealing*, which uses the Metropolis-Hastings method to try to minimize an energy function. In this case, the energy function is just the log sum of the edge distances, and the β coefficient is 1.

5 Experiments

In order to test the viability of the Markov Chain Monte-Carlo method, we test the chain on several types of simulated data. Working with the one-dimensional case, where the base graph is a circle, we simulate networks of different sizes according to Kleinberg's model, by creating the shortcuts through random matching of nodes, and with the probability of shortcuts occurring inverse squarely proportional to their length. We then study the resulting configuration in several ways, depending on whether the base graph is recreated after the experiment, and whether, in case it is not, we stop when reaching a dead-end node of the type described above.

We also study the algorithm in two dimensions, by simulating data on a grid according to Kleinberg's model, and using the appropriate Markov chain for this case. Finally, we study some real life data sets of social networks, to try to determine if the method can be applied to find routes between real people.

The simulator used was implemented in C on Linux and Unix based systems. Source code, as well as the data files and the plots for all the experiments, can be found at:

`http://www.math.chalmers.se/~ossa/swroute/`

6 Experimental Methodology

6.1 One-Dimensional Case We generated different graphs of the size $n = 1000 * 2^r$, for r between 0 and 7. The base graph is taken to be a ring of n points. Each node is then given $3 \log_2 n$ random edges to other nodes. Since all edges are undirected, the actual mean degree is $6 \log_2 n$, with some variation above the base value. This somewhat arbitrary degree is chosen because it keeps the probability that a route never hits a dead end low when the edges are chosen according to Kleinberg's model. Edges are sent randomly clockwise or counterclockwise, and have length between 1 and $n/2$, distributed according to three different models.

1. Kleinberg's model, where the probability that the edge has length d is proportional to $1/d$.
2. A model with edges selected uniformly at random between nodes.
3. A model where the probability of an edge having length d is proportional to $1/d^2$.

Both the latter cases are non-optimal: the uniform case represents "too little clustering", while the inverse square case represents "too much". In Kleinberg's result, the two types of graphs are shown not to have log-polynomial search times in different ways: too much clustering means not enough long edges to quickly advance to our destination, too little means not enough edges that take even closer when we are near it.

Performance on the graphs can be measured in three different ways as well. In all cases, we choose two nodes uniformly, and attempt to find a greedy route between them by always selecting the neighbor closest (in terms of the circular distance) to the destination. The difference is when we encounter a dead end – that is to say a node that has no neighbor closer to the destination then itself. In this case we have the following choices on how to proceed:

1. We can terminate the query, and label it as unsuccessful.
2. We can continue the query, selecting the best node even if it is further from the destination. In this case it becomes important that we avoid loops, so we never revisit a node.
3. We can use a "local connection" to skip to a neighbor in the base from the current node, in the direction of the destination.

For the second case to be practical, it is necessary that we limit the number of steps a query can take. We have placed this limit as $(\log_2 n)^2$, at which point we terminate and mark the query unsuccessful. This value is of course highly arbitrary (except in order), and always represents a tradeoff between success rate and the mean steps taken by successful queries. This makes such results rather difficult to analyze, but it is included for being the most realistic option, in the sense that if one was using this to try to search in a real social network, the third case is unlikely to be an option, and giving up, as in the first case, is unnecessary.

We look at each result for the graph with the positions as they were when it was generated, after shuffling the positions randomly, and finally with positions generated by a running the Markov Chain for $6000n$ iterations. It would, of course, be ideal to be able to base such a number off a theoretical bound on the mixing time, but we do not have any such results at this time. The number has been chosen by experimentation, but also for practical purposes: for large n the numerical complexity makes it difficult to simulate larger orders of iterations in practical time-scales.

Due to computational limitations, the data presented is based off only one simulation at every size of the graph. However, at least for graphs of limited size, the variance in the important qualities has been seen to be small, so we feel that the results are still indicative of larger trends. The relatively regular behavior of the data presented below strengthens this assessment.

After shuffling and when we continue at dead ends, the situation is equivalent to a random walk, since the greedy routing gains from the node positions. Searching by random walk has actually been recommended in several papers ([3], [8]), so this gives the possibility of comparing our results to that.

