A Navigation Algorithm Inspired by Human Navigation

Vijesh M
Student, PESIT, Bangalore, India
Intern, ISI, Chennai, India
mv.vijesh@gmail.com

Sudarshan Iyengar
ISI
Chennai, India
sudarshaniisc@gmail.com

Vijay Mahantesh SM
Student, PESIT, Bangalore, India
Intern, ISI, Chennai, India
vijaym123@gmail.com

Amitash Ramesh
Student, DSCE, Bangalore, India
Intern, ISI, Chennai, India
amitashr@gmail.com

C Pandurangan
ISI
Chennai, India
prangan55@gmail.com

Veni Madhavan
Computer Science and Automation
Indian Institute of Science
Bangalore, India

***Abstract*—Human navigation has been a topic of interest in spatial cognition from the past few decades. It has been experimentally observed that humans accomplish the task of way-finding a destination in an unknown environment by recognizing landmarks. Investigations using network analytic techniques reveal that humans, when asked to way-find their destination, learn the top ranked nodes of a network. In this paper we report a study simulating the strategy used by humans to recognize the centers of a network. We show that the paths obtained from our simulation has the same properties as the paths obtained in human based experiment. The simulation thus performed leads to a novel way of path-finding in a network. We discuss the performance of our method and compare it with the existing techniques to find a path between a pair of nodes in a network.**

***Keywords*-search algorithm; human navigation; hotspots, path concatenation, center strategic paths;**

I. Introduction

The problem of finding a target vertex in a network where one is provided with just the local information is a well studied problem. Starting from the work of Milgram in 1967 [16], the current methods include routing using full-tables [9], interval routing [8], [9], greedy routing [11], [15], geographic routing [11], compass routing [11], etc., These routing techniques are mainly used in transportation networks and in wired as well as wireless communication networks. For detailed survey one can refer to [8], [9], [11]. For surveys of the routing methods in social networks and email networks in particular, one is referred to [1], [2], [15].

In the classical small world experiment by Milgram [16], the letter from Nebraska was to reach Boston. This is a scenario where the sender actively needs to participate in the process of helping the letter reach the destination, while the receiver remains passive.

Motivation

Sudarshan et al., in [12] presents a network analytic approach to understand human navigation. Their experiment comprises of making human participants navigate on a network that is unknown to them. At any instance, participant can view the adjacent nodes of the node that s/he is currently present. Several source-destination node pairs are given to the participant. They show that the participants learn a few landmark nodes which they use as via media to navigate. The strategy of human participants in the above mentioned experiment to learn centers of a network and the pursuit of paths through the centers has motivated the proposal of this novel navigation algorithm.

The Technique

In an unweighted graph $G(V, E)$, given a source vertex s and a destination vertex t, a straight forward approach to establish a path between s and t is to take a random walk from s till we reach t. One way to better this random walk method is to take a two way random walk, one from the source s and the other from the destination t and stop once the two random walks intersect. This method is expected to be quicker than the first in terms of number of hops that is required to establish a path. We provide empirical results to support this.

We describe our algorithm (*path concatenation algorithm* or PCA) and compare it with two other path finding strategies based on random walks. We apply PCA on various synthetic networks and discuss its performance. We show that PCA yields center-strategic paths on scale free networks - as was the case with the paths that the participants took in the two experiments.

II. Related Work

Kleinberg's work [13] on navigation in a small world, highlighted the fact that it is easier to find short chains between points in *some* networks than *others*. Adamic et al., in [2] introduced several local search strategies to find a path to the given destination vertex.

A rigorous graph theory based approach to path finding strategy was proposed by Gavoille et al., [10]. They provide a method of navigation based on vertex labeling. The established labeling strategy allows one to compute the distance between any two vertices directly. A detailed survey on decentralized search algorithms on networks that exhibit small-world phenomena is given by Kleinberg in [14].

Ozgur et al., [18] proposes a navigational technique based on a hybridized method of using degree and homophilly. Cajueiro et al., in [4] proposed a learning framework based on a First-Visit Monte Carlo Algorithm. They show that the *navigation difficulty* and *learning velocity* are strongly related to the network topology.

Recently a novel method of navigating a network using the underlying spanning tree was proposed by Dragan et al. in [6]. They reduce the navigation problem on a graph G to that of the underlying spanning tree and navigate on the latter.

