VI SEMESTER SPECIAL TOPIC

A Bidirectional Search Algorithm in Power Law Networks

By:
Vijay Mahantesh S M
-1PI09CS118
Vijesh M
-1PI09CS119

Guided By:
Dr. Kavi Mahesh
Professor
Computer Science Dept
PESIT, Bangalore

Certificate

This is to certify that Vijay Mahantesh SM (1PI09CS118) and Vijesh M (1PI09CS119) have satisfactorily completed the Special Topic Project prescribed by PES Institute of Technology (Autonomous Institute under VTU) in the year 2012 (January - May).

Signature of Guide
Dr. Kavi Mahesh

Signature of HOD
Prof. Nitin V Pujari

Acknowledgment

We wish to express our sincere gratitude to Nitin V Pujari, HOD, Computer Science Department, PESIT for providing us with an opportunity to carry out the special topic on "A Bidirectional Search Algorithm in Power Law Networks". We also wish to express our gratitude to Dr. Kavi Mahesh, Computer Science Department, PESIT for providing us with the guidance needed. We would like to thank Prof. C Pandu Rangan, IIT, Chennai, India, for providing us an opportunity to intern at IIT Chennai during June-July 2011. We would also like to thank Dr. Sudarshan Iyengar, ISI, Chennai, India, for providing us with a strong foundation of the basic concepts required and his invaluable guidance.

Abstract

Searching in networks is one of the most frequently stumbled upon requirements in network analysis. Most of the Real World networks, such as computer networks and social networks, exhibit a power law degree distribution. The high connectivity nodes play the important role of hubs in communication and networking. This fact is exploited when designing efficient local search algorithms. Lada A. Adamic introduces one such searching strategy in the paper titled "Search in Power-Law Networks". The idea proposed in the paper involves a unidirectional search technique, wherein the search is initiated from the source node and ends at the destination node. Our idea is an extension of the idea proposed in the paper. Instead of a unidirectional search, we use a bidirectional technique. We demonstrate the performance of this technique on synthetic scale-free networks and gnutella peer-to-peer networks.

Contents

1 Problem Definition

We frequently stumble upon applications that requires finding the shortest path between various nodes in a network. From ages, many algorithms have been proposed to solve this problem. One such is algorithm is Dijkstra's single source shortest path algorithm. The complexity involved in finding all pair shortest path in a network of n nodes, using this algorithm is $O(n^3)$. Some applications just require a path establishment between a given pair of nodes. There might not exist a hardbound constraint that the established path must be the shortest. In such a case, Dijkstra's algorithm creates an extra overhead of computing the exact shortest path. The technique proposed by Adamic aims to solve this problem, but with the path length tradeoff. Here we focus on finding an approximate shortest path between any two nodes by tweaking Adamic's unidirectional search algorithm.

2 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 [15], the current methods include routing using full-tables [7], interval routing [6, 7], routing labeling schemes [16, 17], greedy routing [8, 13], geographic routing [8], compass routing [8], 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 [6, 7, 8] and [16]. For surveys of the routing methods in social networks and email networks in particular, one is referred to [2, 3, 5, 13, 14]

In the classical small world experiment by Milgram [15], 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. Let us consider a variation of this problem where both sender and receiver participate in the process of finding a path between them.

2.1 Motivation

The motivation for the extension of Adamic's algorithm arises from the intuition that the network navigation can be performed in a more efficient way by considering bidirectional search. The underlying network is not visible to the walkers in its entirety. At any instance, the walkers can view the adjacent nodes of the node that it is currently present. During the analysis phase, several such source-destination node pairs are given to the walkers. The time taken and the path taken are recorded.

In the current study, we present a *navigation algorithm* that combines Adamic's navigation technique and one of the strategy used by human participants in the experiment reported in [9].

2.2 The Technique

In an unweighted graph $G(V, E)$, given a source vertex s and a destination vertex t, a straightforward 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.

This is a generic algorithm that can be applied on any type of network. Hence, it doesn't exploit the underlying structure of the network and its characteristics. Adamic's greedy walk algorithm exploits the characteristics of the power law network. We describe an extension of the algorithm that involves bidirectional search.

3 Literature Survey

Kleinberg's work [11] on navigation in a small world, was the first paper to shed light on the navigation problem on complex networks. The paper highlighted the fact that it is easier to find short chains between points in *some* networks than *others*.

Adamic et al., in [3] introduced several local search strategies to find a path to the given destination vertex. Their strategies are limited to applications on networks that obey the power law. The results are shown on gnutella peer-to-peer network which is known to obey power law. In [10], Kim et al., numerically compares the local path finding strategies with that of global path finding strategies on scale free networks.

Adamic et al., in [1] addresses the question of how participants in a small world experiment are able to find short paths in a social network using only local information about their immediate contacts. They conduct their experiment on an email network and demonstrate by empirical data that the small world search strategies using a contact's position in physical space or in an organizational hierarchy relative to the target can effectively be used to locate most individuals.