6.2 Two Dimensional Case We also simulate Kleinberg's model in two dimensions, generating different graphs of the size $n = 1024 * 4^r$, for r between 0 and 3. A toric grid as the base graph (that is to say, each line is closed into a loop). Shortcuts were chosen with the vertex degrees as above, and with ideal distribution where the probability that two nodes are connected decreasing inverse squared with distance (the probability of an edge having length d is still proportional to $1/d$, but as d increases there are more choices of nodes at that distance). We do this to compare the algorithm in this setting to that in the one dimensional case.

We also try, for graphs with long range connections generated against a two dimensional base graph, to use the algorithm in one dimension, and vice versa. This is to ask how crucial the dimension of the base grid is to Kleinberg's model: whether the essential characteristics needed for routing carry over between dimensions. Any conclusion on the subject, of course, is subject to the question of the performance of the algorithm.

6.3 Real World Data Finally, we test the method on a real graph of social data. The graph is the "web of trust" of the email cryptography tool Pretty Good Privacy (PGP) [1]. In order to verify that the person who you are encrypting a message for really is the intended recipient, and that the sender really is who he claims to be, PGP has a system where users cryptographically sign each others keys, thereby vouching for the key's authenticity. The graph in question is thus a sample of people that know each other "in real life" (that is outside the Internet), since the veracity of a key can only be measured through face to face contact.

We do not look at the complete web of trust, which contained about 23,000 users, but only at smaller subsets. The reason for this is two-fold. Firstly, the whole network is not a connected component. Secondly a lot of the nodes in the graph are in fact leaves, or have only one or two vertices. Under such conditions, the algorithm (or any greedy routing for that matter) cannot be expected to work.

These were created by starting a single user as the new graph's only vertex, and recursively growing the graph in the following manner. If G_n is the new graph at step n:

1. Let ∂G_n be the vertices with at least one edge into G_n, but who are not in G_n themselves.
2. Select a node x randomly from those members of ∂G_n who have the greatest number of edges into G_n.
3. Let G_{n+1} be the graph induced by the vertices of G_n and x.
4. Repeat until G_{n+1} is of the desired size.

This procedure is motivated by allowing us to get a connected, dense, "local" subgraph to study. It is closest we can come to the case where, having access to the base graph, one uses a only the nodes in a particular section of it and the shortcuts between them.

Daily copies of the web of trust graph are available at the following URL:

`http://www.lysator.liu.se/~jc/wotsap/`

7 Experimental Results and Analysis

7.1 One Dimensional Case Experimental results in the one dimensional case were good in most, but not all, cases. Some of the simulated results can be seen in 1 through 8. Lines marked as "start" show the values with the graphs as they were generated, "random" show the values when the positions have been reassigned randomly (this was not done for the random matchings case, as there is no difference from the start), and "restored" show the values after our algorithm has been used to optimize the positions.

In the ideal graph model, when the original graph is known to allow log polynomial routing, we can see that the algorithm works well in restoring the query lengths. In particular, Figure 3, where queries have been able to use the base graph, shows nearly identical performance before and after restoration.

In the cases where queries cannot use the local connections, we see that proportion of queries that are successful is a much harder property to restore than the

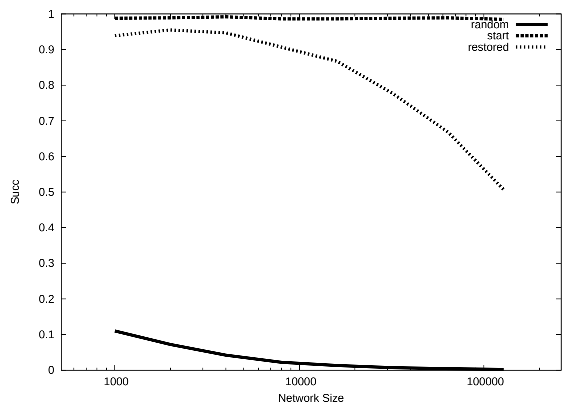

Figure 1: The success-rate of queries when terminating at dead-end nodes, on a graph generated by the ideal model.

