CS224W Project Report: Imperfect Decentralized Search

Jeffrey Wang (jeffreyw) and Bobby Prochnow (prochnow)

December 11, 2011

1 Introduction

The problem of decentralized search has been studied for many years, beginning with Milgram's original social experiment [7]. Participants in the study were asked to forward a letter to some target in Boston, Massachusetts (choosing the person they considered most likely to know the target), and the number of steps that each letter took was observed. Milgram found that, for letters that eventually reached the target, it took only about 6 steps on average to get there, despite the fact that each person in the network had only a localized view (i.e. their acquaintances). Since then, there have been a number of studies investigating what properties of a network make it *navigable*, i.e. searchable in a small number of steps with only localized information. We call such a process *decentralized search* because it has no central view of the network.

One observation about Milgram's study is that the participants were not always able to optimize their forwarding according to the criterion given to them, which was to forward the letter to the acquaintance who was most likely to know the target. That is, the decentralized search was not "perfect" according to the metric with which it was performed. This fact has not been reflected in the theoretical studies, however, which for the most part assume the search process they specify is executed perfectly. We are interested in evaluating how an imperfect decentralized search affects network navigability, as this would provide a more realistic simulation of how people in social networks would perform such a search.

2 Related Work

Watts et al. [8] explain network navigability via decentralized search in large graphs by representing people's identities as a set of hierarchies (H total). Each hierarchy is a tree (e.g. breakdown of university into schools, departments, groups, etc.), and each person is assigned a coordinate in the hierarchy which is a leaf of the tree. This corresponds to the group that the individual is in with respect to that hierarchy. In their model, within any single hierarchy, the probability of a link existing between any two people is a power law distribution dependent on the height of the least common ancestor (LCA) of the two people in the hierarchy (which is equivalent to choosing a height x with probability $p(x) = ce^{-\alpha x}$ and then choosing a random node at that height). This is to capture the idea that people who are similar along a certain dimension, e.g. occupation or geographic location, are more likely to be connected. Watts et al. consider only hierarchies which are identical in size; then the distance between any two people is defined to be the minimum height of the LCA across all hierarchies. To perform decentralized search in the network, the adjacent node which minimizes the distance to the destination is chosen. They present findings which indicate that there exists a broad region in the space of (α, H) for which the network is navigable, hinting at an explanation of why it may be in real social networks.

Clauset and Moore [3] consider a network which is based on a finite d-dimensional lattice in which each node is adjacent to its neighbors in the lattice and also has a single long-range link. They simulate a rewiring process where a node performs a decentralized search to a destination, but if the search takes more than a threshold number of steps, they rewire the single long-range link to where the search reached. Interestingly, this process leads to a distribution of edges which follows a power law, i.e. l^{α} where l is the Manhattan distance between nodes in the lattice, which is similar to the model proposed by Kleinberg [5] that was shown to be navigable.

Chandra and Arora [2] apply the abstract notion of decentralized search to the field of networking in an attempt to produce scale-free peer-to-peer networks. A peer-to-peer network consists of a large number of computers, each with different sets of content to distribute. The authors showed that it was possible to use decentralized search (instead of having a centralized tracking server) to effectively search the network. The success of this search required a variation of the rewiring process discussed in Clauset and Moore [3] - so when a computer performed an unsuccessful search, it would rewire one of its connections to the computer at which the search terminated.

We did not find any studies that empirically or theoretically discussed the subject of imperfect decentralized search.

3 Hierarchical Networks

The primary model of interest is the one presented in Watts et al., which is constructed by creating a set of H hierarchies and using the heuristic measure of minimum distance between the two nodes within any hierarchy, where distance in a hierarchy is defined as the height of the least common ancestor (LCA) of the nodes. A hierarchy is defined by a tree of height l and branching factor b where each leaf corresponds to a group of nodes all of which have the same coordinate within that hierarchy. The edges are randomly created according to a power law distribution based on the heuristic measure. Watts et al. showed that this network is searchable using perfect decentralized search for various parameters. Two limitations of their model we will address are the network size and the degree of each node. Our results are generalizable to very large networks closer in scale to real social networks, and we use a more flexible way of choosing the degree of each node. We believe that these modifications to the model capture a fairly realistic representation of real-world hierarchies to test decentralized search on.

