Efficient Measuring of Node Centrality by Detecting Bridges and Communities.

Sol Kim
Farragut High School

ABSTRACT

Finding node centrality has significant applications to the real world. For example, node centrality can be used to determine a person that has the potential to spread a disease during a pandemic. However, modeling an entire city or nation and finding the most influential people in terms of disease spread would be difficult because of the increasing time complexity. There is a way to remedy this, and that is through the use of bridge and community detection. These two techniques can be used to analyze a graph by splitting it into more digestible pieces called subgraphs. From there, the node centralities of these more manageable subgraphs can be found and used to determine which nodes need to be studied. In addition to these methods, different algorithms to calculate centrality were analyzed for evaluation, which proved the efficiency of this model in regards to time complexity.

I. Introduction

During an epidemic, the main objective is simple: to study and neutralize the widespread disease. However, achieving this is much easier said than done. Controlling the actions of every person in a society is near impossible, making it difficult to simply wait the epidemic out. This is where graph theory comes into play. The graph as a whole would describe a social network, some community that is being affected by the epidemic[6],[7]. Each node can describe a person, while each edge would represent the transmission of the disease. From there, the node, or person in this case, with the most "influence" can be isolated and then studied to gain a better understanding of the disease. In order to determine which person has the most influence, a node centrality algorithm can be applied. Although some of the algorithms may capture the characteristics of an epidemic more so than others, time complexity is of concern as well. What is desired is a satisfactory medium between accuracy and efficiency. Determining the most central node can be broken up into the following process:

First, bridges in the graph will be determined in order to divide the graph into several subgraphs. Afterwards, each subgraph will be separated into communities, which is useful for further isolation of a target. Finally, the most central node(s) can be found from these communities using node centrality metrics such as degree, betweenness, closeness, eigenvector, and Katz centrality. These centralities would then be normalized and

compared among communities to discern which nodes are of the highest priority.

The design of the paper is as follows: Section II provides the definitions of different node centrality metrics and how to calculate them, Section III and IV discuss the how bridges and communities in a graph are detected, Section V recounts the time complexity of this model and the methods by which node centrality are calculated, and Section VI concludes the paper and elaborates on the implications of the results.

II. Measuring Node centrality

Node centrality in complex networks implies that which nodes are the most influential[1],[2] and important [10],[11] in the given graph. [1],[5],[11] showed how the node centrality works in practice and suggested some basic methods those can be helpful finding meaningful nodes. In this section, we introduce various methods: Degree centrality, Closeness Centrality[3], Betweenness Centrality[4], and eigenvector based methods. We explained each method with examples and compared each method at the end of the section.

A. Degree Centrality

As the name suggests, the measurement used for degree centrality is the degree of a node, or how many edges are connected to it. The reasoning behind measuring centrality this way is rather intuitive: because there are many edges and, therefore, nodes that are connected to a specific node, perhaps that node can be viewed as having the most influence. The centrality of the ith vertex v_i is denoted by $C_d(v_i)$.

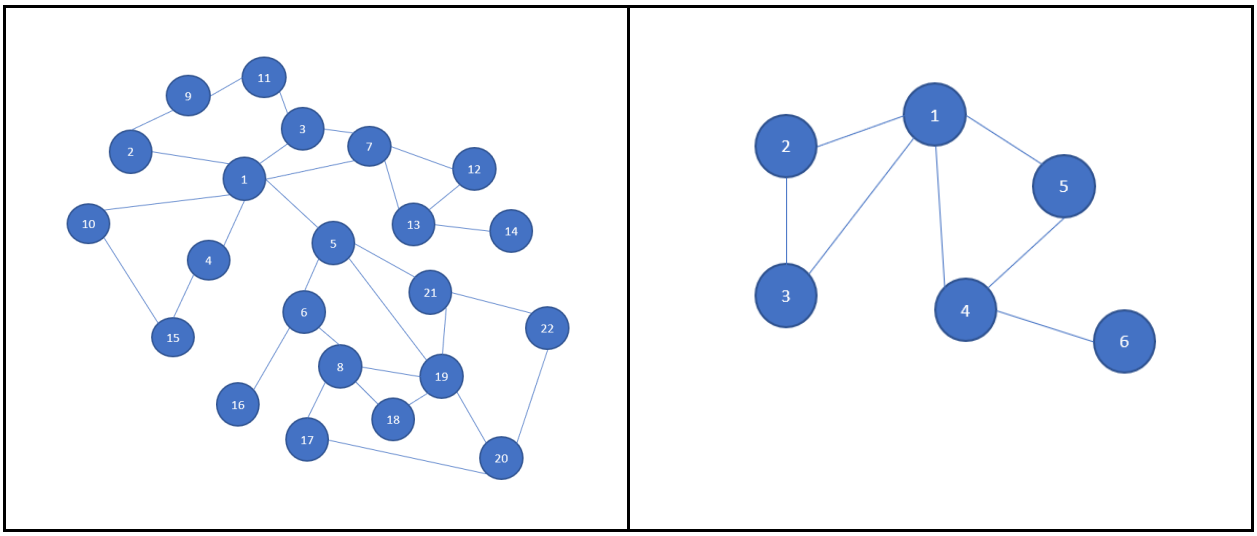
$$C_d(v_i) = d_i \qquad \text{Equation 1}$$