On the application front, such navigational techniques are useful in a problem of finding paths between vertices in peer-to-peer systems. Crespo et al., [5] in their work on routing indices on Peer-to-Peer systems, propose a novel method of query forwarding from vertices to their neighbors that are more likely to have answers. They provide different novel routing schemes and evaluate their performance.

III. Notations and Notions

A graph $G = (V, E)$ is composed of a set of nodes V and a set of edges E, with $|V| = n$ and $|E| = m$. Two nodes are said to be *connected* if there exists a path between them. The *distance* $d(v_1, v_2)$ between any two nodes is defined as the length of a shortest path between them, or set to ∞ if there exists no path between them. Given $v \in V$, $N_G(v)$ denotes the set of all adjacent vertices of v. Let $W_u = (u = v_0, v_1, v_2, ...)$ denote a random walk starting from the vertex u. By $W_u[i]$ we mean the vertex at the ith step of the random walk W_u. e.g. $W_u[2] = v_2$. Let $T(W_u, i)$ denote the set $T(W_u, i) = \{v_0, v_1, ..., v_i\}$ of all vertices that are being visited by the random walk W_u during the first i steps.

A *centrality index* is a real-valued function $C : V \to \mathbb{R}$ on the nodes. The intuition is that the higher the value of this function, the more central this node is for the network. There are various indices; in this article we use the so-called *closeness centrality* [17] $C_C(v)$, which is defined as the reciprocal of sum of the distances of v to all other nodes. For any given graph, a centrality index can be used to define a ranking on the nodes, by sorting the nodes non-decreasingly by their centrality value. Ties if any, are broken arbitrarily. we denote the rank of a vertex by $R(v)$.Throughout the paper, we assume that the graph is an undirected simple graph, unless otherwise stated.

IV. The Algorithm

The path concatenation algorithm is split into two components: The Learning Phase and The Navigation Phase.

The learning phase deals with the preprocessing of the graph G. Once the preprocessing phase is complete, the navigation phase describes how one can find a path between a given pair of vertices.

A. Learning Phase

Let G_d be a distinct graph with vertex set V and edge set E_d. E_d is the directed edge set formed by splitting each edge in E into two directional components. The learning phase involves updating the directed graph G_d, based on the random walks taken on G. Let $u, v \in_r V$ be two randomly picked vertices. Consider the random walks W_u and W_v. As defined above, let $W_u = (W_u[0], W_u[1], ...)$ and $W_v = (W_v[0], W_v[1], ...)$.

Consider least i such that $T(W_u, i) \cap T(W_v, i) \neq \emptyset$. Such an i guarantees a vertex $h \in T(W_u, i) \cap T(W_v, i)$. Clearly, one can construct a path from u to h and another path from v to h. Let us call these directed paths $P_{u,h}$ and $P_{v,h}$ respectively. Let $l(P_{u,h})$ and $l(P_{v,h})$ denote the respective path lengths.

We define two reward functions on V and E_d as follows:

1) Vertex Reward: Consider a function $flag : V \to \mathbb{Z}$, where $flag(v) = 0, \forall v \in V$ to begin with. In an iteration, we pick a random pair of vertices u and v, and take random walks W_u and W_v. Once the walks intersect at a vertex, say h, we increment $flag(h)$ by 1, i.e. $flag(h) = flag(h) + 1$.

2) Edge Reward: Given the graph $G(V, E)$, at the end of the learning phase, we need to obtain a weighted directed graph $G(V, E_d)$. We define a reward function $R : E_d \to \mathbb{R}$. We first initialize $R(u, v) = 0, \forall (u, v) \in E_d$. For every edge (a, b) belonging to the path $P_{u,h}$, we increment the reward of that edge by $R(a, b) = R(a, b) + \frac{1}{l(P_{u,h})}$. A similar action is performed for all edges along the path $P_{v,h}$. Algorithm 1 depicts the pidgin code of the algorithm for the Learning Phase.