A detailed survey on decentralized search algorithms on networks that exhibit small-world phenomena is given by Kleinberg in [12]. This survey also contains an exhaustive list of open problems in this area.

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., [4] 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.

4 The Algorithm

Adamic's search technique involves the initiation of the search from the source. The walker, initially placed at the source, hops to that neighbor with the highest degree. The walker then checks to see if the destination is reached. If the walker hasnt reached the destination, a simlar action is performed from the current node. The algorithm terminates if the walker has reached the destination. A potential problem that could arise from this algorithm is the formation of cycles. According the technique, if the next hop tends to form a cycle, the neighbor with the next highest degree is chosen.

The Bidirectional Search Algorithm functions on similar lines, but with a tweak. It can be split into two parallelizable components.

- Search from the source
- Search from the destination

The logic for individual searches from either the source or the destination is same as that of Adamic's technique. At every time step, both the walkers advance one step in the manner described above. After each advancement, we check whether there is an intersection of the paths traced. When an intersection occurs, we note the time taken and the paths traversed.

In our experiments that follow, we show the results of application of our algorithm on classes of graphs such as Barabasi Albert Networks, Erdos Renyi Networks and Gnutella peer-to-peer network.

4.1 Comparison Between Various Navigation Techniques

This section highlights the effectiveness of greedy navigation over other methods of navigation. Consider two vertices $s, t \in V$. Let s be source and t be the target vertex. In the navigation phase of the algorithm, our aim is to establish a path between s and t. 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:

4.1.1 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 $s, t \in V$.

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

Figure. 1 illustrates the technique of 1-way random walk from s to t. W_s terminates when it reaches t. W_s is indicated by the red walk. The source s and target t is denoted by the red dots. The blue dots indicate the other vertices in the network.

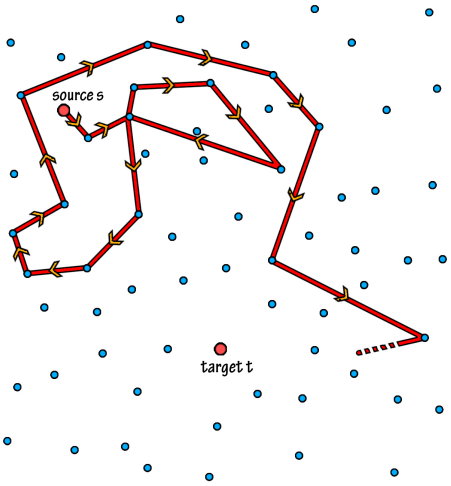

Figure 1: 1-way Random Walk

4.1.2 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.

This idea is similar to the one implemented in the learning phase in Section **??**. Let $\gamma_{s,t}$ denote the length of the path thus obtained. Let γ be the average ratio of length of 2-way Random Walk and the length of the shortest path taken over all the unordered vertex pairs (s, t), such that $s, t \in V$.

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

Figure 2 illustrates the technique of 2-way random walk between s and t. W_s and W_t are constructed simultaneously until they intersect. The intersection point of the two walks is indicated by H. W_s and W_t are indicated by the red walk and green walk respectively. Source s and the target t are denoted by the red dots.

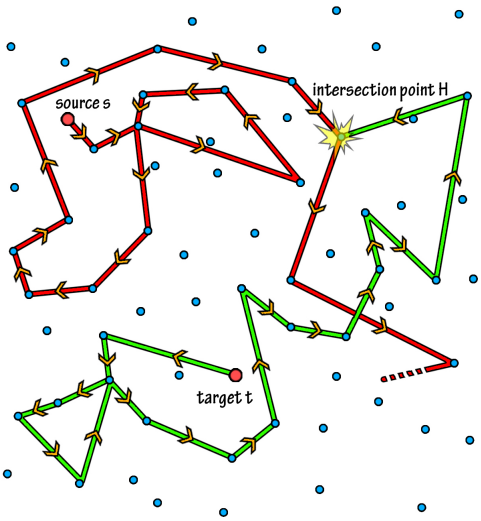

Figure 2: 2-Way Random Walk

4.2 The Measure of Performance

The measure of performance of any algorithm forms an important part of its analysis. The performance of our algorithm is measured on a relative scale. We choose the following parameters as performance measures.

- Running Time:
  The time required for an algorithm to execute forms an important factor to measure performance. In this study, we empirically compare the running time of our algorithm with that of unidirectional Adamic technique. Note that, for a given pair of nodes, we have run both the algorithms under similar computing environments. This eliminates the effects that the computing environments pose towards the performance of the algorithm.

- Average value of $\frac{actual\ path\ length}{shortest\ path\ length}$:
  The fraction $\frac{path\ length}{shortest\ path\ length}$ denotes the factor by which the path length obtained by our algorithm is greater than the actual shortest path length. We consider the average of this value over a large number of randomly chosen node pairs.