$$\text{Normalization: } \frac{d_i}{n} \qquad \text{Equation 2}$$

$$\text{Normalization: } \frac{d_i}{max(d_i)} \qquad \text{Equation 3}$$

d_i denotes the degree of ith vertex, and $max(d_i)$ denotes the greatest degree out of the n vertices. The purpose of normalization is to be able to compare graphs with each other. For example, if one node on a graph has 100 edges and another node on another graph has only 5, then it would seem like the first node has much more influence. However, if these were normalized, then it is possible that the centrality of the first node is equal or even less than that of the second, making it have just as much if not less influence relative to its own graph.

Figure 1	Figure 2



Example 1:

Degree Centralities of Figure 1:

- $C_d(v_1) = 6$
- $C_d(v_2) = 2$
- $C_d(v_3) = 3$
- $C_d(v_4) = 2$
- $C_d(v_5) = 4$
- $C_d(v_6) = 3$
- $C_d(v_7) = 4$
- $C_d(v_8) = 4$
- $C_d(v_9) = 2$
- $C_d(v_{10}) = 2$
- $C_d(v_{11}) = 2$
- $C_d(v_{12}) = 2$
- $C_d(v_{13}) = 3$
- $C_d(v_{14}) = 1$
- $C_d(v_{15}) = 2$
- $C_d(v_{16}) = 1$
- $C_d(v_{17}) = 2$
- $C_d(v_{18}) = 1$
- $C_d(v_{19}) = 5$
- $C_d(v_{20}) = 3$
- $C_d(v_{21}) = 3$
- $C_d(v_{22}) = 2$

From this, it can clearly be seen that v_1 is the most central node in Figure 1. If these were to be normalized using Equation 2, then each of these would be divided by 22:

- $C_d(v_1) = \frac{3}{11}$
- $C_d(v_2) = \frac{1}{11}$
- $C_d(v_3) = \frac{3}{22}$
- $C_d(v_4) = \frac{1}{11}$
- $C_d(v_5) = \frac{2}{11}$
- $C_d(v_6) = \frac{3}{22}$
- $C_d(v_7) = \frac{2}{11}$
- $C_d(v_8) = \frac{2}{11}$
- $C_d(v_9) = \frac{1}{11}$
- $C_d(v_{10}) = \frac{1}{11}$
- $C_d(v_{11}) = \frac{1}{11}$
- $C_d(v_{12}) = \frac{1}{22}$
- $C_d(v_{13}) = \frac{3}{22}$
- $C_d(v_{14}) = \frac{1}{22}$
- $C_d(v_{15}) = \frac{1}{11}$
- $C_d(v_{16}) = \frac{1}{22}$
- $C_d(v_{17}) = \frac{1}{11}$
- $C_d(v_{18}) = \frac{1}{22}$
- $C_d(v_{19}) = \frac{5}{22}$
- $C_d(v_{20}) = \frac{3}{22}$
- $C_d(v_{21}) = \frac{3}{22}$

- $C_d(v_{11}) = \frac{1}{11}$
- $C_d(v_{22}) = \frac{1}{11}$

And similarly, if they were to be normalized using Equation 3, or dividing by the greatest degree among d_i:

- $C_d(v_1) = 1$
- $C_d(v_{12}) = \frac{1}{3}$
- $C_d(v_2) = \frac{1}{3}$
- $C_d(v_{13}) = \frac{1}{2}$
- $C_d(v_3) = \frac{1}{2}$
- $C_d(v_{14}) = \frac{1}{6}$
- $C_d(v_4) = \frac{1}{3}$
- $C_d(v_{15}) = \frac{1}{3}$
- $C_d(v_5) = \frac{2}{3}$
- $C_d(v_{16}) = \frac{1}{6}$
- $C_d(v_6) = \frac{1}{2}$
- $C_d(v_{17}) = \frac{1}{3}$
- $C_d(v_7) = \frac{2}{3}$
- $C_d(v_{18}) = \frac{1}{6}$
- $C_d(v_8) = \frac{2}{3}$
- $C_d(v_{19}) = \frac{5}{6}$
- $C_d(v_9) = \frac{1}{3}$
- $C_d(v_{20}) = \frac{1}{2}$
- $C_d(v_{10}) = \frac{1}{3}$
- $C_d(v_{21}) = \frac{1}{2}$
- $C_d(v_{11}) = \frac{1}{3}$
- $C_d(v_{22}) = \frac{1}{3}$

