HIGH SCHOOL EDITION
Journal of Student Research
Volume 11 Issue 3 (2022)

Implementation of Computing Node Centrality with Bridge and Community Detection

Leonardo Chung[1] and Kun Young Park[#]

[1]Phillips Exeter Academy, Exeter, NH, USA
#Advisor

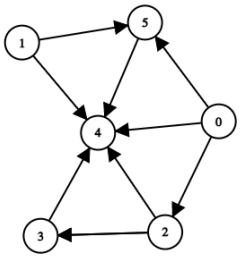
ABSTRACT

As our society grows, the potential of certain people affecting the masses has increased dramatically with the presence of media, virality, and celebrities. Therefore, it is essential to know which persons might be influencing, swaying, or manipulating the public the most in a social network. Similarly, finding the most important webpage can impact advertisement spending for corporations. I propose and determine a better method to find the most significant "influencer" and other real-world applications using graph theory in discrete mathematics. There are many methods of node centrality, and some have more advantages than others. As measuring this score becomes more complex, accuracy is guaranteed, but time complexity increases simultaneously. In particular, when the score is a case where the relationship between nodes is important, the time complexity shows an extreme increase. Graphs representing the real world have a lot of nodes and edges in many cases, and experiments have found that if applied as they are, the time efficiency will be extremely low. To compensate for this point, bridge detection and community detection, a method of dividing a large graph into several subgraphs at a low level, were applied to change the nested loop operation to a simple sum of operations in series. Furthermore, a model, which is the most appropriate combination, was proposed and experimentally proved in consideration of trade-offs. The reason for selecting the ratio of node and edge numbers to increase the experiment's credibility was also described.

Introduction

With the development of communication and convenience, public accessibility to the internet has increased significantly. Modern individuals spend most of their time on the internet. Therefore, it is instrumental for many businesses to know which web pages are most "influential.". This can be due to advertisements, efficiency, resource usage, and popularity.

We can utilize a method called "node centrality" that is prominent in graph theory.



<Figure 1>

Figure 1. Small internet network. The nodes in the graph symbolize pages (websites). The arrows serve as paths to get to certain websites from other ones. Together, nodes and arrows create a graph that can be analyzed. For example, in-degree centrality, the node that has the most edges directed towards it can be said to be the most influential page. In **Figure 1**, this would be node 4. We can also say that the page with the most edges directed from it is a starting page on the internet (think Google). In this graph, this would be node 0.

There are many methods of node centrality, and some have more advantages than others. However, as we attempt to approach more accurate scores of certain nodes, the time taken to reach these points rises exponentially. Therefore, it is necessary to use a method to reduce the time duration and complexity by splitting the entire graph. This paper compares the time complexity for each node centrality of splitting a graph through bridge detection and the method of dividing the entire graph into clustered community units through community detection.

We first introduce centrality, bridge detection, and community detection as related methods. We then establish a proposed model in which we will try to limit the time complexity of bridge detection and community detection because they increase by a significant amount currently. Thereafter, we conduct an evaluation in which the time complexities of the centralities, bridge detection, community detection, and the proposed model are all compared. We conclude with our analysis and suggest future work that might be done with our proposed model and information.

Methods

Node Centrality

Degree Centrality

A **degree** is the number of edges attached to a node. A **degree centrality** can be calculated based on degree information. If the degree of a node is higher, then the node is considered more influential. A degree centrality can be written as:

$$C_d(v_i) = degree\ of\ v_i = D(v_i)$$

We can normalize C_d in 2 ways,

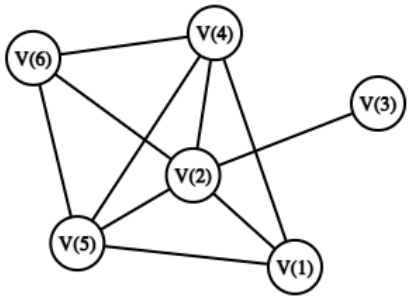
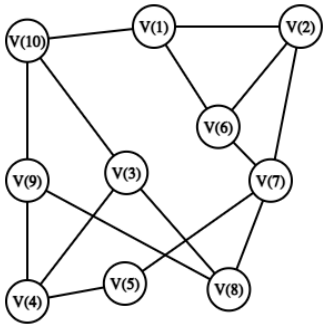
1) First, you can divide the number of edges in a graph by the degree of a node:

$$\frac{degree}{number\ of\ edges\ in\ graph} = \frac{D(v_i)}{|E|}$$

2) Or by the maximum degree:

$$\frac{degree}{maximum\ degree} = \frac{D(v_i)}{D_{max}(v_i)}$$

In this case, the following must be true: $0 \leq C_d \leq 1$

	
<Figure 2>	***<Figure 3>***

Example 1: Consider **Figure 2**. The basic degree centralities are:

$C_d(v_1) = 3$
$C_d(v_2) = 5$
$C_d(v_3) = 1$
$C_d(v_4) = 4$
$C_d(v_5) = 3$
$C_d(v_6) = 3$

Therefore, using degree centralities, the most influential node is v_2.

Example 2: Consider **Figure 2** and **Figure 3**. The **normalized** degree centralities of the figures are shown in **Table 1**.

Table 1.

Divide by $\|E\|$ in ***Figure 2.*** $\|E\|= 6$	Divide by $\|E\|$ in ***Figure 3.*** $\|E\| = 15$
$C_d(v_1) -> \frac{3}{6}$ $C_d(v_2) -> \frac{5}{6}$ $C_d(v_3) -> \frac{1}{6}$ $C_d(v_4) -> \frac{4}{6}$ $C_d(v_5) -> \frac{4}{6}$	$C_d(v_1) -> \frac{3}{15}$ $C_d(v_2) -> \frac{3}{15}$ $C_d(v_3) -> \frac{3}{15}$ $C_d(v_4) -> \frac{3}{15}$ $C_d(v_5) -> \frac{2}{15}$ $C_d(v_6) -> \frac{3}{15}$ $C_d(v_7) -> \frac{4}{15}$ $C_d(v_8) -> \frac{3}{15}$ $C_d(v_9) -> \frac{3}{15}$

	$C_d(v_{10}) -> \frac{3}{15}$
Divide by D_{max} in ***Figure 2.*** D_{max} = 5	Divide by D_{max} in ***Figure 3.*** D_{max}= 4
$C_d(v_1) -> \frac{3}{5}$ $C_d(v_2) -> \frac{5}{5}$ $C_d(v_3) -> \frac{1}{5}$ $C_d(v_4) -> \frac{4}{5}$ $C_d(v_5) -> \frac{4}{5}$	$C_d(v_1) -> \frac{3}{4}$ $C_d(v_2) -> \frac{3}{4}$ $C_d(v_3) -> \frac{3}{4}$ $C_d(v_4) -> \frac{3}{4}$ $C_d(v_5) -> \frac{2}{4}$ $C_d(v_6) -> \frac{3}{4}$ $C_d(v_7) -> \frac{4}{4}$ $C_d(v_8) -> \frac{3}{4}$ $C_d(v_9) -> \frac{3}{4}$ $C_d(v_{10}) -> \frac{3}{4}$
<Table 1>	

In small graphs, both methods provide accurate and understandable results; however, when looking at a large graph with many nodes, dividing by $|E|$ yields more accurate results as we can better see how the nodes compare to the graph as a whole. A denominator of 15 is much different than a denominator of 4, so we can understand this data in two different ways based on the method used.

Betweenness Centrality

Betweenness centrality is measured by dividing the number of shortest paths from Node S to Node T by the number of shortest paths from Node S to Node T that includes v_i.

The betweenness centrality of v_i can be written as:

$$C_b = \sum_{s \neq t \neq v_i} \left(\frac{\gamma_{st}(v_i)}{\gamma_{st}}\right)$$

Multiplying the sum of the centralities by 2 is necessary because *s* and *t* can be interchanged.