- Performance factor $\frac{1}{time * average \frac{path\ length}{shortest\ path\ length}}$:
  The performance of the algorithm is said to be high if it produces a shorter path and takes a relatively lesser amount of time. Assuming that the performance varies inversely as the parameters mentioned above, we empirically consider this fraction as an effective performance measure.

4.3 Subtle issues involving constraints and decision logic

The implementation of an algorithm reveals its subtleties. Many issues / design decisions come into light only when we implement the algorithms. In this section, we present the subtleties and optimizations that has been brought about during the implementation of the algorithm.

1. Parallelization:
   A parallel algorithm is an algorithm that can be executed one piece at a time on many different processing devices. The results from individual processing threads are then combined back into one single result. All the algorithms cannot be divided into parallel pieces. It is the inherent nature of the algorithm which makes it parallelizable.

   The core step of our algorithm is the advancements that the walkers make at each time step. Since either of the walkers do not influence the hop decision of the other walker, our algorithm can be *theoritically* parallelized. Although this may sound theoretically feasible, the practical implementation poses a variety of problems. One such example is presented below.

   Let A and B be two threads running in parallel. Note that the threads are running in parallel, not simultaneously. The interleaving period for each thread is sufficient for one of the walkers to race past the other. Ideally, each walker must take turns in advancing a single step. This leads to a race condition. Consider a snippet of the actual scenario shown below.
   $Path_A : [\prime 24\prime, \prime 3\prime, \prime 23\prime]$

$Path_B['26','4','23','12','7','12','9','4','18','3']$
Intersection Point: '3'
This lack of synchronization leads to the wrong choice of the intersection point, thereby increasing the effective path length.

2. Formation of Cycles
On an abstract level, unidirectional Adamic walk suggests that the next hop should be made to the neighbor with the highest degree. Trivially, the idea seems to be functional in all the cases. Further analysis reveals that such a step leads to the formation of cycles. If a walker hops to the richest neighbor and causes a cycle formation, it can never exit the cycle. This causes the walker to stay on the cycle for eternity. To avoid such a scenario, Adamic suggested that the walker should hop on to the neighbor with the next highest degree in case a cycle formation is detected in prior.

   In this design decision, our algorithm differs from that of Adamic's. The computational cost (time) involved in deducing the neighbor with the next highest degree is relatively high. Hence, we make a random choice for next hop.

   This solution works with a very high probability. But there are situations wherein a hop to any of the neighbors would lead to the formation of a cycle. In other words, the walker would have already visited all the neighbors. In such a case, we suggest to take 2 random hops. Ofcourse, this causes a cycle, but the difference is that the walker is not placed within the cycle. This allows the walker to continue its quest for the intersection point.

3. Performance Optimization - Flagging technique instead of Linear Search
A trivial algorithm for identifying the formation of cycles in the network is to search whether the next node is already in the path traversed by the walker. This requires a linear traversal through the list that represents the path. The formation of cycles must be checked before every hop. This leads to a high degree of complexity. A more efficient way to handle this scenario is to flag the nodes that have already been visited. Before every hop, the flag is checked. If the flag is set, it indicates that the node has been visited, thus leading to the formation of a cycle. If the flag is not set, the hop is made and the flag is set.

5 Application of our algorithm to various classes of networks

In the subsections below, we present and discuss four results for Erdos-Renyi Graphs and Barabasi-Albert Network. The first figure depicts a comparison of the average $\frac{lengthofactualpath}{lengthofshortestpath}$ between two-way random walk and two-way adamic walk, for varying number of nodes. The second figure compares the relative time taken by one-way adamic walk and two-way adamic walk, for varying number of nodes. The third figure compares the average $\frac{lengthofactualpath}{lengthofshortestpath}$ for one-way random walk, two-way random walk, one-way adamic walk and two-way adamic walk. The fourth figure depicts the variation of the performance measure that we had defined earlier - $\frac{1}{time*average\frac{path\ length}{shortest\ path\ length}}$ for all the four methods, with varying number of nodes.

5.1 Application of the algorithm on Erdos-Renyi Graphs

In this section, we present the results obtained when our algorithm was applied to a class of networks known as Erdos Renyi networks. An Erdos Renyi network G(n,p) is a random graph obtained by starting with n vertices and adding edges between them at random with a probability p. The experiments were conducted for varying number of nodes in a graph, keeping the attachment probability p as 0.3. The following figures show the results obtained from our experiments.