The same can be done to Figure 2:

i	d_i	$\frac{d_i}{n}; n = 6$	$\frac{d_i}{max(d_i)}; = 4$
1	4	$\frac{2}{3}$	1
2	2	$\frac{1}{3}$	$\frac{1}{2}$
3	2	$\frac{1}{3}$	$\frac{1}{2}$
4	3	$\frac{1}{2}$	$\frac{3}{4}$
5	2	$\frac{1}{3}$	$\frac{1}{2}$
6	1	$\frac{1}{6}$	$\frac{1}{4}$

If Equation 2 were to be used to normalize the results of these examples, one can observe that the nodes of Figure 1 have much less of an influence on their network than the nodes in Figure 2. This should make sense, as this difference can be thought of as spreading a virus through a household versus a large city. On the other hand, Equation 3 depicts the difference between the most and second most central node as much larger than Equation 2.

B. Betweenness Centrality

The definition of Betweenness Centrality is more complex than that of Degree Centrality. The mathematical definition of calculating the betweenness centrality can be described as follows:

$$C_b(v_i) = \sum \frac{\sigma_{st}(v_i)}{\sigma_{st}} \quad \text{Equation 4}$$

where σ_{st} represents the number of shortest distance between two nodes labeled *s* and *t* that are

not v_i, while $\sigma_{st}(v_i)$ is the number of these shortest distance paths that go through v_i. Technically speaking, the betweenness centrality of each node is already normalized because of the σ_{st} that is in the denominator. However, in order to normalize the betweenness centrality for the sake of comparing graphs, the amount of paths between two points *s* and *t* must be calculated. If a graph has *n* nodes, there are $n-1$ nodes to consider when counting the number of paths because one of v_i is excluded. Thus there are $\frac{n-1}{2}$ different paths. However, because the order of the assignments of *s* and *t* need to be considered, there are a total of $2\ \frac{n-1}{2}$ different paths.

Normalization: $\frac{C_b(v_i)}{2\frac{n-1}{2}}$ Equation 5

Example 2:

Figure 2 will be examined to demonstrate the calculation of the betweenness centrality of v_1. First, v_2 will be assigned as *s*, and v_5 will be assigned as *t*. Thus, $\sigma_{st} = 1$.This is because the shortest path from v_2 to v_5 crosses through v_1, and there is only one such path of the same length. With that being said, $\sigma_{st}(v_1) = 1$ as well, since, as mentioned, the shortest path crosses through v_1. The same will be repeated for all permutations of nodes *s* and *t*:

s, *t*	σ_{st}	$\sigma_{st}(v_1)$
2, 6	1	1
2, 5	1	1
2, 4	1	1
2, 3	1	0
3, 6	1	1
3, 5	1	1
3, 4	1	1
4, 5	1	0
4, 6	1	0
5, 6	1	0

As the calculation for $C_b(v_1)$ states, all of $\frac{\sigma_{st}(v_1)}{\sigma_{st}}$ must be summed in order to calculate the betweenness centrality of v_1. It can be seen from the table that $C_b(v_1) = 6$.However, this needs to be multiplied by 2 because of the permutations of *s* and *t*, so $C_b(v_1) = 12$. The same calculations can be done to find $C_b(v_i)$ for all $1 \leq i \leq 6$.

i	$C_b(v_i)$
1	12
2	0
3	0
4	6
5	0
6	0

It can be seen that v_1 is the most central. Again, these results need to be normalized in order to compare the results between different graphs. As mentioned, the factor used to normalize these centralities would be $2\ \frac{n-1}{2}$,which in this case would be $2\ \frac{5}{2} = 20$.Thus, the normalized centralities are as follows:

i	$C_b(v_i)$
1	$\frac{3}{5}$

2	0
3	0
4	$\frac{3}{10}$
5	0
6	0

Overall, betweenness centrality doesn not seem to capture the features of the graph as desired. It can be seen, for example, that v_5 and v_6 have the same centrality despite v_5 being intuitively more central.

C. Closeness Centrality

Closeness centrality is relatively similar to betweenness centrality in the sense that it uses this concept of shortest distance. The closeness centrality of a node can be expressed as follows:

$C_c(v_i) = \frac{1}{I_{v_i}}$	Equation 6
$I_{v_i} = \frac{1}{n-1}\sum_{v_i \neq v_j} d_{i,j}$	Equation 7

where $d_{i,j}$ is the shortest distance between node v_i and node v_j. The idea behind calculating centrality in this manner is to take the average of the shortest distances, I_{v_i}, between v_i and all other nodes v_j. Intuitively, the most central node would have the smallest I_{v_i}, which is why the reciprocal of this value is taken: if the most central node has the smallest I_{v_i}, then it would have the largest $\frac{1}{I_{v_i}}$.