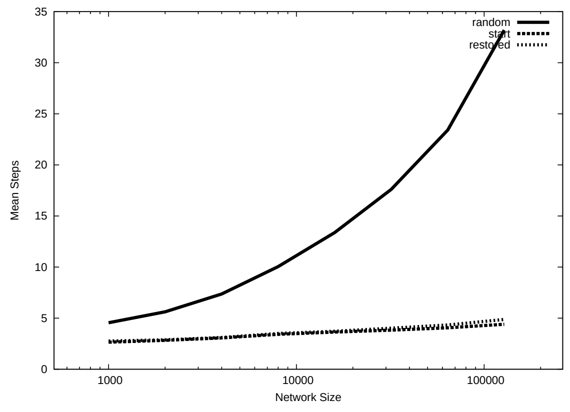

Figure 3: Mean number of steps of successful queries when allowed to use local connections, on a graph generated by the ideal model.

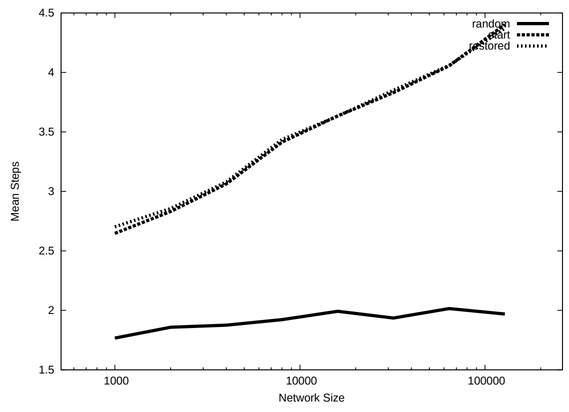

Figure 2: Mean number of steps of successful queries when terminating at dead-end nodes, on a graph generated by the ideal model.

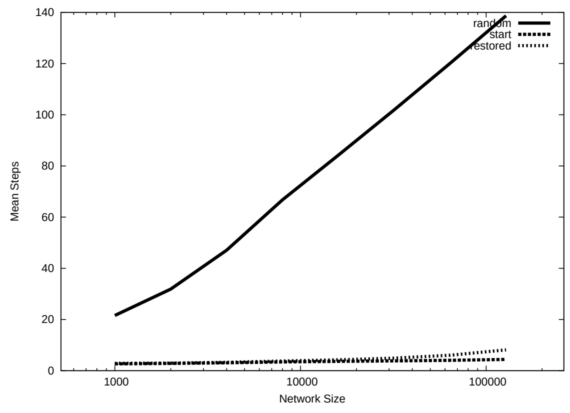

Figure 4: Mean number of steps of successful queries when terminating after $(\log_2(n))^2$ steps, on a graph generated by the ideal model.

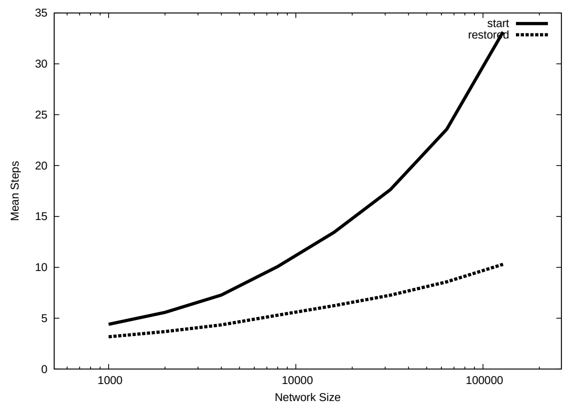

Figure 5: Mean number of steps of successful queries when allowed to use local connections, on a graph generated by random matchings.

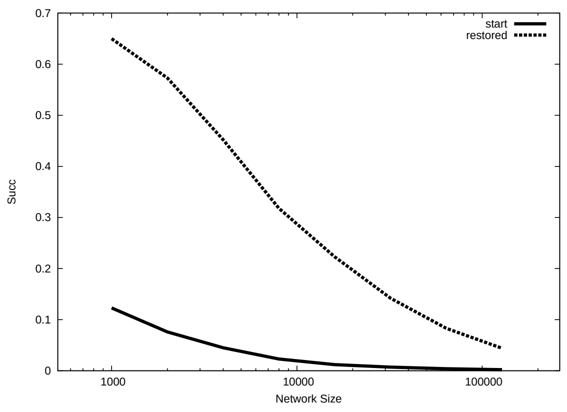

Figure 7: The success-rate of queries when terminating at dead-end nodes, on a graph generated by random matchings.