3.1 Generating Random Hierarchical Networks

There are a number of parameters that control a randomly generated a hierarchical network:

- H, the number of hierarchies,
- l, the number of levels in each hierarchy,
- b, the branching factor of each hierarchy,
- g, the average number of nodes in each hierarchy coordinate group,
- α, the rate parameter of the exponential distribution over LCA height,
- $d(x)$, the degree distribution of the nodes.

The resulting network $G(V, E)$ has $|V| = gb^l$ nodes and the expected number of edges is $|E| = |V| \cdot \mathbb{E}[d(x)]$. For each node, we choose the coordinates for each hierarchy randomly from the interval $[0, b^l)$ corresponding to the leaf in the hierarchy tree that the node is in the group of. Hence g is not a fixed group size as in Watts et al. but the expected number of nodes in each group, and the actual distribution will be binomial. We choose this method of assigning coordinates because it is more realistic without complicating the structure of the network.

For generating edges, we consider each node s individually. We choose a degree d_s randomly from the distribution $d(x)$ and proceed to generate d_s edges $s \to t$ by the following procedure:

1. choose a random hierarchy h;
2. choose a distance x from the distribution $p(x) \propto e^{-\alpha x}$;
3. choose t uniformly at random among all nodes whose LCA in hierarchy h with s has height x (if $x = 0$, ensure $s \neq t$).

Doing this explicitly, however, takes $O(|E| \cdot |V|)$ time because for each edge we need to sample across potentially all $|V|$ nodes. This is impractical because we would like to experiment with $|V| > 10^7$ at least to address the issue of scale that was not sufficiently investigated in Watts et al. As such, we describe an improved algorithm that allows us to generate random hierarchical networks much more efficiently.

The idea is to exploit the hierarchy structure to do the second step of the algorithm more efficiently than in $O(|V|)$ time. When choosing hierarchy coordinates for each of the nodes in the graph, we also maintain some extra information about how the nodes are distributed in each hierarchy. Let $HL(h, i)$ be the list of nodes with coordinate i in hierarchy h and $HC(h, i, j)$ be the number of nodes under subtree j of level i in hierarchy h (levels are numbered from bottom to top and subtrees are numbered from left to right). For example, $HC(h, l, 0) = |V|$ is the count of all nodes in the graph, while $HC(h, l-1, 0)$ through $HC(h, l-1, b-1)$ count the nodes according to the first branch. Lastly, $HC(h, 0, i)$ is the number of nodes with coordinate i in hierarchy h.

Using these precomputed values, we can improve the time it takes to select a random node at distance x significantly. If $x = 0$, then just choose t randomly from $HL(h, s_h)$, where s_h is the coordinate of s in hierarchy h. Otherwise, let u be the x-th ancestor of s in hierarchy h, i.e. the hierarchy node at level x that is an ancestor of s. Similarly, let v be the $(x-1)$-th ancestor of s in hierarchy h. Then t should be randomly chosen from nodes in the subtree of u but not in the subtree of v, since these have distance less than x. From here, we choose a random index between 0 (inclusive) and $HC(h, x, u) - HC(h, x-1, v)$ (exclusive). Given this index and the HC values, we can decide which child of u to traverse and which index within that child's subtree t is, effectively recursing one level down the hierarchy. Thus choosing t will only take $O(lb)$ time since at each level we decide among the b

children to traverse, and once we reach the bottom level of the hierarchy we just choose a node randomly from the corresponding HL. Since $|V| = gb^l$, this is a substantial improvement in efficiency that allows us to generate much larger networks. In particular, it brings the total run-time of hierarchical graph generation to $O(|E|lb)$.

3.2 Implementation

We originally used NetworkX as our graph library for its simplicity, which was convenient for prototyping. We quickly found out, however, that it was not possible to scale NetworkX easily to 10^7 nodes due to memory limitations. Because scale is a significant aspect of our project, we ported the Python code to C++ using our own custom hierarchical network representation. Since we do not actually need a lot of general network functionality provided by NetworkX and SNAP, it was easiest and most efficient to do it this way. As a result, we are able to handle networks of over 100x the size as we were able to in NetworkX with similar improvements in speed as well. It takes about 30 minutes to generate a graph with 25 million nodes and 2.5 billion edges on an 8-core 2.7GHz processor with 32GB of RAM.