Example 3:

The closeness centrality of each node of Figure 2 will be calculated. The first step is to use Equation 7. For v_1, I_{v_1} would be calculated as $\frac{1}{6-1}(1+1+1+1+2) = \frac{6}{5}$, since the shortest distance between v_1 and v_2, v_3, v_4, and v_5 would be 1; and the shortest distance between v_1 and v_6 would be 2. Then $C_c(v_1) = 1$. This can be done for the other vertices as well.

i	I_{v_i}	$C_c(v_i)$
1	$\frac{6}{5}$	$\frac{5}{6}$
2	$\frac{9}{5}$	$\frac{5}{9}$
3	$\frac{9}{5}$	$\frac{5}{9}$
4	$\frac{7}{5}$	$\frac{5}{7}$
5	$\frac{8}{5}$	$\frac{5}{8}$
6	$\frac{11}{5}$	$\frac{5}{11}$

It can be seen that v_1 is the most central by this algorithm as well. Closeness centrality, for the purposes of modeling an epidemic, seems to capture the features of the graph relatively well compared to betweenness centrality: nodes v_2 and v_3 have equal centralities, v_6 is about half as central as v_1, etc.

D. Eigenvector Centrality

Just as the name suggests, calculating the centrality of each node is based on the calculation of eigenvectors and eigenvalues. As a brief summary of these two terms, an eigenvector is a vector that only scales under the transformation of a matrix, while an eigenvalue is the factor by which the vector is scaled by. The mathematical definition of the eigenvector centrality, $C_e(v_i)$, is as follows:

$$C_e(v_i) = \frac{1}{\lambda}\sum_{j=1}^{n} A_{j,i}C_e(v_j) \quad \text{Equation 8}$$

where λ is a constant that will result in an eigenvalue, and A is the adjacency matrix of the graph, and $A_{j,i}$ represents the entry in the jth row and the ith column of A. The adjacency matrix is constructed by first making an $n \times n$ matrix, where n is the number of nodes in the graph. If there is an edge that connects v_i and v_j, then $A_{i,j}$ and $A_{j,i}$ are filled as 1. Every other entry is equal to 0, making A symmetric across the diagonal of the matrix.

Proof:

Suppose there is a $n \times 1$column matrix $C_e$such that the ith entry is $C_e(v_i)$and n is the number of nodes in a given graph. Now, the largest eigenvalue, λ, and the eigenvector, C_e, are desired. Let A be the adjacency matrix of the graph. The following equation and steps can be written:

$$AC_e = \lambda C_e$$

From there, λ can be solved for as such:

$$AC_e - \lambda C_e = 0$$
$$C_e(A - \lambda I) = 0$$

where I is the $n \times n$ identity matrix. The determinant of $A - \lambda I = 0$ in order for this equation to be true, and this is how the values of λ are found. The largest value of λmust be used to find the centrality of each node.

Example 4:

Figure 2 will be used for this example. The adjacency matrix of Figure 2 is:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Thus, $\lambda I - A$would be equal to the following:

$$A - \lambda I = \begin{bmatrix} -\lambda & 1 & 1 & 1 & 1 & 0 \\ 1 & -\lambda & 1 & 0 & 0 & 0 \\ 1 & 1 & -\lambda & 0 & 0 & 0 \\ 1 & 0 & 0 & -\lambda & 1 & 1 \\ 1 & 0 & 0 & 1 & -\lambda & 0 \\ 0 & 0 & 0 & 1 & 0 & -\lambda \end{bmatrix}$$

The determinant of this matrix must be equal to 0 in order for $C_e(A - \lambda I) = 0$:

$$\lambda^6 - 7\lambda^4 + 4\lambda^3 + 9\lambda^2 - 6\lambda - 1 = 0$$

This leads to the following solutions:

$$\lambda \approx \{2.629, 1.230, 0.140, -1, -1.320, -1.678\}$$

Thus, 2.629 is taken as λ.This value can be used to solve for $C_e$as follows:

$$\begin{bmatrix} -\lambda & 1 & 1 & 1 & 1 & 0 \\ 1 & -\lambda & 1 & 0 & 0 & 0 \\ 1 & 1 & -\lambda & 0 & 0 & 0 \\ 1 & 0 & 0 & -\lambda & 1 & 1 \\ 1 & 0 & 0 & 1 & -\lambda & 0 \\ 0 & 0 & 0 & 1 & 0 & -\lambda \end{bmatrix} C_e = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This matrix can be turned into an augmented matrix, which can then be row-reduced to get