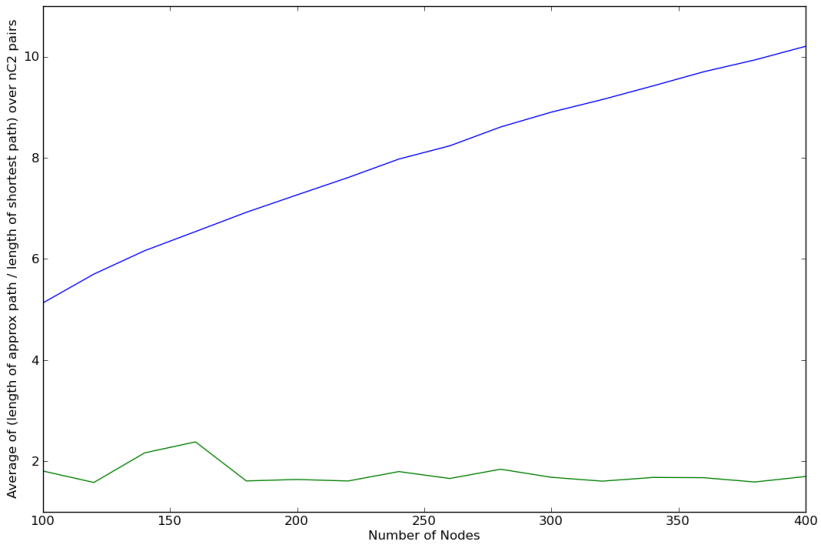

Figure 3: average $\frac{lengthofactualpath}{lengthofshortestpath}$ for Erdos Renyi Networks

This plot shows the variation of average $\frac{lengthofactualpath}{lengthofshortestpath}$ for Two-way Random walk and Two-way Adamic walk. The algorithm has been tested on Erdos Renyi graphs with 0.3 probability and the number of nodes varying from between 100 to 400. The length of the path obtained Two-way adamic walk is around 1.75 times the shortest path length, on an average. As the number of nodes increase, the average value of $\frac{lengthofactualpath}{lengthofshortestpath}$ increases for the Two-way Random walk technique. Our proposed method is almost an invariant with respect to the number of nodes.

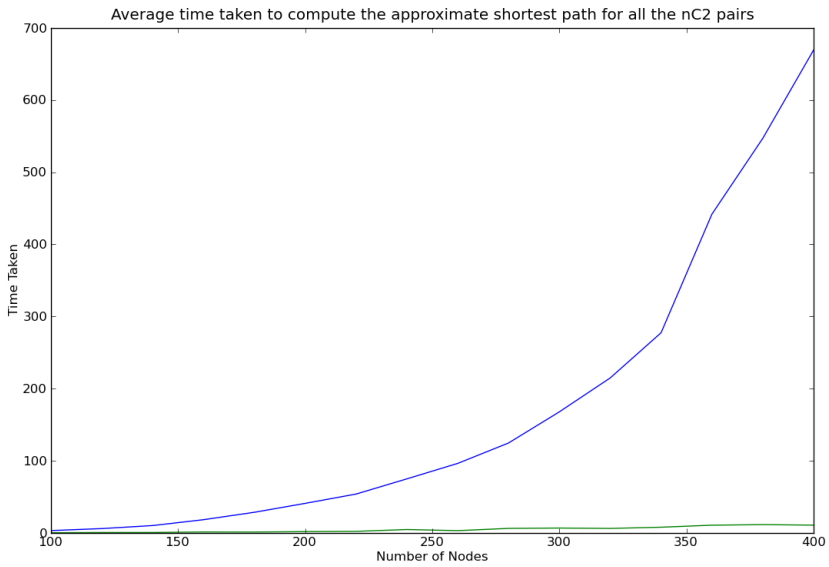

Figure 4: Time Taken

This plot shows the variation of the average time taken to establish connection between arbitrary pair of nodes in an Erdos Renyi network. This figure depicts the contrast between the connection establishment time using one-way adamic walk and two-way adamic walk. As seen from the figure, the connection establishment time exponentially increases in the case of one-way Adamic walk.

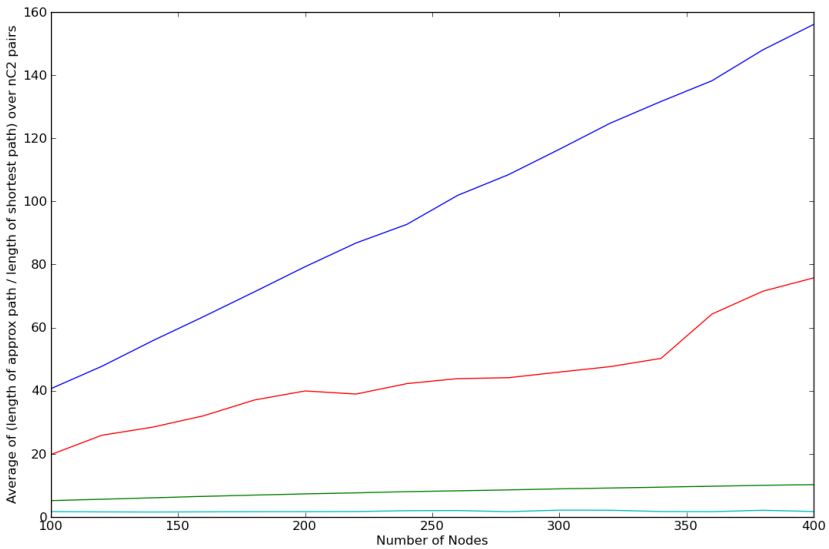

Figure 5: average $\frac{lengthofactualpath}{lengthofshortestpath}$ for Erdos Renyi Graph with 0.3 probability