Figure. 1 illustrates the learning phase of Algorithm 1. u and v are the chosen pair of vertices. The red walk from u and the green walk from v indicate the random

Input: $G(V, E)$, α
Output: $G_d(V, E_d)$, R

1: $R(a, b) \longleftarrow 0, \forall (a, b) \in E_d$
2: $flag(v) \longleftarrow 0, \forall v \in V$
3: **for all** $(u, v) \in V \times V$ **do**
4:
- Take random walks $W_u = (W_u[0], W_u[1], ...W_u[i])$ and $W_v = (W_v[0], W_v[1], ...W_v[i])$ until $T(W_u, i) \bigcap T(W_v, i) \neq \emptyset$.
- Pick $h \in T(W_u, i) \bigcap T(W_v, i)$ Compute $P_{u,h}$ and $P_{v,h}$
- $l(P_{u,h}) \longleftarrow$ path length of $P_{u,h}$ $l(P_{v,h}) \longleftarrow$ path length of $P_{v,h}$
- $flag(h) \longleftarrow flag(h) + 1$
- **for** every directed edge $(a, b) \in P_{u,h}$ **do** $R(a, b) \longleftarrow R(a, b) + \frac{1}{l(u,h)}$
- **for** every directed edge $(c, d) \in P_{v,h}$ **do** $R(c, d) \longleftarrow R(c, d) + \frac{1}{l(v,h)}$

5: **end for**
6: Let $HotSpotOrdering \longleftarrow$ list of vertices in descending order of their $flag$ values (with ties arbitrarily broken).
7: Let the first α number of vertices from $HotSpotOrdering$ constitute the $HotSpots$ set.
8: Compute and store the shortest paths between all pairs of vertices in the $HotSpot$ set. Given $u, v \in HotSpots$, $HotSpotLookup(u, v)$ returns a shortest path from u to v.

Algorithm 1: The Learning phase

walks W_u and W_v respectively. h is the vertex at which the intersection has occurred. The blue path from u and the pink path from v indicate $P_{u,h}$ and $P_{v,h}$ respectively. $l(P_{u,h})$ and $l(P_{v,h})$ are the lengths of the blue and pink path respectively. The directed edges along the blue path and the pink path are rewarded with $\frac{1}{l(P_{u,h})}$ and $\frac{1}{l(P_{v,h})}$ respectively. $flag(h)$ is incremented by 1.

B. Navigation Phase

The navigation phase makes use of the reward functions $flag$ and R to find a path between a given source and destination. Both these information are embedded in $G_d(V, E_d)$. The technique involves 3 stages:

1) Finding a path from s to h_s, where $h_s \in HotSpots$
2) Finding a path from t to h_t, where $h_t \in HotSpots$
3) Concatenating the paths s to h_s, h_s to h_t and h_t to t.

The paths s to h_s and t to h_t are constructed by successively choosing edges with the highest weight. We call this the *greedy traversal technique*. This is illustrated in Figure 2. The paths thus obtained are finally concatenated to get a path from s to t. See Figure 3 for illustration.

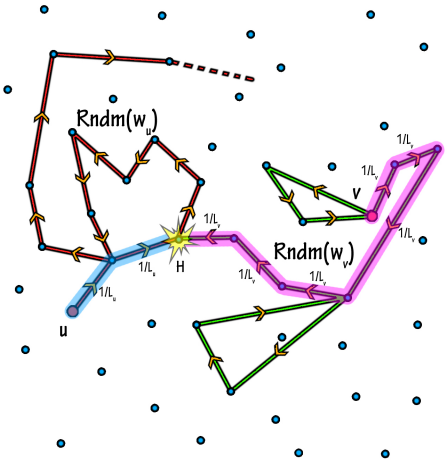

Figure 1. The Learning Phase

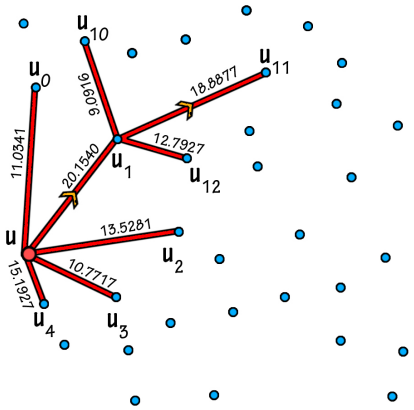

Figure 2. Greedy Traversal Technique

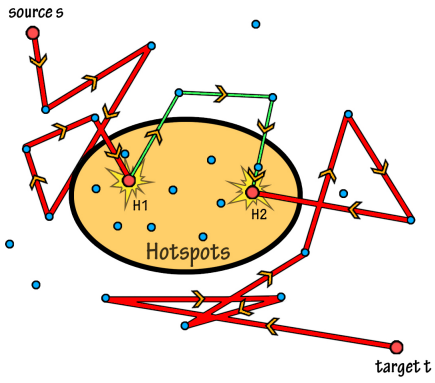

Figure 3. The Navigation Phase

V. Results and Discussions

We discuss in this section, the results that were obtained when PCA was applied to several classes of graphs. We mainly test our algorithm on two types of synthetic networks: Barabasi-Albert graphs [3] and Erdos-Renyi graphs [7].