Using these values without a common base can result in difficulty when comparing node efficiency and importance. Therefore, it is important to normalize these values. We can do this by using this equation representation:

$$C_b = \sum_{s \neq t \neq v_i} \left(\frac{\gamma_{st}(v_i)}{\gamma_{st}}\right) [Use\ the\ maximum\ value] = \sum_{s \neq t \neq v_i} (1) = 2 \times \binom{n-1}{2} = \frac{(n-1)!}{(n-3)!\,2!} \times 2 = (n-1)(n-2)$$

where n is the number of nodes in the graph.

To normalize C_b, divide its result by $(n-1)(n-2)$.

Example 3: Take the graph in **Figure 2** as an example. Let us find $C_b(v_2)$ through calculations of the starting and ending nodes. This data is shown in **Table 2**.

Table 2.

s	t	**Centrality**
1	3	1
1	4	0
1	5	0
1	6	$\frac{1}{3}$
3	4	1
3	5	1
3	6	1
4	5	0
4	6	0
5	6	0
<Table 2>		

The sum of the centralities is $1 + \frac{1}{3} + 1 + 1 + 1 = \frac{13}{3}$. Multiply by 2 to get $\frac{26}{3}$.

We can now normalize by dividing $\frac{26}{3}$ by (6 - 1)(6 - 2). Therefore, $C_b(v_2) = \frac{13}{30}$(normalized).

Example 4: Here are all the normalized betweenness centralities of **Figure 2** shown in **Table 3**.

Table 3.

$C_b(v_2)$	normalized $C_b(v_2)$
1	0
2	$\frac{13}{30}$
3	0
4	0
5	$\frac{2}{45}$
6	0
<Table 3>	

It is pretty easy to see, in some graphs, which points are most influential. Clearly, the node with the largest centrality, v_2, is the most influential. Betweenness centrality is useful when many of a graph's betweenness centralities are 0, making it easy to identify better nodes.

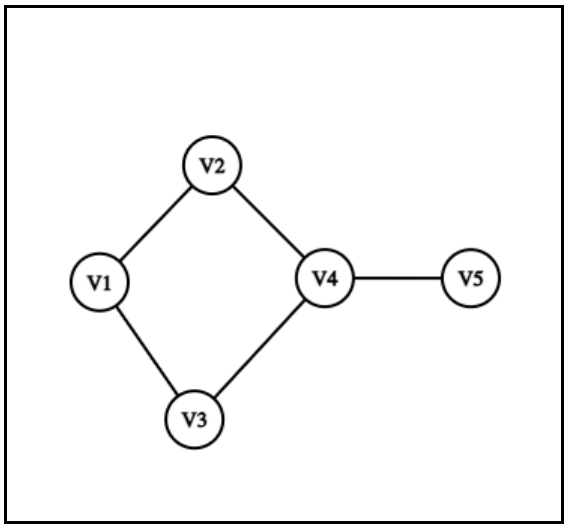
Closeness Centrality

The **closeness centrality** of a node is the reciprocal of the product of $\frac{1}{n-1}$ and the sum of the distances of the node to all other nodes (if n is the number of nodes)

It can be calculated by using the formula:

$$C_c = \frac{1}{l_{vi}};\ l_{vi} = \frac{1}{n-1}\sum_{v_i \neq v_j} l_{i,j}$$

This looks complicated, but it is actually very straightforward.



<Figure 4>

Let's calculate $C_c\ (v_i)$ as seen in **Table 4**.

Table 4.

Node	l_{vi}	**Centrality**
V1	7	$\frac{4}{7}$
V2	6	$\frac{4}{6}$
V3	6	$\frac{4}{6}$
V4	5	$\frac{4}{5}$
V5	8	$\frac{4}{8}$

<Table 4>

Eigenvector Centrality

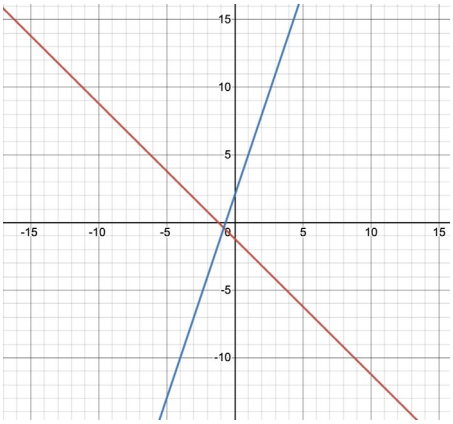
It is first important to know what an identity element is. Suppose a number (say *number1*) is operated upon by an operation using a certain number (say *number2*); if the result is *number1*, then *number2* is an **identity element**. For example, if a is a constant, a + 0 = a. This means that for addition, 0 is its **identity element.**