This plot shows the variation of average $\frac{lengthofactualpath}{lengthofshortestpath}$ for One-way Random walk, Two-way Random walk One-way Adamic walk and Two-way Adamic walk.

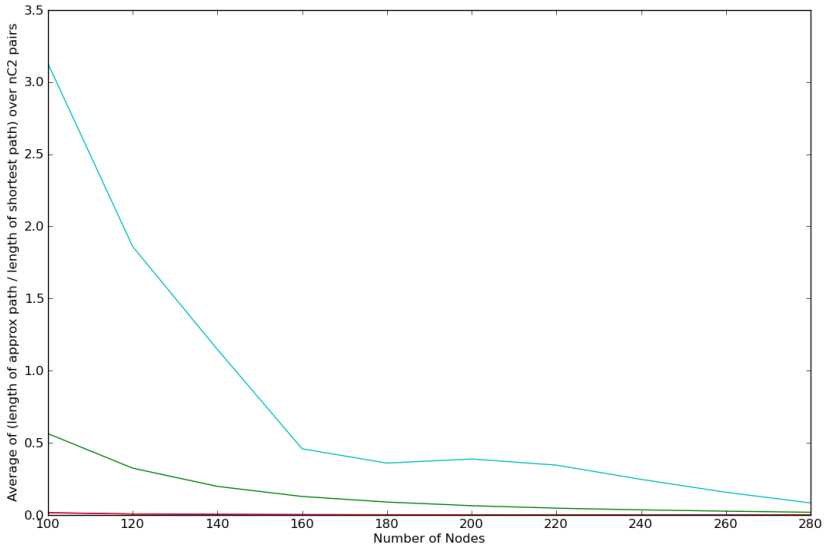

Figure 6: average $\frac{lengthofactualpath}{lengthofshortestpath}$ for Erdos Renyi Networks

The performance measure that we introduced earlier is plotted against the varying number of nodes for an Erdos Renyi Network. As expected, the performance of the two-way adamic walk better than the other methods. But the performance factor decreases as number of nodes increases. This is probably due to the fact that the formula was chosen empirically.

5.2 Application of the algorithm on Barabasi Albert Networks

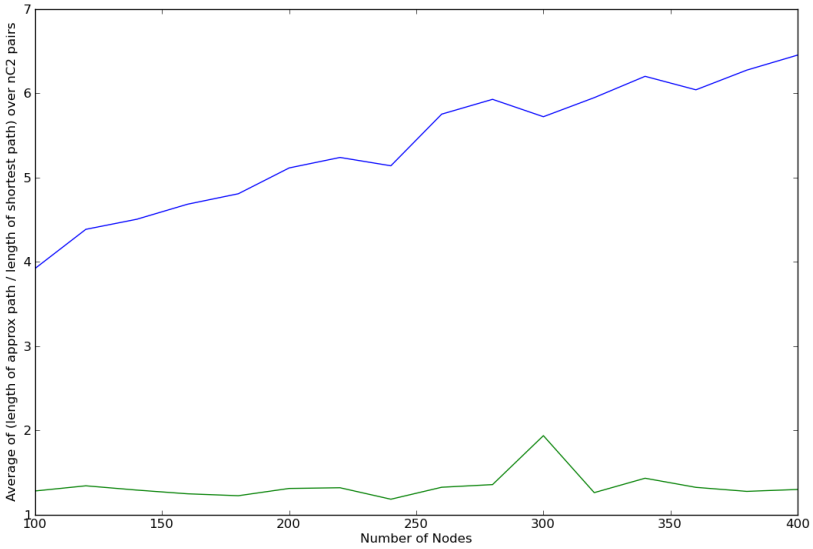

Figure 7: average $\frac{lengthofactualpath}{lengthofshortestpath}$ for Barabasi Albert Network of degree 3

This plot shows the variation of average $\frac{lengthofactualpath}{lengthofshortestpath}$ for Two-way Random walk and Two-way Adamic walk. The algorithm has been tested on Barabasi Albert Networks with degree 3 and the number of nodes varying from between 100 to 400.

The length of the path obtained Two-way adamic walk is around 1.25 times the shortest path length, on an average. Clearly, The average in the case of Power Law Networks is smaller than that of Erdos Renyi Networks. As the number of nodes increase, the average value of $\frac{lengthofactualpath}{lengthofshortestpath}$ increases for the Two-way Random walk technique. Our proposed method is almost an invariant with respect to the number of nodes.