4 Modeling Imperfect Decentralized Search

Let $G(V, E)$ be a network and metric $h : V \times V \to \mathbb{R}$ be heuristic measure of distance between two nodes in the network (e.g. lowest LCA in hierarchy, Manhattan distance, or probability of being adjacent). Let $A_h : V \times V \to V$ be a decentralized search algorithm based on heuristic h which outputs the next step in a decentralized search (arguments are current node and target node). In previous models of decentralized search, e.g. Watts et al. , they have always used

$$A_h(s) = \operatorname*{argmin}_{v:(s,v)\in E} h(v, t)$$

where t is the target node, i.e. choose the neighbor of s which minimizes the heuristic distance to the target.

We would like to model an imperfect decentralized search mechanism, where a neighbor v is chosen with probability proportional to $p(h(v, t))$ for some function p. That is, for all v such that $(s, v) \in E$,

$$\Pr[A_h(s) = v] \propto p(h(v, t)).$$

In general, we would like p to be a decreasing function, indicating that the probability of traversing that edge decreases as the heuristic distance between v and t increases. It also controls the degree to which a difference in the heuristic distance affects the probability of choosing that edge. We observe that we can approximate the perfect search by taking $p(v) = ce^{-\beta h(v,t)}$ (for normalization constant c) and letting $\beta \to \infty$ (softmax function), and we can simulate a uniform distribution over all neighbors by choosing $\beta = 0$. This function seems general enough that we can just vary β and determine whether the results found in previous work still hold.

5 Navigability

Here we discuss the issue of what it means for a network to be navigable. One possible definition is presented in Watts et al. [8], which is as follows. Let p be a probability that any step in the decentralized search will fail, e.g. in the case of Milgram's experiment, the letter is not forwarded. Then the average path length L should be such that the probability of a search reaching the target is at least some value r, i.e. $(1 - p)^L \geq r$. They use a fixed p, r, and thus threshold of L regardless of the network size, which seems unrealistic. This is also why their regions of navigability (with respect to α and H) shrink as the network size increases, a somewhat counterintuitive result.

In this work we avoid the issue of absolute navigability because it is more or less arbitrary in nature (based on choices of p and r). Instead, we run decentralized search with some probability of failure at each step and simply observe the fraction of these which are successful, i.e. reach the target. The success rate of the perfect search is the benchmark that we compare imperfect search against. We call the ratio of these success rates the *relative navigability* of a decentralized search algorithm on a particular network. Previous work suggests that perfect decentralized search produces absolute navigability on a wide range of hierarchical networks so high relative navigability should be indicative of a good search algorithm.

6 Experiments and Results

We performed a number of experiments on both perfect and imperfect decentralized search using the hierarchical graph model. The primary aim is to understand how the parameters of hierarchical graph generation and imperfect search affect relative navigability, but we consider a couple of other interesting aspects of the networks as well.

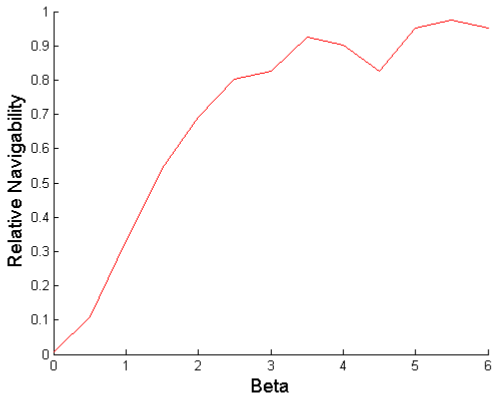


Figure 1: Relative navigability for imperfect decentralized search.

6.1 Varying β in Imperfect Search

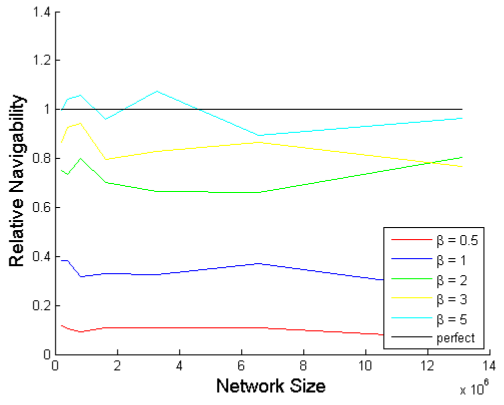
One of the initial questions we want to explore is how the choice of β - the degree to which decentralized search is imperfect - affects the overall performance of search. We use a network which is similar parameters to those used in Watts et al. with $H = 2$ hierarchies, a branching factor $b = 2$, $l = 13$ levels in hierarchies, groups of expected size $g = 100$, rate parameter $\alpha = 1$, and constant node degree of $d(x) = 99$ friends. Note that our graph contains $2^{13} \cdot 100 = 819{,}200$ nodes, twice that of the largest network in Watts et al. 's experiments.