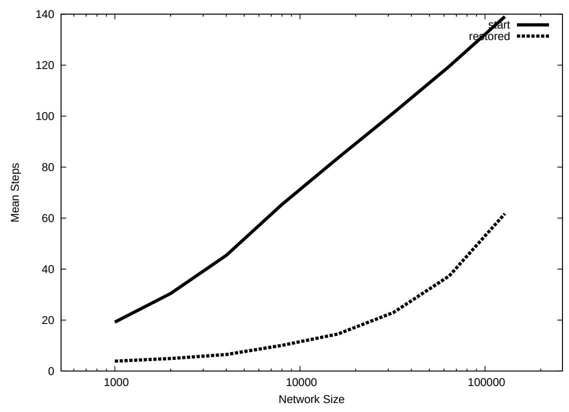

Figure 6: Mean number of steps of successful queries when terminating after $(\log_2(n))^2$ steps, on a graph generated by random matchings.

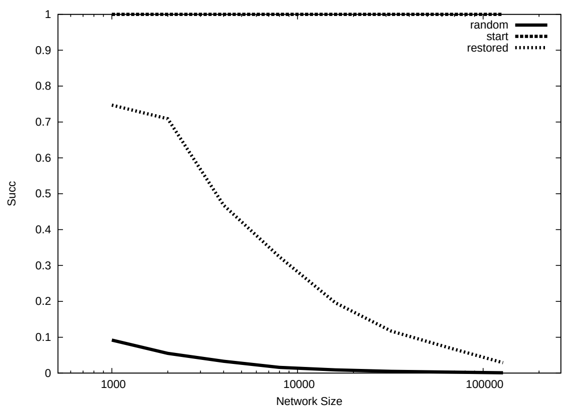

Figure 8: The success-rate of queries when terminating at dead-end nodes, on a graph generated with connection probabilities inverse square proportional to the length.

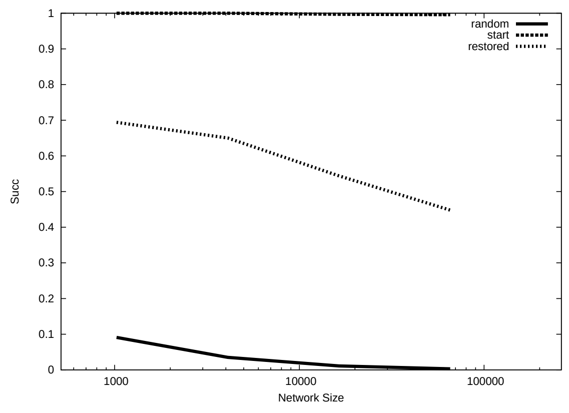

Figure 9: Matching Kleinberg's model in 2 dimensions against a graph generated according to it. Success rate when failing at dead-end nodes.

number of steps taken. Figure 1 shows this: for large graphs the number of queries that never encounter a dead-end falls dramatically. A plausible cause for this is that it is easy for the algorithm to place the nodes in the approximately right place, which is sufficient for the edges to have approximately the necessary distribution, but a good success rate depends on nodes being exactly by those neighbors to which they have a lot edges.

Along with the ideal data, two non-ideal cases were examined. In the first case, where the long range connections were added randomly, the algorithm performs surprisingly well. At least with regard to the number of steps, we can see a considerable improvement at all sizes tested. See in particular Figures 6 and 5. However, it is impossible for the success rate to be sustained for large networks when the base graph is not used - in this case there simply is no clustering in the graph - and as expected the number of successful queries does fall as n grows (Figure 7.

The other non-ideal case, that of too much clustering, was the one that faired the worst. Even though this leads to lots of short connections, which one would believe could keep the success rate up, this was not found to be the case. Both the success rate and the mean number of steps of the successful queries were not found to be significantly improved by the algorithm in this case. The results in Figure 8 if particularly depressing in this regard. It should be noted that it has been shown [14] that graphs generated in this way are not small-world graphs - their diameter is polynomial in their size, so there is no reason to believe that they can work well for this type of application.