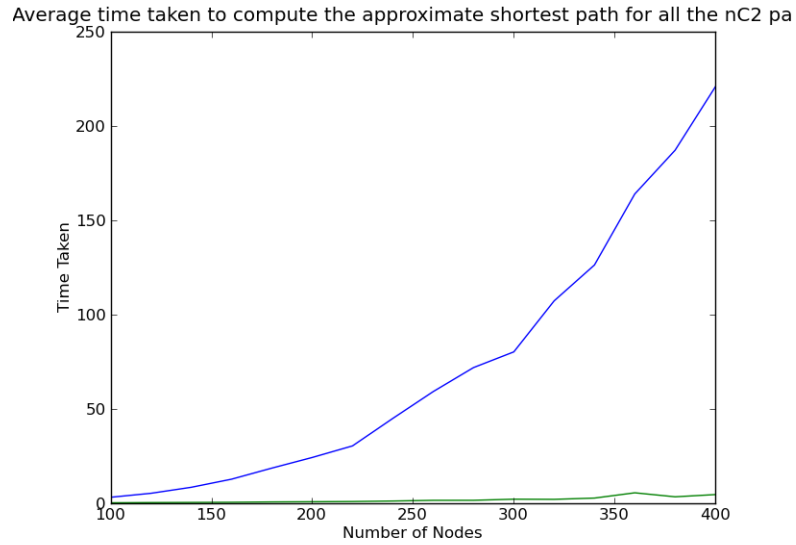

Figure 8: Time Taken

This plot shows the variation of the average time taken to establish connection between arbitrary pair of nodes in a Barabasi Albert network. This figure depicts the contrast between the connection establishment time using one-way adamic walk and two-way adamic walk. As seen from the figure, the connection establishment time exponentially increases in the case of one-way Adamic walk.

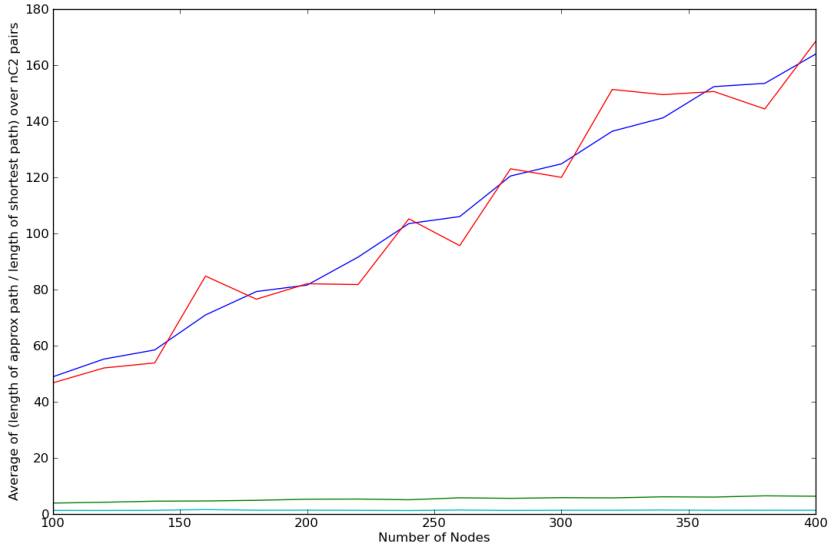

Figure 9: average $\frac{lengthofactualpath}{lengthofshortestpath}$ for Barabasi Albert Network of degree 3

This plot shows the variation of average $\frac{lengthofactualpath}{lengthofshortestpath}$ for One-way Random

walk, Two-way Random walk One-way Adamic walk and Two-way Adamic walk. The performance of One-way Random walk and One-way Adamic walks are comparable. The performance of One-way Random walk is as expected. The One-way adamic walk fails to perform well because the nodes in the periphery are hard to reach using Adamic's technique. Hence, it is comparable to a One-way Random walk.

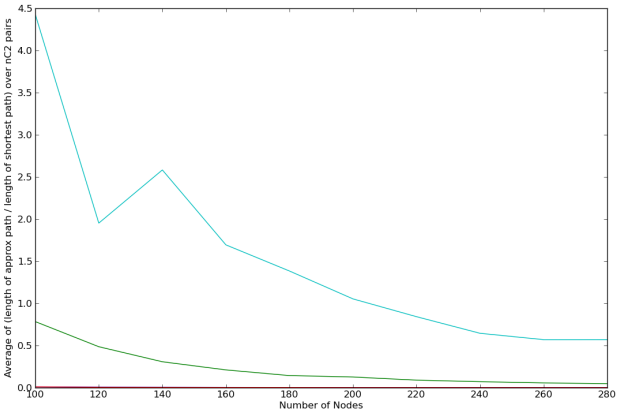

Figure 10: average $\frac{lengthofactualpath}{lengthofshortestpath}$ for Barabasi Albert Network of degree 3

The performance measure that we introduced earlier is plotted against the varying number of nodes for a Barabasi Albert Network. As expected, the performance of the two-way adamic walk better than the other methods. But the performance factor decreases as number of nodes increases. This is probably due to the fact that the formula was chosen empirically.