For each value of β, for 100,000 random pairs of points, we performed decentralized search with $p = 0.25$ chance of terminating at each step. We recorded the relative navigability observed across these experiments in Figure 1. As we expect, the success rate has an upward trend as we increase β, approaching that of perfect decentralized search. We note, however, that even for modest values of β, e.g. $\beta = 1.5$ or $\beta = 2$, the relative navigability is over 50%. To put this in context, $\beta = 1.5$ means the search process is only $e^{1.5} \approx 4.5$ times more likely to choose a node at distance d over a node at distance $d + 1$, so there is considerable room for error (with respect to perfect search). When $\beta = 5$, the ratio of these probabilities becomes $e^5 \approx 150$ which means that the imperfect search will very closely approximate the perfect search, so we see the relative navigability approach one.

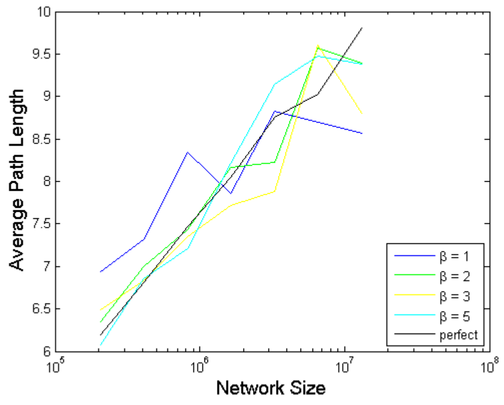
6.2 Network Size

In this experiment, we generated networks with the same parameters as Section 6.1 except letting the number of levels in the hierarchy range from $l = 11$ (204,800 nodes) to $l = 17$ (13,107,200 nodes) in order to observe how relative navigability is affected by network size. We let β take the values $0.5, 1, 2, 3, 5$ which provides a representative set of data points. In general, larger networks lead to lower absolute success rates and longer average path lengths, but in fact relative navigability appears to be unaffected (see Figure 2(a)). We consider this a strength of imperfect decentralized search, and it suggests that such an algorithm can still be effective for large real-world networks.

One of our criticisms with the work of Watts et al. was that they fixed the path length irrespective of network size, so we would like to address the growth rate of the path length as the network gets larger. The average path lengths of the successful searches for each network size are shown in Figure 2(b), for each value of β as well as perfect search. We omit $\beta = 0.5$ here because the path lengths are noisy due to very low absolute success rates (on the order of 10^{-4}), and we see that $\beta = 1$ already displays high variance. Overall these grow logarithmically, empha-

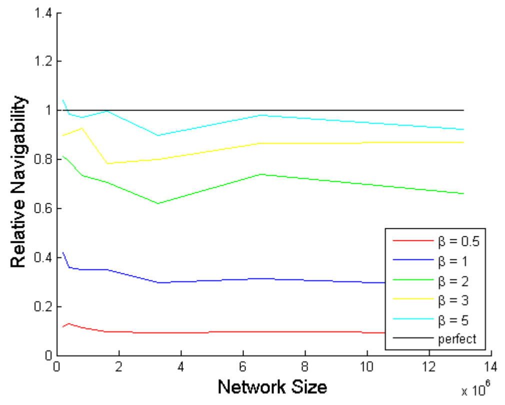


(a) Relative navigability

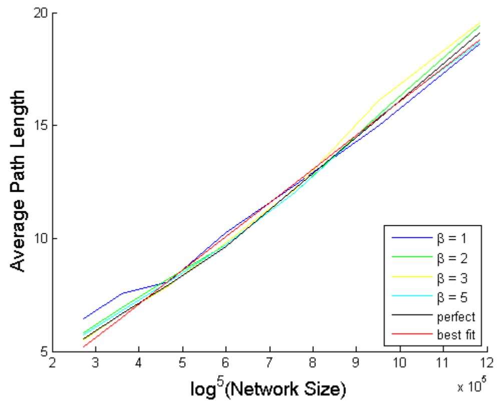


(b) Average path lengths

Figure 2: Imperfect Search for varying network sizes (p fixed).