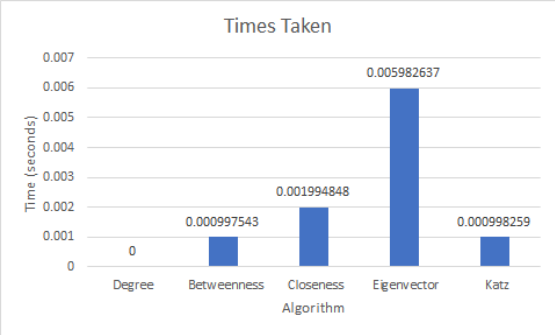
$$C_e = \begin{bmatrix} 3.557x \\ 2.184x \\ 2.184x \\ 2.629x \\ 2.353x \\ x \end{bmatrix}$$

each of $C_e(v_i)$.

where x is some free variable. It can be seen that relative to each other, $C_e(v_1)$ is the largest, meaning v_1 is the most central node.

E. Katz Centrality

Katz centrality is extremely similar to eigenvector centrality. It can be recalled that the eigenvector centrality of a node v_i can be expressed as:



$$C_e(v_i) = \frac{1}{\lambda}\sum_{j=1}^{n} A_{j,i}C_e(v_j)$$

The Katz centrality of a node, $C_k(v_i)$, can be defined as follows:

$$C_k(v_i) = \alpha\left(\sum_{j=1}^{n} A_{j,i}C_k(v_j)\right) + \beta \quad \text{Equation 9}$$

Immediately, the resemblance between the two centralities can be seen. The only difference is the addition of a constant term, β.

Using each of these methods, the three most central nodes of Figure 1 are displayed in Table 1 below.

Table 1

	1st	2nd	3rd
Degree	v_1	v_7	v_8
Betweenness	v_1	v_7	v_3
Closeness	v_1	v_5	v_6
Eigenvector	v_1	v_5	v_7
Katz	v_1	v_7	v_{19}

It can be seen that there is a general trend among the most central nodes of Figure 1: v_1 is the most central node, regardless of what centrality metric is used; v_7 is the second most frequent node that appears; and the third most central node is mostly a random one that is inconsistent among the centralities. For practical purposes, the nodes highlighted by the Katz centrality algorithm seem the most significant.

Figure 3 displays the time it takes to find the centralities of every node according to the algorithm.

Figure 3

III Detecting Bridges

To be precise, a bridge is an edge that separates a graph into an increased number of connected subgraphs. In Figure 1, the bridges include the edges between v_1 and v_5, v_6 and v_{16}, and v_{13} and v_{14}. These bridges will be used to find subgraphs of Figure 1. Finding the centralities of these subgraphs would be more efficient than finding the most central nodes of the graph as a whole, as the time complexity would be lower. We used an algorithm that was introduced in [8].

In order to determine whether an edge is a bridge, a Depth-First Search (DFS) will be used for this algorithm. The gist of this algorithm is to pick and mark an arbitrary node as a root. From there, an unmarked adjacent node will be moved to until there are no more unmarked adjacent nodes. The final node marked will then be backtracked from, and then other unmarked nodes will be traversed to.

Figure 4	**Figure 5**

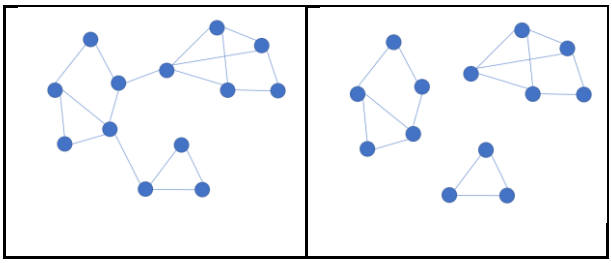


Figure 5 demonstrates Figure 4 without the bridges. It can be seen that these edges cause the number of connected portions to increase, to the point of having three subgraphs. This is when normalization comes into play. For Figure 4, the k most central nodes for some positive integer k can simply be found. However, once the graph has been split as in Figure 5, normalization must be used to determine and compare the most central nodes of each subgraph, from which the k most central nodes can be found.

Table 2 demonstrates the process by which bridges are found using DFS. It should be noted that the order that the nodes were followed is not going to be the same every time this search algorithm is performed and that which adjacent, unvisited node is visited next is completely random.