A. Selection of optimum value of α

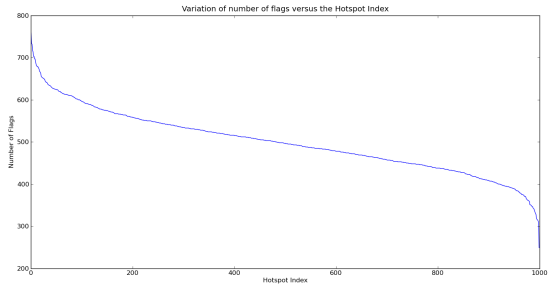

Figure 4. Plot of variation of number of flags versus $HotSpotIndex$ for an Erdos-Renyi network of 1000 vertices and edge probability of 0.15. $\alpha = 33$

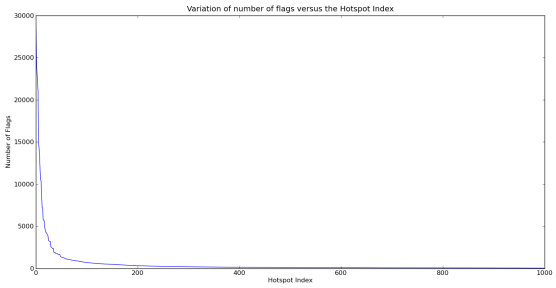

Figure 5. Plot of variation of number of flags versus $HotSpotIndex$ for a Scale Free Graph of 1000 vertices and 4 connections. $\alpha = 100$

The $HotSpotOrdering$ list contains the vertex labels arranged in the decreasing order of their number of flags. Let $HotSpotIndex(u)$ denote the position of u in $HotSpotOrdering$.
Let $HotSpotOrdering = \{h_1, h_2, h_3, ..h_{|V|}\}$.
Now, $HotSpotIndex(h_i) = i$. The plots in Figure 4 and 5 represent the variation of $flag$ values v/s $HotSpotIndex$.

Let $\alpha = |HotSpots|$. As α decreases, the navigation through the network during the navigation phase (Section. IV-B) becomes increasingly difficult since it takes a longer path to reach vertices in $HotSpots$. As α increases, the difficulty involved in computing the all pairs shortest path between the elements of $HotSpots$ using Djikstra's Algorithm increases cubically, $O(\alpha^3)$. Thus, there is a need to determine an optimal value for α.

By analyzing the plot of number of flags versus $HotSpotIndex$, we see that the rate of decrease of flagging decreases drastically at the point where the curve takes a sharp turn. Further addition of vertices into $HotSpots$ doesn't contribute significantly towards improving the efficiency of our algorithm. In fact, it causes a computational overhead during the creation of $HotSpotLookup$. Hence, we set α to a value of the curve at which it takes a sharp turn (i.e. a drastic slope change value).

B. Comparison Between Various Navigation Techniques

This section highlights the effectiveness of greedy navigation over other methods of navigation. Let $s \in V$ be the source and $t \in V$ be the target vertex. Let $d(s,t)$ denote the length of the shortest path between s and t. Here are a few methods one can adopt to accomplish the task of establishing a path between s and t:

1) 1-Way Random Walk: The idea here is to start from the source vertex s and take a random walk W_s until the target vertex t is reached. Since this technique involves a single random walk, we refer to it as a 1-way random walk. Let $\beta_{s,t}$ denote the path length of the path corresponding to the random walk W_s from s to t. Let β be the average ratio of length of 1-way random walk and length of the shortest path, taken over all unordered vertex pairs (s,t), such that $s,t \in V$.

$$\beta = \frac{1}{\binom{|V|}{2}} \sum_{s,t \in V} \frac{\beta_{s,t}}{d(s,t)}$$

2) 2-Way Random Walk: After taking random walks W_s and W_t from s and t simultaneously until the two walks intersect, one can find a path from s to t. Let $\gamma_{s,t}$ denote the length of the path thus obtained. γ is defined on similar lines with β, as follows:

$$\gamma = \frac{1}{\binom{|V|}{2}} \sum_{s,t \in V} \frac{\gamma_{s,t}}{d(s,t)}$$

3) Path Concatenation Algorithm: Let $\delta_{s,t}$ denote the length of the path from s to t obtained after executing the path concatenation algorithm. δ is also defined on similar lines with β, as folows:

$$\delta = \frac{1}{\binom{|V|}{2}} \sum_{s,t \in V(G)} \frac{\delta_{s,t}}{d(s,t)}$$