(a) Relative navigability



(b) Average path lengths

Figure 3: Imperfect Search for varying network sizes (p adjusted).

sizing the small diameter aspect of real-world networks. We note, however, that there is a hidden variable p, the probability of failure at each step of the search. In practice, p determines an upper bound on the average path length because long paths have a very low probability of succeeding.

In order to normalize for this effect, we ran another experiment where we adjusted p so that the success rate of the perfect search was constant and observed the resulting average path lengths. The results are shown in Figure 3(b) with the x-axis scaled so that the data points lie on a line. Since the value of p decreases as the network size grows to compensate for it being harder to find the target node, the average path length naturally increases. In fact, we observed that it grows as $O((\log N)^5)$ with a linear fit of $r^2 = 0.9966$ in the graph shown. While this is much faster than the growth with a fixed value of p, it is still polylogarithmic and thus confirms our belief that path lengths grow slowly with network size.

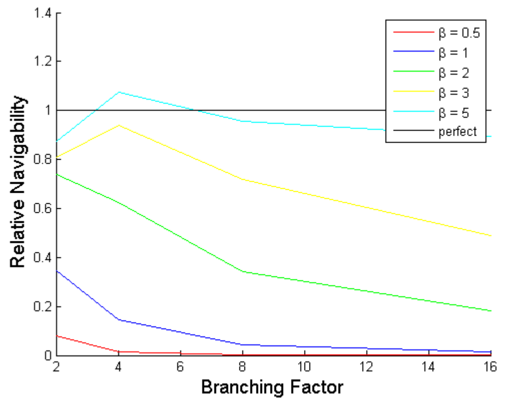


Figure 4: Relative navigability for varying branching factors.

6.3 Hierarchy Branching Factor

While the results above show that the size N of the network does not seem to affect the performance of imperfect search, this says nothing about how imperfect search is affected by changes in the network structure. One parameter that adjusts the structure of the network is the branching factor b - and all experiments performed thus far have fixed $b = 2$. By using the same parameters used in Section 6.1 except growing b exponentially while decreasing l exponentially (to hold the network size $N = 409600$ constant), we can see how this change in structure affects the performance of imperfect search.

As we can see in Figure 4, increases in the branching factor b has a noticeably detrimental effect on the relative navigability for imperfect search. While this seems fairly abstract, these results are actually a useful confirmation of network search intuition. Essentially, the branching factor represents how granular the hierarchies are in the network being searched. To see this, consider the following trivial example. If we have $N = 16$ and a branching factor of 4 (group size of $g = 1$), then a given node has 3 neighbors at distance 1, and 12 neighbors at distance 2. But if we lower b to 2 (while keeping $N = 16, g = 1$), then suddenly a node has 1 neighbor at distance 1, 2 neighbors at distance 2, 4 neighbors at distance 3, and 8 neighbors at distance 4. This demonstrates how a lower branching factor represents a more granular search heuristic.

6.4 Degree Distribution

The work of Watts et al. made the assumption that the degree distribution of nodes in the hierarchical network was constant. Since this is far from the case in real social networks, we believe it is valuable to consider a more realistic degree distribution

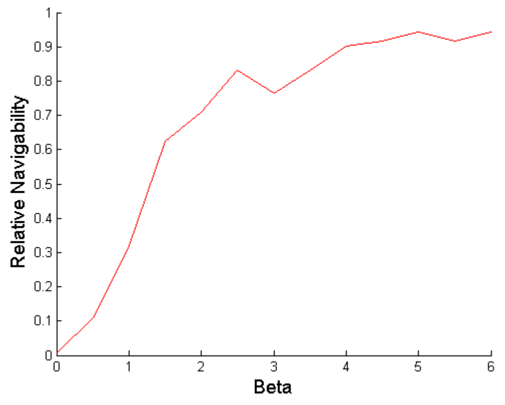


Figure 5: Relative navigability for power law degree distribution.

to see how both perfect and imperfect decentralized search perform. We use a power law degree distribution $d(x) \propto x^{-\gamma}$ with $\gamma = 2.5$ to emulate real-world networks [4]. There have been a number of previous studies which use power law degree distributions (e.g. [2], [6]) in the context of decentralized search with different networks and heuristics, so we believe it is valuable for us to consider. We bound the degrees such that $40 \leq d(x) \leq 1500$ so that $E[d(x)] \approx 99$, allowing us to compare the results to those of Section 6.1. Other than the degree distribution, we use the same parameters for generating the hierarchical networks.