Table 2:

Current node	Visit order	Low order	Validate (Whether it's a bridge or not)	Update (changing low order)
1	1	1		
2	2	2		
9	3	3		
11	4	4		
3	5	5		
7	6	6		
12	7	7		
13	8	8		
14	9	9	Since the low order of v_{14} is greater than the visit order of v_{13}, this edge is a bridge.	
13	8	6	Since the low order of v_{13} is less than the visit order of v_{12}, this edge is not a bridge.	When backtracking from v_{13}, the low order of v_{13} must be replaced by the low order of v_7, from 8 to 6.
12	7	6	Since the low order of v_{12} is equal to the visit order of v_7, this edge is not a bridge.	When backtracking from v_{12}, the low order of v_{12} must be replaced by the low order of v_7, from 7 to 6.
7	6	1	Since the low order of v_7 is less than the visit order of v_3, this edge is not a bridge.	When backtracking from v_7, the low order of v_7 must be replaced by the low order of v_1, from 6 to 1.
3	5	1	Since the low order of v_3 is less than the visit order of v_{11}, this edge is not a bridge.	When backtracking from v_3, the low order of v_3 must be replaced by the low order of v_7 from 5 to 1.
11	4	1	Since the low order of v_{11} is less than the visit order of v_9, this edge is not a bridge.	When backtracking from v_{11}, the low order of v_{11} must be replaced by the low order of v_3, from 4 to 1.
9	3	1	Since the low order of v_9 is less than the visit order of v_2, this edge is not a bridge.	When backtracking from v_9, the low order of v_9 must be replaced by the low order of v_{11} from 3 to 1.
2	2	1	Since the low order of v_2 is equal to the visit order of v_1, this edge is not a bridge.	When backtracking from v_2, the low order of v_2 must be replaced by the low order of v_9 from 2 to 1.
1	1	1		
4	10	10		
15	11	11		
10	12	12→1	Since the low order of v_{10} is less than the visit order of v_{15}, this edge is not a bridge.	When backtracking from v_{10}, the low order of v_{10} must be replaced by the low order of v_1, from 12 to 1.
15	11	1	Since the low order of v_{15} is less than the visit order of v_4, this edge is not a bridge.	When backtracking from v_{15}, the low order of v_{15} must be replaced by the low order of v_{10}, from 11 to 1.
4	10	1	Since the low order of v_4 is equal to the visit order of v_1, this edge is not a bridge.	When backtracking from v_4, the low order of v_4 must be replaced by the low order of v_{15}, from 10 to 1.
1	1	1		
5	13	13	Since the low order of v_5 is greater than the visit order of v_1, this edge is a bridge.	
6	14	14		
16	15	15	Since the low order of v_{16} is greater than the visit order of v_6, this edge is a bridge.	
6	14	13	Since the low order of v_6 is equal to the visit order of v_5, this edge is not a bridge.	When backtracking from v_6, the low order of v_6 must be replaced by the low order of v_5, from 14 to 13.
8	16	16		
17	17	17		
20	18	18		
22	19	19		
21	20	20		

19	21	21		
18	22	22→ 16	Since the low order of v_{18} is less than the visit order of v_{19}, this edge is not a bridge.	When backtracking from v_{18}, the low order of v_{18} must be replaced by the low order of v_8, from 22 to 16.
19	21	16	Since the low order of v_{19} is less than the visit order of v_{21}, this edge is not a bridge.	When backtracking from v_{19}, the low order of v_{19} must be replaced by the low order of v_{18}, from 21 to 16.
21	20	13	Since the low order of v_{21} is less than the visit order of v_{22}, this edge is not a bridge.	When backtracking from v_{21}, the low order of v_{21} must be replaced by the low order of v_5, from 20 to 13.
22	19	13	Since the low order of v_{22} is less than the visit order of v_{20}, this edge is not a bridge.	When backtracking from v_{22}, the low order of v_{22} must be replaced by the low order of v_{21}, from 19 to 13.
20	18	13	Since the low order of v_{20} is less than the visit order of v_{17}, this edge is not a bridge.	When backtracking from v_{20}, the low order of v_{20} must be replaced by the low order of v_{22}, from 18 to 13.
17	17	13	Since the low order of v_{17} is less than the visit order of v_{20}, this edge is not a bridge.	When backtracking from v_{17}, the low order of v_{17} must be replaced by the low order of v_{20}, from 17 to 13.
8	16	13	Since the low order of v_8 is less than the visit order of v_6, this edge is not a bridge.	When backtracking from v_8, the low order of v_8 must be replaced by the low order of v_{17}, from 16 to 13.
6	15	13		When backtracking from v_6, the low order of v_6 must be replaced by the low order of v_5, from 15 to 13.

IV. Community Detection

Now that bridges can be detected and can be detached, communities must be detected so that the node centrality algorithm can finally be put to use to evaluate the graph. To elaborate, a community is simply a group of nodes that has a high modularity, which takes on a value between $-\frac{1}{2}$ and 1, inclusive. In order to measure this modularity and therefore determine the communities of a node, the Louvain algorithm will be used.

