Relating centrality of graph nodes to searchability

Peter Horváth

January 2016

1 Introduction

In this project I focus on different types of nodes (depending on centrality) and study whether centrality affects searchability. In other words, if central (or important) nodes are easier to search in networks. By the term searchability it's possible to imagine the fact, that there exist underlying distributed way to route messages through the network. I am experimenting with Kleinberg's greedy myopic search algorithm in the networks which simulate aspects of real-world networks with goal to find out if this algorithm is more likely to find shortest paths between important nodes or between not important nodes (or between a mixture of nodes).

2 Tasks

The main task (find effect of centrality to searchability) was divided into smaller tasks, each focusing on different aspect and using different algorithms.

2.1 Task 1

Create a graph based on Barabasi-Albert model. Use Fruchterman-Reingold layout of igraph library to embed nodes into geographical space. Based on closeness centrality select the most important nodes of the graph (first quarter of the nodes). Then:

1. Find path with myopic search from each to each node of the most important group (first quarter). Find out how many times the shortest path was found (find real shortest paths with Dijkstra algorithm).
2. Find path with myopic search from each node of the most important group to each node of the rest of the graph. Find out how many times the shortest path was found.
3. Find path with myopic search from each to each node of the rest of the graph (three quarters). Find out how many times the shortest path was found.

2.2 Task 2

The same as Task 1, but now use Reingold-Tilford and circle layout.

2.3 Task 3

The same as Task 1 but now select the most important nodes based on eigenvector centrality.

2.4 Task 4

The same as Task 1 but now with graph based on Forest Fire model.

2.5 Task 5

Study the results of the experiments and discuss whether the centrality is important in terms of searchability.

3 Results

In this section I provide all the tables which shows the data from all the experiments. In the section Methods you can find descriptions of used types of centralities and layout algorithms mentioned in this section.

The properties of the graphs used in experiment are in table 1 where:

- N is number of nodes
- E is number of edges
- $< k >$ is obtained with: $k = \frac{E}{N}$
- The network density δ is calculated with: $d = \frac{2E}{N(N+1)}$
- C is clustering coefficient of the network

Graph Model	N	E	$< k >$	δ	C
Barabasi-Albert	1000	2994	2.994	0.00598	0.010998
Barabasi-Albert	500	1494	2.988	0.01193	0.031523
Forest-Fire	1000	2106	2.106	0.00421	0.118642
Forest-Fire	300	664	2.213	0.01471	0.158986

Table 1: Graphs based on Barabasi-Albert model

First I tested random graph based on Barabasi-Albert model. This model is scale-free and simulates different aspects from real networks. I select one quarter of nodes which are the most important based on their centrality and then with myopic search I am looking for a path from each to each node of this

group. Then I check if the shortest path was found. For this experiment I use the first graph of table 1 which has 1000 nodes. The result of experiment is visible in the table 2. Legend of the table:

- centrality - centrality algorithm used for computing the importance of the nodes
- layout - layout algorithm used
- N-imp - amount of most important nodes of the graph
- searches - amount of searches executed
- same-path - how many times the myopic search path was the same as dijkstra search path
- different-path - how many times the myopic search path was different (longer) than dijkstra search path
- p-sp - percentual rate of same-path according to number of searches
- p-dp - percentual rate of different-path according to number of searches

centrality	layout	N-imp	searches	same-path	different-path	p-sp	p-dp
closeness	Fruchterman-Reingold	250	31125	4525	26600	14.54%	85.46%
closeness	Circle	250	31125	2934	28191	9.43%	90.57%
closeness	Reingold-Tilford	250	31125	12854	18271	41.3%	58.7%
eigenvector	Fruchterman-Reingold	250	31125	6321	24804	20.31%	79.69%
eigenvector	Circle	250	31125	3657	27468	11.75%	88.25%
eigenvector	Reingold-Tilford	250	31125	11311	19814	36.34%	63.66%

Table 2: Experiment on Barabasi-Albert model graph (1000 nodes) - search among most important nodes

In the second experiment I select one quarter of the most important nodes again but this time I use myopic search to get a path from each node of the most important group to each node of the rest of the graph (other three quarters). Then I check if the shortest path was found. For this experiment I use the second graph of table 1 which has 500 nodes, because of limited computation power of my laptop. The result of experiment is visible in the table 3. Legend of the table is the same as of table 2, with one more column N-others, which is the amount of nodes of the rest of the graph (other three quarters).

In the third experiment I am using myopic search to find paths among all the nodes in the group of not important nodes (three quarters). This experiment uses second graph of table 1 which has 500 nodes. The result of experiment is visible in the table 4.

The tables 5 6 7 are of the same format as previous, but this time using graph based on model Forest-Fire

centrality	layout	N-imp	N-others	searches	same-path	different-path	p-sp	p-dp
closeness	Fruchterman-Reingold	125	375	46875	7614	39261	16.24%	83.76%
closeness	Circle	125	375	46875	4054	42821	8.65%	91.35%
closeness	Reingold-Tilford	125	375	46875	16753	30122	35.74%	64.26%
eigenvector	Fruchterman-Reingold	125	375	46875	9353	37522	19.95%	80.05%
eigenvector	Circle	125	375	46875	6639	40236	14.16%	85.84%
eigenvector	Reingold-Tilford	125	375	46875	19776	27099	42.19%	57.81%