Figure. 6 is a plot of variation of β, γ and δ versus the number of vertices for an Erdos-Renyi network with an edge probability of 0.1. From the plot, it is clear that the

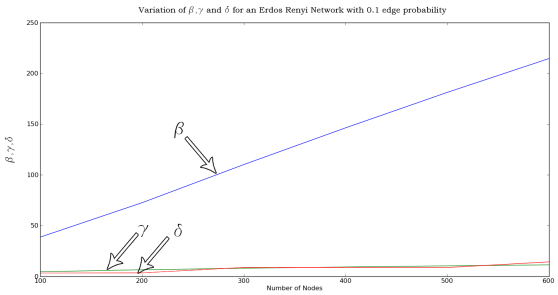

Figure 6. Plot of β, γ and δ versus the number of vertices for an Erdos-Renyi network with an edge probability of 0.1

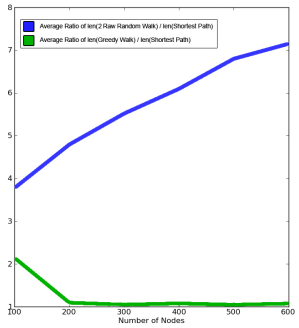

Figure 7. Plot of γ (blue curve) and δ (green curve) versus the number of vertices for a Scale-free Graph with 4 connections

δ-curve always lies below the β curve. This implies that, the greedy navigation performs better than 1-way Random Walk.

In case of Erdos-Renyi networks, the δ curve and the γ curve lie very close to each other, and also intersect at certain points in the plot. Hence, the performance of PCA is as good as a 2-way random walk technique.

From Figure 7, it is evident that in case the of scale-free graphs, the δ curve always lies below that of the γ curve. Hence, the performance of PCA is better than that of the 2-way random walk technique.

C. Hotspot Distribution for Erdos-Renyi and Scale-free graphs

The hotspot set we introduced exhibit some characteristic features. In the case of Erdos-Renyi graphs, we note that the PCA doesn't perform significantly better than the 2-way random walk. The reason being that the Erdos-Renyi graphs do not contain vertices of relatively high degree. Whereas in case of a scale-free graph, some vertices have very high degree (*hubs*) while the rest have relatively low degree. These *hubs* form valid candidates for the hotspot set. They can be reached in a few hops from any vertex in the graph, thus making the PCA an efficient strategy.

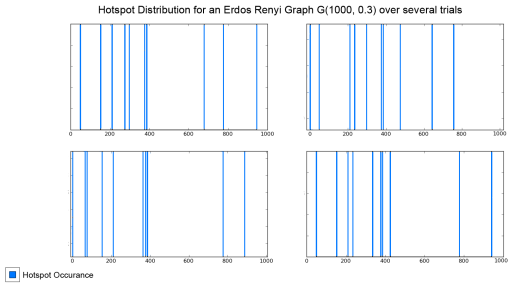

Figure 8. Hotspot Distribution for an Erdos-Renyi Graph. x-axis represents the vertex name and the vertical lines represent the hotspots

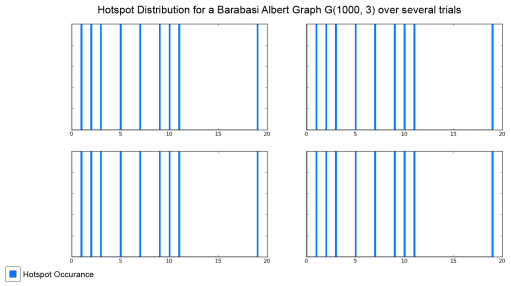

Figure 9. Hotspot Distribution for a Scale-free Graph. x-axis represents the vertex name and the vertical lines represent the hotspots

Fixing an Erdos-Renyi graph G on 1000 vertices with probability $p = 0.3$, as we run the learning phase of the PCA 4 times on the same graph, we note that the hotspot sets are different each time we run the algorithm. The top hotspot set is shown in Figure 8. E.g. if a vertex, say 306, has a vertical line, it means that the vertex is chosen as a hotspot. Repeating the same procedure on a scale-free graph, we note that the top hotspot set remains more or less the same as shown in Figure 9.