The Louvain algorithm[9] is a well-known community detection method that is widely used in network analysis. The inspiration for this method is to optimize modularity, which would thus imply that the communities have been determined in the most optimal way possible. The calculation for the algorithm is as follows:

$$Q = \frac{1}{2m}\sum_{i,j}\ [A_{i.j} - \frac{k_i k_j}{2m}]\ \delta(c_i, c_j)$$

where Q is modularity, $A_{i,j}$ is the weight of the edge between nodes v_i and v_j, k_i and k_j are the sum of the weights of the edges attached to nodes v_i and v_j, m is the sum of all of the edge weights in the graph, c_i and c_j are the communities of nodes v_i and v_j, and $\delta$is the Kronecker delta function, which is defined as 1 if $c_i = c_j$ and 0 otherwise.

However, the weakness of this algorithm arises from this motivation: because the algorithm is testing all possible combinations of communities to find the most optimal modularity, the algorithm trades increased accuracy for increased calculation time. In order to combat this inefficiency, the Louvain algorithm is often split into two iterative processes.

The first step to maximize modularity efficiently is to assign each node to its own community. Afterwards, for each v_i, the change in modularity is found by removing v_i into a neighboring community of v_j. This change in modularity can be calculated using the following:

$$\Delta Q = [\frac{\Sigma_{in} + 2k_{i,in}}{2m} - (\frac{\Sigma_{tot} + k_i}{2m})^2] - [\frac{\Sigma_{in}}{2m} - (\frac{\Sigma_{tot}}{2m})^2 - (\frac{k_i}{2m})^2]$$

where $\Sigma_{in}$is the sum of all the weights of the edges inside the community that node v_i is moving into, Σ_{tot} is the sum of the weights of all connection occurring in the new community, k_i is the weighted degree of v_i, and $k_{i,in}$ is the sum of the weights of the links between v_i and the other nodes in the community that v_i is moving into.

Since Q is supposed to be maximized for the most optimal representation of communities in a network, a positive ΔQ is desirable, as it

demonstrates that modularity has increased, thus becoming more optimal. This calculation is repeated until no increase in modularity can occur, after which the second step can be used.

The second step is to build a new community from the communities that were created from the first step, where the weights of the new edges between nodes are simply the sum of the weights of edges between the communities. Essentially, the communities are replaced by single nodes that represent them. The iterative part of this process arises from performing the first and second step, in that order, repeatedly until the optimal communities have been reached.

V. Evaluation

A. Device Specs

The CPU on the computer that was used is an Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz. The GPU is an NVIDIA GeForce RTX 2070 with Max-Q Design. The computer also runs on Windows 10 and has 16.0 GB of installed RAM. The language used to evaluate these results was Python, and the IDE used is Pyzo.

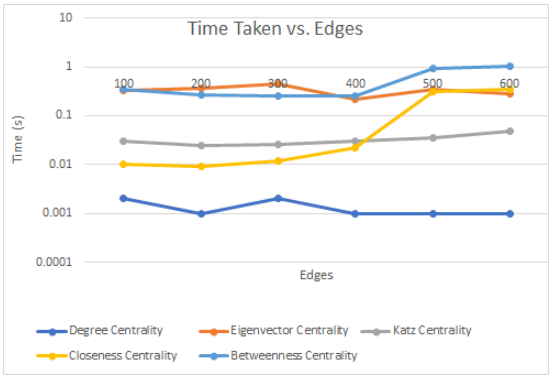
B. Data

Synthetic data would be used to emulate a community as closely as possible. The first types of graphs that were used all had 800 nodes, and a number of edges ranging from 100 to 600. The next graphs have 200 to 1,400 nodes, where the number of edges is directly proportional to the number of nodes. Finally, to test the effects of using bridge and community detection, a graph with 1,443 nodes and 897 edges was used. As shown by Table 1, the results of the node centrality algorithms seem to align with intuition. Each of these graphs were randomly generated.

C. Evaluation of Node Centralities and Proposed Methods

Figure 6 shows the time taken using each method to calculate the node centrality of a random graph with 800 nodes.

Figure 6



Looking at Figure 6, the differences in time computed for each algorithm do not begin to show until there are about 420 edges between the 800 nodes. Thus, a direct proportion will be used to run trials on graphs with different amounts of nodes; the ratio between edges and nodes will be 420:800. Although it is difficult to say whether this would be the case in a real-world scenario, it is expected that the density — the ratio of edges to nodes — would remain constant.