Table 3: Experiment on Barabasi-Albert model graph (500 nodes) - mixed search

centrality	layout	N-others	searches	same-path	different-path	p-sp	p-dp
closeness	Fruchterman-Reingold	375	70125	7980	62145	11.38%	88.62%
closeness	Circle	375	70125	4286	65839	6.11%	93.89%
closeness	Reingold-Tilford	375	70125	21661	48464	30.89%	69.11%
eigenvector	Fruchterman-Reingold	375	70125	8306	61819	11.84%	88.16%
eigenvector	Circle	375	70125	6360	63765	9.07%	90.93%
eigenvector	Reingold-Tilford	375	70125	17839	52286	25.44%	74.56%

Table 4: Experiment on Barabasi-Albert model graph (500 nodes) - search among not important nodes

centrality	layout	N-imp	searches	same-path	different-path	p-sp	p-dp
closeness	Fruchterman-Reingold	250	31125	15237	15888	48.95%	51.05%
eigenvector	Fruchterman-Reingold	250	31125	18237	12888	58.59%	41.41%

Table 5: Experiment on Forest-fire model graph (1000 nodes) - search among most important nodes

centrality	layout	N-imp	N-others	searches	same-path	different-path	p-sp	p-dp
closeness	Fruchterman-Reingold	75	225	16875	6034	10841	35.76%	64.24%
eigenvector	Fruchterman-Reingold	75	225	16875	4542	12333	26.92%	73.08%

Table 6: Experiment on Forest-fire model graph (300 nodes) - mixed search

centrality	layout	N-others	searches	same-path	different-path	p-sp	p-dp
closeness	Fruchterman-Reingold	375	25200	5271	19929	20.92%	79.08%
eigenvector	Fruchterman-Reingold	375	25200	6477	18723	25.7%	74.3%

Table 7: Experiment on Forest-fire model graph (300 nodes) - search among not important nodes

4 Discussion

In the tables introduced in section Results it's clearly visible that centrality of nodes really can affect their searchability. If you see the results of the ta-

ble 2 you can notice that if the myopic search is used only between important nodes (based on closness centrality), the shortest path is found around 14% times with Fruchterman-Reingold layout and around 9% times with circle layout. With Reingold-Tilford it's even more - 43%. In this table we can also see that using eigenvector centrality gives us even better values (except one).

The table 3 tells us that when we use myopic search from important nodes to the rest of the graph, the values are quite similar. Again the eigenvector centrality gives us shortest path more times.
When you see the table 4 where we use myopic search only among not important nodes, the shortest path is found much fewer times than in previous 2 tables. This is a proof of that centrality of the nodes effects searchability. When important node is involved as an initial node of the myopic search, the shortest path is found more times. Also using eigenvector centrality affects this more in a positive way.

5 Methods

In this section I introduce and explain methods and algorithms used in this project.

5.1 Layout

When a graph traversal is supposed to be "move as close to target as possible", nodes of the graph have to be embedded into geographic space, so it's possible to measure the distance among them. To accomplish this goal a layout algorithm is applied on the graph. Igraph library implements many layout algorithms. The result of applying layout algorithm are Cartesian coordinates which I could assign to each node and work in this 2D geographic space. As a distance between two nodes I use Euclidean distance which computes the distance in Euclidean n-space (in this case its Euclidean 2-space). The formula is:

$$d(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

I use three types of 3 types of layout for this project:

- Circle layout - places the vertices of the graph uniformly on a circle or a sphere
- Reingold-Tilford layout - a tree layout. If the given graph is not a tree, a breadth-first search is executed first to obtain a possible spanning tree
- Fruchterman-Reingold - places the vertices of the graph on a plane according to Fruchterman-Reingold algorithm, which is a force directed layout

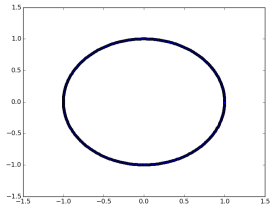

Figure 1: Circle layout example

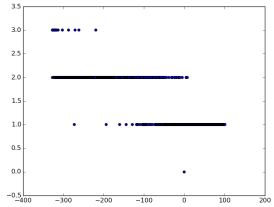

Figure 2: Reingold-Tilford layout example

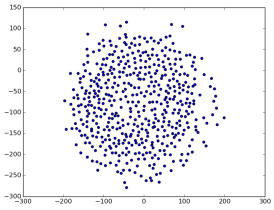

Figure 3: Fruchterman-Reingold layout example

5.2 Centrality

I assume two types of centrality:

- Closeness centrality - the more a node is close to all the other nodes the more important it is. Important nodes are easily reachable and also have power to quickly reach others
- Eigenvector centrality - centrality of a node is proportional to the sum of scores of its neighbors

5.3 Myopic Search

The core of the project is to find out if more important nodes (based on centrality) are easier to find than less important nodes using decentralized search

algorithm. For this purpose I implemented myopic search, which is a greedy algorithm. Each node knows it's location and the location of the target. Each node also know it's neighbors (local information). Node forwards the "message" greedily - moving as close to target as possible.
My implementation of myopic search (from source to target) works as follows:

1. Prepare an empty list for the path
2. If the target is direct neighbor of the source, return the target
3. If the target is not direct neighbor, mark source as temporary target and prepare empty list where ids of visited nodes will be stored
4. While the temporary target is not equal to the target take the first neighbour of the temporary target which is not in list of visited nodes and mark him as closest to the target. Now go through all of the other neighbors of temporary target and compare if Euclidean distance of actual neighbor to the target is lower than Euclidean distance of actual closest node to the target. If so, mark actual neighbor as closest to the target. Finally, after checking all the neighbors, mark the closest-to-target node as a new temporary target. There is possibility that temporary target has all the neighbors in the list of visited nodes, in this case it's possible to continue to the node even if it's in the list of visited nodes.
5. Add temporary target to list of visited nodes and to the path list. The while loop in point 4. will continue with next iteration.