5.3 Application of the algorithm on Gnutella network

Gnutella is a system in which individuals can exchange files over the Internet directly without going through a Web site. This is achieved using a peer-to-peer arrangement. In our experiments, we consider the gnutella network provided at http://snap.stanford.edu/data/p2p-Gnutella08.html. A sequence of snapshots of the Gnutella peer-to-peer file sharing network had been sampled during August 2002. There are total of 9 snapshots of Gnutella network collected in August 2002. Nodes represent hosts in the Gnutella network topology and edges represent connections between the Gnutella hosts. We consider the sample taken on 8th August 2002. The degree distribution of the considered gnutella network is shown in the figure 11. Clearly, the network exhibits Scale Free behavior.

We make 2 important assumptions for our algorithm to be applied for the dataset. First, we convert the directed edges of the network into undirected edges to indicate the bidirectional data transfer. Second, we consider the largest connected subgraph from the given dataset. We observe that two of the nodes are disconnected in the given dataset. Hence, we consider an undirected connected subgraph of the gnutella network consisting of 6299 nodes and 20776 edges. For each experiment, 5000 node pairs are picked at random and the algorithms are applied for comparison. The following figures depict the results obtained.

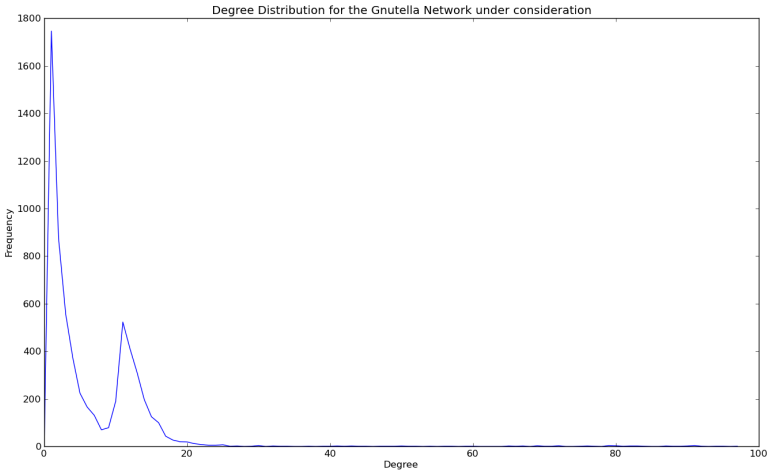

Figure 11: Scale Free Nature of the considered gnutella network

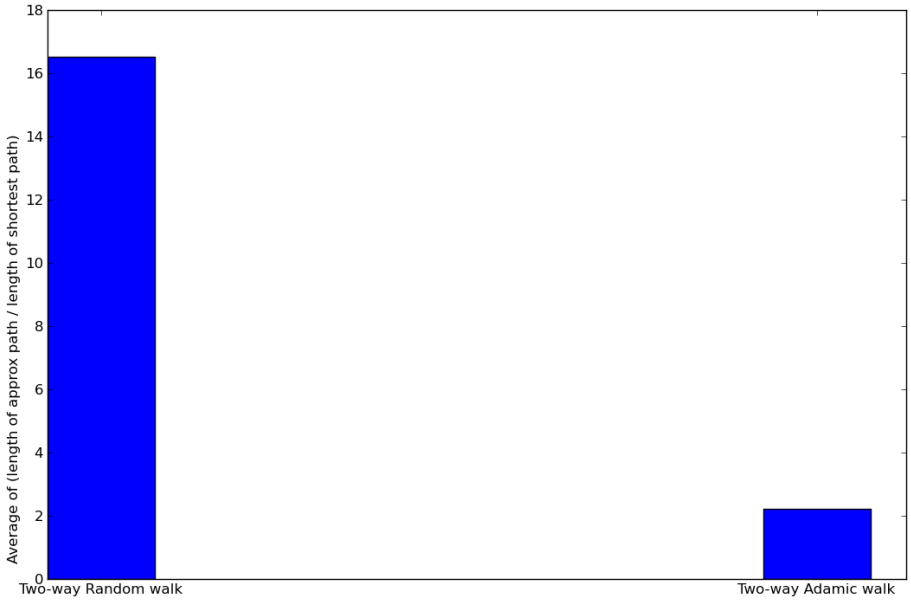

Figure 12: average $\frac{lengthofactualpath}{lengthofshortestpath}$ for a gnutella network consisting of 6299 nodes and 20776 edges

The above figure shows the average time required to establish connection between a randomly chosen pair of nodes in the considered gnutella network. Clearly, the average ratio of $\frac{lengthofactualpath}{lengthofshortestpath}$ using two-way random walk technique exceeds the average ratio using two-way Adamic walk by a large factor. The Two-way random technique yields a ratio of approximately 16, while the Two-way Adamic walk yields a ratio of approximately 2. This shows that the connection can be established in the gnutella network using Two-way Adamic walk is much shorter, relative to the path obtained using Two-way Random walk.