Identity elements are also important in matrix transformations. We can transform a line y to y' using a matrix A. This can be represented by Ay = y'. There are several changes that we can make, specifically, the **angle** (direction) and **magnitude** (length) of the line. A change that alters only the length of y is called a **scalar**, and a change that alters the length and direction of y is called a **vector**.

For example, Let matrix A = $\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$, and y = 3x + 2.

$$y' = Ay = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} y$$

$$y' = -x' - \frac{6}{5}$$



<Figure 5>

Figure 5, let the blue line be y and the red line be y'. As you can see, y has undergone a transformation that alters both the **direction** and **magnitude** in y's (vector change). Consider matrix transformations as operations. If the transformed blue line becomes a straight line that continues in the same direction as the blue line after matrix operation A, then A can be viewed as an **identity element**.

Being aware that size doesn't matter much in the case of vectors, we can use the constant lambda (λ) as an "**eigenvalue**." The transformation of the case, therefore, can be like this:

$$A_x = \lambda x$$

(x is the **eigenvector**, and λ is the **eigenvalue**)

Here is a good example of finding the eigenvalues of a 2x2 matrix.

$$A = \begin{bmatrix} 4 & 8 \\ 6 & 26 \end{bmatrix}$$

$$A - \lambda I = \begin{bmatrix} 4 & 8 \\ 6 & 26 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 4-\lambda & 8 \\ 6 & 26-\lambda \end{bmatrix}$$

$$\det(A - \lambda I) = \begin{vmatrix} 4-\lambda & 8 \\ 6 & 26-\lambda \end{vmatrix} = (4 - \lambda)(26 - \lambda) - (8)(6) = 0$$

$104 - 30\lambda + \lambda^2 - 48 = 0$

$\lambda^2 - 30\lambda + 56 = 0$

$(\lambda - 28)(\lambda - 2) = 0$

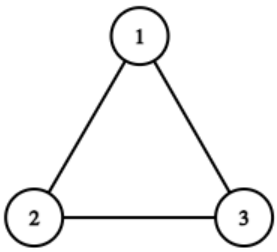
Eigenvalues of matrix A: $\lambda_1 = 28, \lambda_2 = 2$

Basically, for every matrix A, there are vectors whose **directions** are unchanged by A. These are the **eigenvectors**, and their lengths are scaled by the **eigenvalue.**

To calculate eigenvector centrality, use the following formula:

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

A_{ij} is called an "adjacency matrix." It is a matrix that represents a graph.



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

<Figure 6>

Figure 6. We can see how to create an adjacency matrix. For every node connected to another one, we add a 1. $C_e$can be represented by $(C_e(v_1), C_e(v_2), C_e(v_3), \ldots, C_e(v_i))^T$. The exponent T means to switch the rows and columns.

This is the equation we use to find eigenvectors, where $\lambda$and$A^T$are matrices and $C_e$is an eigenvector.

$$\lambda \cdot C_e = A^T \cdot C_e$$

Example 6: Finding Eigenvector centrality of **Figure 4.**
First, A (the adjacency matrix) of the figure is:

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

Organizing the equation above:

1) First of all, A^T is equal to A because all adjacency matrices are symmetrical.
$$\lambda \cdot C_e = A \cdot C_e$$
2) Move everything to one side.
$$\lambda \cdot C_e - A \cdot C_e = 0$$
3) Multiply the λmatrix by an identity matrix, and factor out C_e.
$$(\lambda I - A) \cdot C_e = 0$$
4) From this point on, multiplying $\lambda I - A$ by $C_e$can be said to have the same effect as finding the determinant of matrix $\lambda I - A$.