VI. Center-strategic Paths on Scale Free Networks

Given a path $(v_1, v_2, v_3, ..., v_k)$ with closeness centrality values $(c_C(v_1), c_C(v_2), ..., c_C(v_k))$ as defined in Section. III, we compute the ranking of vertices $(R(v_1), R(v_2), R(v_3), ..., R(v_k))$. By *rank-plot* of this path, we mean a plot of the ranks $(R(v_1), R(v_2), R(v_3), ..., R(v_k))$. If the rank-plot of a path has no more than one maxima, we call such a path a *center-strategic path*.

The algorithm presented in this article is inspired by the strategy adapted by humans to learn the centers of a network and navigate in a center-strategic way. The presented

algorithm resonates with the technique used by humans. We establish this fact by showing that the PCA algorithm yields center-strategic paths on scale free networks.

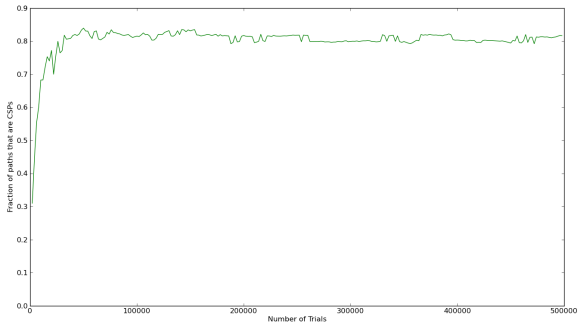

Figure 10. Center Strategic Paths Using Path Concatenation Algorithm

Consider a scale-free graph $G(V, E)$. Let k denote the number of iterations (or trials) during the navigation phase of the path concatenation algorithm. After $k = 100$, for every k, we jump to navigation phase, take ${}^{|V|}C_2$ greedy traversals (as explained in Section. IV-B) between all vertex pairs and check for the number of paths that are center-strategic. The ratio of the number of center-strategic paths to the number of greedy traversals ${}^{|V|}C_2$ is denoted by ψ.

Figure. 10 shows the variation of ψ as k increases. We note that the path concatenation algorithm yields center-strategic paths 80% of the times on scale free networks.

VII. Conclusion

Motivated by the strategy adopted by humans to navigate in an unknown environment, we presented in this paper, an algorithm which simulates human navigation. We showed that the algorithm performs better than the 1-way and 2-way random walk technique. We further showed that the proposed algorithm generates center-strategic paths on scale-free networks.

A possible further work would be to study the convergence of the learning phase of the algorithm. One can start with the basic graph structures like the paths, grids, trees and study the hotspot distribution. It would also be interesting to classify the networks on which the algorithm performs well and those on which it does not. One can study the performance of PCA algorithm on real world networks as compared to other other navigational techniques.

References

[1] L. A. Adamic, R. M. Lukose, and B. A. Huberman. Local search in unstructured networks. In *Handbook of Graphs and Networks*, pages 295–317. Wiley-VCH, 2003.

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

[3] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, January 2002.

[4] D. O. Cajueiro and R. F. S. Andrade. Learning paths in complex networks. *EPL (Europhysics Letters)*, 87(5):58004, 2009.

[5] A. Crespo and H. Garcia-molina. Routing indices for peer-to-peer systems, 2002.

[6] F. F. Dragan and M. Matamala. Navigating in a graph by aid of its spanning tree metric. 25(1):306–332, 2011.

[7] P. Erdös and A. Renyi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[8] C. Gavoille. A survey on interval routing. *Theoretical Computer Science*, 245:785–796, 1999.

[9] C. Gavoille. Routing in distributed networks: overview and open problems. *SIGACT News*, 32:36–52, March 2001.

[10] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.

[11] S. Giordano, I. Stojmenovic, and L. Blazevic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad Hoc Wireless Networking*, pages 103–136. Kluwer, 2001.

[12] Sudarshan Iyengar, Nina Zweig, Abhiram Natarajan, and Veni Madhavan. A network analysis approach to understand human-wayfinding problem. In *CogSci 2011 Proceedings*, number 33, pages 2836–2841, July 2011.

[13] J. Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.

[14] J. Kleinberg. Complex Networks and Decentralized Search Algorithms. In *In Proceedings of the International Congress of Mathematicians (ICM*, 2006.

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

[16] S. Milgram. The Small World Problem. *Psychology Today*, 1:61–67, 1967.

[17] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31, 1966.

[18] O. Simsek and D. Jensen. Navigating networks by using homophily and degree. *Proceedings of the National Academy of Sciences*, 105, 2008.