One might expect networks with a power law degree distribution to be more navigable due to the existence of "hubs," i.e. highly-linked nodes, which generally reduce the diameter of the network. As shown in Figure 5, however, this is not the case; in fact, using a power law degree distribution reduces the success rate of decentralized search slightly. We hypothesize that this is due to the existence of a large number of lower-degree nodes that searches are run to/from. Nevertheless, as expected, these networks still appear to have good relative navigability when compared to the networks in Section 6.1, providing further verification that the property of navigability is shared by a wide range of networks and, in particular, real-world social networks with power law degree distributions.

7 Conclusion and Future Work

Over the years, there have been numerous empirical and theoretical studies about the small world phenomenon first documented by Milgram [7]. These previous attempts assumed that all decentralized search was performed perfectly, which is not necessarily a reasonable assumption. This paper set out to model and investigate the properties of imperfect decentralized search. We modeled imperfect search by choosing a neighbor v as the next step with probability $p(v) = ce^{-\beta h(v,t)}$ with normalization constant c and tunable parameter β.

Our initial batch of experiments confirmed that for a well-studied hierarchical network structure [8], imperfect search performs fairly well for relatively low values of β. For instance, the value of $\beta = 1.5$ (which has a considerable degree of imperfection) has a relative navigability of 50%. Slightly higher values of β quickly approach perfect decentralized search, as expected.

Our experiments also studied the effects of network size on the average path length for successful searches. For both perfect and imperfect search, holding the drop rate p fixed demonstrated that average path length grew logarithmically with respect to network size, as expected; however, a criticism of this experiment is that with a constant p, the potentially longer paths are less represented in the statistics. We re-executed the experiments by modifying p to hold the overall success rate constant, and determined that average path length grew polylogarithmically with respect to network size. By the definition proposed in Kleinberg [5], this indicates that the network exhibits the "small-world phenomenon," even when imperfect decentralized search is used. Regardless of the value of p, the performance of imperfect search relative to perfect search remained fairly constant as the network grew, indicating that the size of the network appeared to have little effect on the relative performance of imperfect search.

Given that network size did not affect relative navigability, we then wanted to explore the effect of network structure on the performance of imperfect search. By modifying the branching factor b and holding the network size N constant, we determined that lower values of b dramatically improve performance of imperfect search. Within the hierarchicial network model, a lower branching factor represents a more granular heuristic (which is used during decentralized search). Thus, in this model, when attempting to improve imperfect decentralized search, improving the heuristic is considerably more worthwhile than shrinking the network size.

Ultimately, these results help to verify some of the real-world phenomena observed by Milgram [7] and others: even when the decentralized search is error-prone (as it would be with human participants), we still achieve the results we'd expect. This paper solely focused on the hierarchical model presented in Watts [8]. Interesting future work could include investigating the properties of imperfect search on different graph structures, such as a small-world graph, a lattice structure [3], or a real social network [2].

References

[1] L. A. Adamic and E. Adar. How to search a social network. Social networks, 27 3, 187-203, 2005.

[2] P. Chandra and D. Arora. Decentralized Search in Scale-Free P2P Networks. Proc. 16th International Conference on Parallel and Distributed Systems, 776-781, 2010.

[3] A. Clauset and C. Moore. How Do Networks Become Navigable? arXiv:cond-mat/0309415v2, 2003.

[4] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. SIAM Review 51(4), 661-703, 2009.

[5] J. Kleinberg. The small-world phenomenon: an algorithmic perspective. Proc. 32nd ACM Symposium on Theory of Computing, 163170, 2000.

[6] O. Simsek and D. Jensen. Decentralized Search in Networks Using Homophily and Degree Disparity. Proc. 19th International Joint Conference on Artificial Intelligence, 304-310, 2005.

[7] J. Travers and S. Milgram. Sociometry 32, 425, 1969.

[8] D. J. Watts, P. S. Dodds, and M. E. J. Newman. Identity and Search in Social Networks. Science, 296, 1302-1305, 2002.