We now will find the eigenvalues of the adjacency matrix A.

$$\det(\lambda I - A) = 0$$

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

$$\det\left(\begin{bmatrix} \lambda & -1 & -1 & 0 & 0 \\ -1 & \lambda & 0 & -1 & 0 \\ -1 & 0 & \lambda & -1 & 0 \\ 0 & -1 & -1 & \lambda & -1 \\ 0 & 0 & 0 & -1 & \lambda \end{bmatrix}\right) = 0$$

$$-\lambda^5 + 5\lambda^3 - 2\lambda = 0$$

The solutions to this equation are:

$$\lambda = -\sqrt{\frac{5}{2} - \frac{\sqrt{17}}{2}}, \sqrt{\frac{5}{2} - \frac{\sqrt{17}}{2}}, -\sqrt{\frac{\sqrt{17}}{2} + \frac{5}{2}}, \sqrt{\frac{\sqrt{17}}{2} + \frac{5}{2}}, 0$$

For simplicity, we will be using the largest value, which is $\sqrt{\frac{\sqrt{17}}{2} + \frac{5}{2}}$. This value is approximately 2.14.

$$\begin{bmatrix} 2.14 & -1 & -1 & 0 & 0 \\ -1 & 2.14 & 0 & -1 & 0 \\ -1 & 0 & 2.14 & -1 & 0 \\ 0 & -1 & -1 & 2.14 & -1 \\ 0 & 0 & 0 & -1 & 2.14 \end{bmatrix}$$

Multiplying this matrix by C_e (the matrix with the eigenvector scores of each node) will result in 0.

$$\begin{bmatrix} 2.14 & -1 & -1 & 0 & 0 \\ -1 & 2.14 & 0 & -1 & 0 \\ -1 & 0 & 2.14 & -1 & 0 \\ 0 & -1 & -1 & 2.14 & -1 \\ 0 & 0 & 0 & -1 & 2.14 \end{bmatrix} \text{x} \begin{bmatrix} C_e(v_1) \\ C_e(v_2) \\ C_e(v_3) \\ C_e(v_4) \\ C_e(v_5) \end{bmatrix} = 0$$

The eigenvector centrality scores for each of the 5 nodes are, respectively:

$$\begin{bmatrix} C_e(v_1) \\ C_e(v_2) \\ C_e(v_3) \\ C_e(v_4) \\ C_e(v_5) \end{bmatrix} = \begin{bmatrix} 1.67 \\ 1.78 \\ 1.78 \\ 2.14 \\ 1 \end{bmatrix}$$

Katz Centrality

Katz centrality is very similar to eigenvector centrality because they utilize nearly identical equations. Katz centrality is used when eigenvector centrality returns an error or reports a very generalized score, which are common issues when using eigenvectors. The equation for Katz centrality is:

$$C_{Katz}(v_i) = \alpha \sum_{j=1}^{n} A_{ij} C_{Katz}(v_j) + \beta$$

α and β are constants. Every graph has a different optimal pair of these two constants, which is why they are not fixed. In this case, optimal means that the constants should most accurately show the centralities and should most accurately show differences in scores.

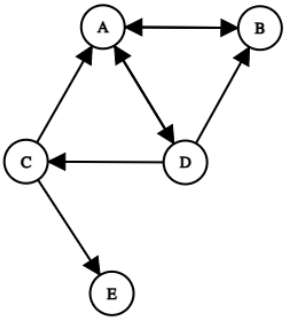
Page Rank Centrality

Page Rank centrality is unique because, unlike other centralities, it can be infinitely done for more accurate results. Instead of providing a score by counting, Page Rank instead calculates the probability of landing on a node. The equation for Page Rank centrality is:

$$C_p(v_i) = \alpha \sum_{j=1}^{n} \quad A_{j,i} \frac{C_p(v_j)}{d_j^{out}} + \beta$$

This equation may look complicated, but the calculations are reasonably straightforward. When iterating the calculations repeatedly (to be more accurate), as the number of iterations approaches infinity, the score of the does will also approach a certain number. This constant is modeled by the equation:

$$PR(K; t+1) = \frac{1-d}{N} + d \sum_{P \in \Gamma_K} \frac{PR(P; t)}{|\Gamma_P|}$$