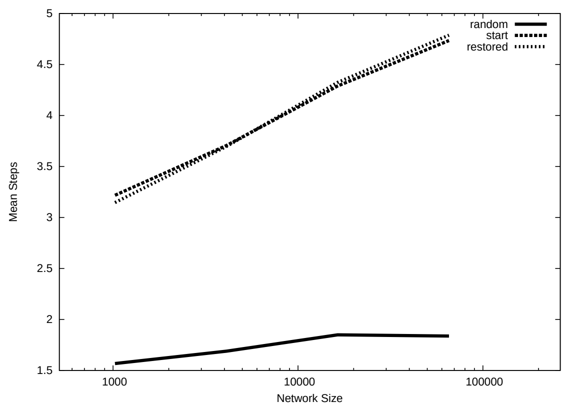

Figure 10: Matching Kleinberg's model in 2 dimensions against a graph generated according to it. Mean number of steps of successful queries when failing at dead-end nodes.

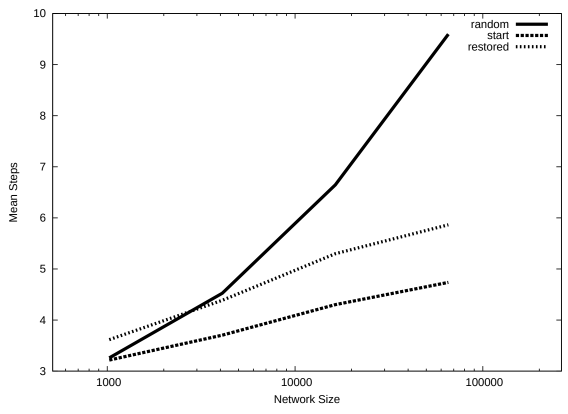

Figure 11: Matching Kleinberg's model in 2 dimensions against a graph generated according to it. Mean number of steps of queries when they are allowed to use local connections.

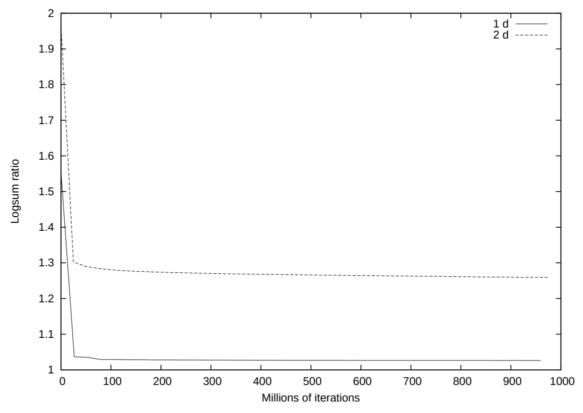

Figure 12: The target function of the optimization (log sum of shortcut distances) as the algorithm progresses. The graphs have 10000 nodes with edges generated using the ideal model. The values are normalized by dividing by the log sum of the original graph: it can be seen that we come much closer to restoring this value in 1 dimension.

7.2 Two Dimensional Case The algorithm was also simulated with a pure two dimensional model. In general, the algorithm does not perform as well as in the one dimensional case, but it performs better than against the one dimensional algorithm did on the graphs generated from non-ideal models. See Figures 9 to 11 for some of the data.

It seems that the algorithm proposed here simply does not function as well in the two-dimensional case. In Figure 12 the sum of the logarithms of the shortcut distances for a graph is plotted as the optimization is run for a very large number of iterations. It indicates that results in two-dimensions cannot be fixed by simply running more iterations, in fact, it seems like it fails to converge to one completely.

Graphs generated according to the two dimensional model were also given to the one dimensional algorithm, and vice versa. We found that data from either model was best analyzed by fitting it against a base graph of the same dimension - but the two dimensional method actually did slightly better on one-dimensional data than its own. For example at a network size of 4096, we were able to restore a success rate of 0.670 when failing at dead-ends using the two dimensional method for one dimensional data, but only 0.650 on data from the two dimensional model. This indicates that the worse performance in two dimensions may be largely due to Kleinberg's model in higher dimensions being more difficult to fit correctly.