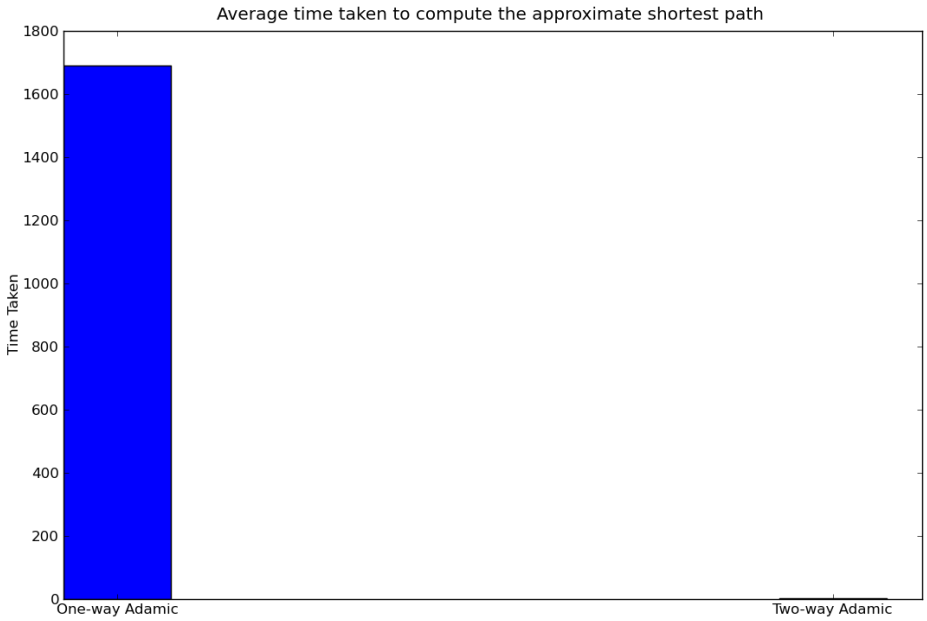

Figure 13: average $\frac{lengthofactualpath}{lengthofshortestpath}$ for a gnutella network consisting of 6299 nodes and 20776 edges

The above figure shows the average time required to establish connection between a randomly chosen pair of nodes in the considered gnutella network. Clearly, the average time required to establish connection using One-way Adamic walk is far larger than the time taken using Two-way Adamic walk. The ratio of the time intervals taken to perform One-way Adamic walk and Two-way Adamic walk is 524.062. Hence, Two-way Adamic walk is faster than One-way Adamic walk by a factor of 0.001908169, on the considered gnutella network.

6 Implementation

- The above algorithm has been implemented using Python 2.7.
- Python Development Kits used :
  - NetworkX
  - Matplotlib
  - Numpy
  - Scipy
  - Pickle
  - Random
- yEd network visualization tool from yWorks.
  http://www.yworks.com/en/products_yed_download.html

7 Conclusion

Motivated by the strategy adopted by humans to navigate in an unknown environment and the One-way Adamic graph navigation technique, we presented in this study, an extension to the algorithm proposed by Adamic. We showed that the algorithm performs better than the One-way Random walk, Two-way Random walk and One-way Adamic walk techniques on Power Law Networks.

A possible further work would be to formulate the thoeritical complexity and bounds of the algorithm. It would be interesting to compare our algorithm with the complexities of other navigational techniques. It would also be interesting to classify the networks on which the algorithm performs well and those on which it does not.

References

[1] L. A. Adamic and E. Adar. How to search a social network. *Social Networks*, 27:2005, 2005.

[2] 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.

[3] 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.

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

[5] P. Fraigniaud. Small worlds as navigable augmented networks: model, analysis, and validation. In *Proceedings of the 15th annual European conference on Algorithms*, ESA'07, pages 2–11, Berlin, Heidelberg, 2007. Springer-Verlag.

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

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

[8] 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.

[9] 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.

[10] B. J. Kim, C. N. Yoon, S. K. Han, and H. Jeong. Path finding strategies in scale-free networks. *Phys Rev E Stat Nonlin Soft Matter Phys*, 65(2 Pt 2), February 2002.

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

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

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

[14] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11623–11628, August 2005.

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

[16] D. Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[17] M. Thorup and U. Zwick. Compact routing schemes. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, SPAA '01, pages 1–10, New York, NY, USA, 2001. ACM.