Figure 7

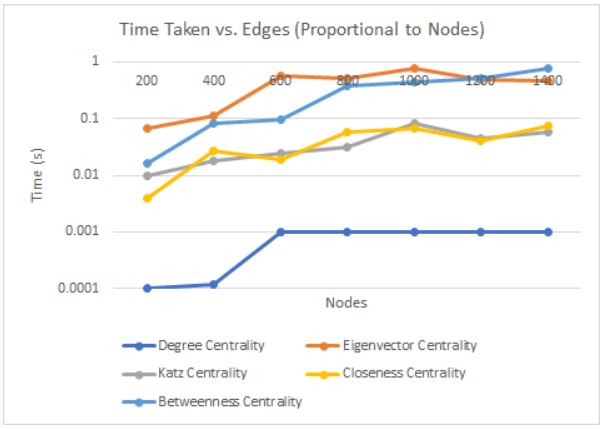


Figure 7 compares the time it takes to compute the node centrality based on the ratio of edges to

nodes that was determined from Figure 6. Generally, it can be seen that eigenvector centrality takes the longest, while degree centrality takes the shortest amount of time. In addition, these times can be further improved by implementing the concepts of bridge and community detection.

Figure 8

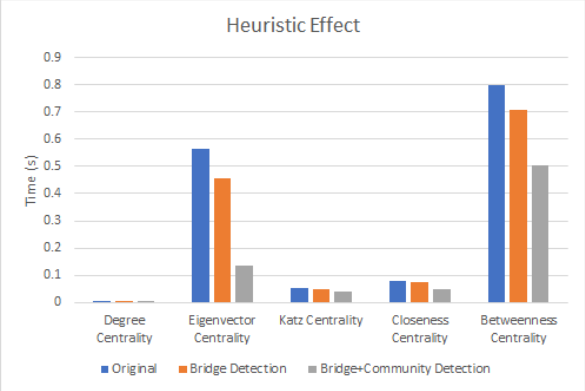


Figure 8 demonstrates the time taken to compute the node centrality for each algorithm using neither bridge nor community detection, bridge detection only, and both bridge and community detection. As shown in Figure 8, the time it takes to compute these node centralities decreases, sometimes by dramatic amounts, by incorporating bridge and community detection.

VI. Conclusion and Future works

In summary, node centrality, as its names suggests, measures the connectivity of a node to others. The significance behind this concept is that the most influential node can be determined from a given graph, which can be used to represent any type of network that appears in a society, including communities in social media, analyzing business interactions, and even modeling contact patterns during an epidemic.

The proposed method of calculation for node centrality (using bridge and community detection) differentiates from simply calculating the node centrality of the entire graph because of its efficiency: bridge and community detection break the graph into communities to calculate the node centrality of these. Then, the node centralities of each community are compared to determine the most central nodes. As shown in Figure 8, the incorporation of bridge and community detection proved to be effective in cutting the calculation time for all node centrality algorithms. The results are especially noticeable when looking at calculating the eigenvector centrality, as the time has been cut from about 0.57 seconds to about 0.46 seconds with bridge detection alone. This time was decreased even farther with the use of community detection, dropping it down to an astonishing 0.14 seconds.

In future works, the proposed model will further consider the statistical properties of the graph in order to make the analysis more accurate. In addition to an improvement in accuracy, the computation time would also be reduced in order to make the model have tolerable time.

Reference

[1] X. Zhao, S. Guo, and Y. Wang, "The Node Influence Analysis in Social Networks Based on Structural Holes and Degree Centrality," in IEEE International Conference on CSE and IEEE International Conference on EUC, 2017, pp. 708-711.

[2] L. Lv, K. Zhang, T. Zhang, D. Bardou, J. Zhang, and Y. Cai, "PageRank centrality for temporal networks," in Physics Letters A, 2019, pp. 1-8.

[3] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck, “Computing classic closeness centrality, at scale,” in COSN, 2014, pp. 37–50.

[4] Freeman, L.C., "A set of measures of centrality based on betweeness", Sociometry, Vol. 40, pages 35-41, 1977.

[5] S. Gao, J. Ma, Z. Chen, G. Wang, C. Xing, "Ranking the spreading ability of nodes in complex networks based on local structure," in Physica A 403 (6), 2014, pp. 130–147.

[6] G. Sabidussi, "The centrality index of a graph," in Psychometrika 31 (4), 1966, pp. 581–603.

[7] L.C. Freeman, "Centrality in social networks conceptual clarification," in Soc. Netw. 1 (3), 2008, pp. 215–239.

[8] Corradini, Enrico, "Defining and detecting k-bridges in a social network: The Yelp case, and more"in Knowledge-Based Systems,Volume 195,2020, 105721

[9] Ghosh, Sayan, "Distributed Louvain Algorithm for Graph Community Detection" in 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS) IPDPS Parallel and Distributed Processing Symposium, 2018 IEEE International. :pp. 885-895 ,2018

[10] L. Page, S. Brin, R. Motwani, T. Winograd, "The Pagerank Citation Ranking: Bringing Order to the Web," in Technical report, Stanford InfoLab, 1999, http://ilpubs.stanford.edu:8090/422/.

[11] F. Cadini, E. Zio, and C. Petrescu, "Using centrality measures to rank the importance of the components of a complex network infrastructure," in CRITIS, 2008, pp. 155–167.