7.3 Real World Data We treated the real world data in the same way as the simulated graphs. 2000 and 4000 vertex subgraphs were generated using the procedure defined above, the nodes were given random positions in a base graph, and then $6000n$ iterations of the Metropolis-Hastings algorithm was performed. We tried embedding the graph both in the one dimensional case (circle) and two (torus). In one dimension, the results were as follows:

Size	2000	4000
Mean degree	64.6	46.4
F Success	0.609	0.341
F Steps	2.99	3.24
C Succ	0.981	0.798
C Steps	13.4	26.0
LC Steps	4.58	7.21

Here "F Success/Steps" denotes the values when we fail upon hitting a dead end, "C Succ/Steps" when we continue and "LC steps" is the mean number of steps for queries that use the local connections at dead ends.

The data was also tested using two-dimensional coordinates and distance. The results are rather similar, with some of the tests performing a little bit better, and some (notably the success rate when failing on dead ends) considerably worse.

Size	2000	4000
F Success	0.494	0.323
F Steps	2.706	3.100
C Succ	0.984	0.874
C Steps	13.116	22.468
LC Steps	3.920	5.331

It perhaps surprising that using two dimensions does not work better, since one would expect the greater freedom of the two dimensional assignment to fit better with the real dynamics of social networks (people are, after all, not actually one a circle). The trend was similar with three-dimensional coordinates, which led to success rates of 0.42 and 0.26 respectively for the large and small graphs when failing at dead-ends, but similar results to the others when continuing. As can be seen from simulations above, the algorithm does not seem to perform very well in general in higher dimensions, and this may well be the culprit.[5]

[5]There is a general perception that the two-dimensional case represents reality, since peoples geographical whereabouts are two-dimensional. We find this reasoning somewhat specious. The true metric of what makes two people closer (that is, more likely to know one another) is probably much more complicated than

The two thousand node case has about the same degree as the simulated data from the graphs above, so we can compare the performance. From this we can see that the "web of trust" does not nearly match the data from the ideal model in any category. It does, however, seem to show better performance than the uniform matchings in some cases - most notably the crucial criteria of success rate when dropping at dead ends.

To look at the 4000 nodes case, the mean degree is considerably less than the experiments presented below, and it the results are unsurprisingly worse. In this case however, the dataset does have a lot of nodes with only a few neighbors, and it is easy to understand it is difficult for the algorithm to place those correctly.

At first glance, these results may seem rather negative, but we believe there is cause for cautious optimism. For one thing, success rates when searching in real social networks have always been rather low. In [13], when routing using actual geographic data, only 13% of the queries were successful. They used a considerably larger and less dense graph than ours, but on the other hand they required only that the query would reach the same town as the target. [2] showed similar results when attempting to route among university students. Real world Milgram type experiments have never had high success rates either: Milgram originally got only around 20% of his queries through to the destination, and a more recent replication of the experiment using the Internet [6] had as few as 1.5% of queries succeed.

Moreover, there have not been, to the authors knowledge, any previously suggested methods for routing when giving nothing but a graph. Methods suggested earlier for searching in such situations have been to either walk randomly, or send queries to nodes of high degree. With this in mind, even limited success may find practical applications.

just geography (the author of this article is, for instance, perhaps more likely to know somebody working in his field in New Zealand, than a random person a town or two away). In any case, there is a trade-off between the realism of a certain base graph, and how well the optimization seems to function, which may well motivate less realistic choices.

8 Distributed Implementation and Practical Applications

The proposed model can easily be implemented in a distributed fashion. The selection kernel used in the simulations above is not decentralized, in that it involves picking two nodes x and y uniformly from the set. However, the alternative method is that nodes start random walks of some length at random times, and then propose to switch with the node at which the walk terminates. Simulating this with random walks of length $\log_2(n)/2$ (the log scaling motivated by the presumed log scaling of the graphs diameter) did not perform measurably worse in simulations than a uniform choice (nor on the collected data in the last section)[6]. For example, in a graph of 64,000 nodes generated with the ideal distribution, we get(with the tests as described above):

Test	Success Rate	Mean Steps
Fail	0.668	4.059
Continue	0.996	6.039
Base Graph	1.0	4.33

Once the nodes x and y have established contact (presumably via a communication tunnel through other nodes), they require only local data in order to calculate the value in (4.3) and decide whether to switch positions. The amount of network traffic for this would be relatively large, but not prohibitively so.

In a fully decentralized setting, the algorithm could be run with the nodes independently joining the network, and connecting to their neighbors in the shortcut graph. They then choose a position randomly from a continuum, and start initiating exchange queries at random intervals. It is hard to say when such a system could terminate, but nodes could, for example, start increasing the intervals between exchange queries after they have been in the network long. As long as some switching is going on, of course, a nodes position would not be static, but at any particular time they may be reachable.

The perhaps most direct application for this kind of process, when the base graph is a social network between people, is an overlay network on the Internet, where friends connect only to each other, and then wish to be able to communicate with people throughout the network. Such networks, because they are difficult to analyze, have been called "Darknets", and sometimes also "Friend-to-Friend" (F2F) networks.

9 Conclusion

We have approached a largely unexplored question regarding how to apply small-world models to actually find greedy paths when only a graph is presented. The method we have chosen to explore is a direct application of the well known Metropolis-Hastings algorithm, and

[6]The most direct decentralized method, that nodes only ever switch positions with their neighbors, did not work well in simulation.

it yields satisfactory results in many cases. While not always able to restore the desired behavior, it leads to better search performance than can be expected from simpler methods like random searches.

Much work remains to be done in the area. The algorithm depends, at its heart, on selecting nodes who attempt to switch positions with each other in the base graph. Currently the nodes that attempt to switch are chosen uniformly at random, but better performance should be possible with smarter choice of whom to exchange with. Something closer to the Gibbs sampler, where the selection kernel is the distribution of the sites being updated, conditioned on the current value of those that are not, might perhaps yield better results.

Taking a step back, one also needs to evaluate other methods of stochastic optimization, to see if they can be applicable and yield a better result. No other such method, to the author's knowledge, applies as directly to the situation as the Metropolis-Hastings/simulated annealing approach used here, but it may be possible to adapt other types of evolutionary methods to it.

Also, all the methods explored here are based on the geographic models that Kleinberg used in his original small-world paper [11]. His later work on the dynamics of information [12] (and also [18]), revisited the problem with hierarchical models, and finally a group based abstraction covering both. It is possible to apply the same techniques discussed below to the other models, and it is an interesting question (that goes to the heart of how social networks are formed) whether the results would be better for real world data.

The final question, whether this can be used successfully to route in real life social networks is not conclusively answered. The results on the limited datasets we have tried have shown that while it does work to some respect, the results are far from what could be hoped for. Attempting to apply this method, or any derivations thereof, to other real life social networks is an important future task.

References

[1] A. Abdul-Rahman. The pgp trust model. *EDI-Forum: the Journal of Electronic Commerce*, 1997.

[2] L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27:187–203, 2005.

[3] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power-law networks. *Physical Review E*, 64 (46135), 2001.

[4] B. Bollobas and F. Chung. The diameter of a cycle plus a random matching. *SIAM Journal on Discrete Mathematics*, 1:328–333, 1988.

[5] A. Clauset and C. Moore. How do networks become navigable? *Preprint*, 2003.

[6] P. S. Dodds, M. Roby, and D. J. Watts. An experimental study of search in global social networks. *Science*, 301:827, 2003.

[7] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. *Theory of Computer Science*, 1:237–267, 1978.

[8] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *INFOCOM*, 2004.

[9] O. Häggström. *Finite Markov Chains and Algorithmic Applications.* Number 52 in London Mathematical Society Student Texts. Cambridge University Press, 2002.

[10] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[11] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[12] J. Kleinberg. Small-world phenomena and the dynamics of information. In *Advances in Neural Information Processing Systems (NIPS) 14*, 2001.

[13] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geograph routing in social networks. In *Proceedings of the National Academy of Science*, volume 102, pages 11623–11628, 2005.

[14] C. Martel and V. Nguyen. Analyzing kleinberg's (and other) small-world models. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 179–188, New York, NY, USA, 2004. ACM Press.

[15] S. Milgram. The small world problem. *Psychology Today*, 1:61, 1961.

[16] O. Sandberg and I. Clarke. An evolving model for small world neighbor selection. draft, 2005.

[17] D.J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness.* Princeton University Press, 1999.

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

[19] D.J. Watts and S. Strogatz. Collective dynamics of small world networks. *Nature*, 393:440, 1998.

[20] B. Yu and M. Singh. Search in referral network. In *Proceedings of AAMAS Workshop on Regulated Agent-Based Social Systems: Theories and Applications*, 2002.