<Figure 7>

Example 7: Calculations of the PageRank Centrality of **Figure 7**:

First, we can initialize the centralities: $C(A), C(B), C(C), C(D), C(E) = \frac{1}{5}$

In Iteration 1:

- $C(A) = 1 \cdot C(B) + \frac{1}{2} \cdot C(C) + \frac{1}{3} \cdot C(D) + 0 \cdot C(E) = \frac{11}{30}$
- $C(B) = \frac{1}{2} \cdot C(A) + 0 \cdot C(C) + \frac{1}{3} \cdot C(D) + 0 \cdot C(E) = \frac{1}{6}$
- $C(C) = 0 \cdot C(A) + 0 \cdot C(B) + \frac{1}{3} \cdot C(D) + 0 \cdot C(E) = \frac{1}{15}$
- $C(D) = \frac{1}{2} \cdot C(A) + 0 \cdot C(B) + 0 \cdot C(C) + 0 \cdot C(E) = \frac{1}{10}$
- $C(E) = 0 \cdot C(A) + 0 \cdot C(B) + \frac{1}{2} \cdot C(C) + 0 \cdot C(D) = \frac{1}{10}$

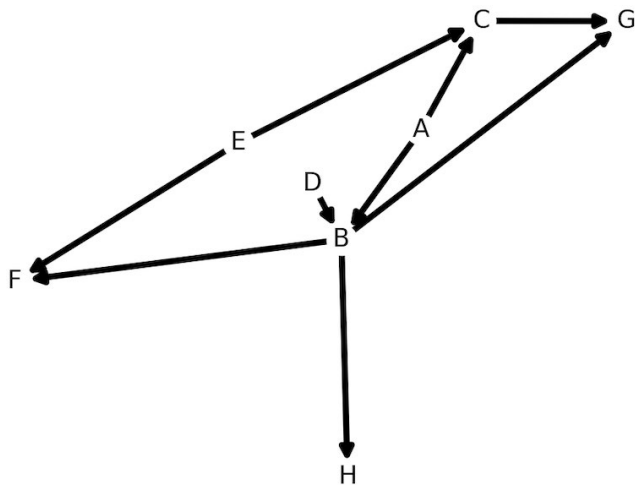
We can start Iteration 2 with $C(A) = \frac{11}{30}, C(B) = \frac{1}{6}, C(C) = \frac{1}{15}, C(D) = \frac{1}{10}, C(E) = \frac{1}{10}$:

- $C(A) = 1 \cdot C(B) + \frac{1}{2} \cdot C(C) + \frac{1}{3} \cdot C(D) + 0 \cdot C(E) = \frac{7}{30}$
- $C(B) = \frac{1}{2} \cdot C(A) + 0 \cdot C(C) + \frac{1}{3} \cdot C(D) + 0 \cdot C(E) = \frac{13}{60}$
- $C(C) = 0 \cdot C(A) + 0 \cdot C(B) + \frac{1}{3} \cdot C(D) + 0 \cdot C(E) = \frac{1}{30}$
- $C(D) = \frac{1}{2} \cdot C(A) + 0 \cdot C(B) + 0 \cdot C(C) + 0 \cdot C(E) = \frac{11}{60}$

- $C(E) = 0 \cdot C(A) + 0 \cdot C(B) + \frac{1}{2} \cdot C(C) + 0 \cdot C(D) = \frac{1}{30}$

When comparing the scores of iteration 1 and iteration 2, one can see that each centrality score has changed by a small amount. In this way, by continuing these iterations, these scores will continue to increase/decrease in infinitely smaller amounts until a specific constant is met.

Centrality Score Comparison



<Figure 8>

Centrality	Top 1	Top 2	Top 3
degree	**B** - 0.714	**C** - 0.429	**G** - 0.286
closeness	**G** - 0.446	**F** - 0.381	**C** - 0.286
betweenness	**B** - 0.131	**C** - 0.036	**G** - 0.000
Katz	**G** - 0.389	**F** - 0.383	**C** - 0.376
eigenvector	**G** - 0.816	**F** - 0.409	**H** - 0.408
PageRank	**G** - 0.229	**B** - 0.162	**F** - 0.147

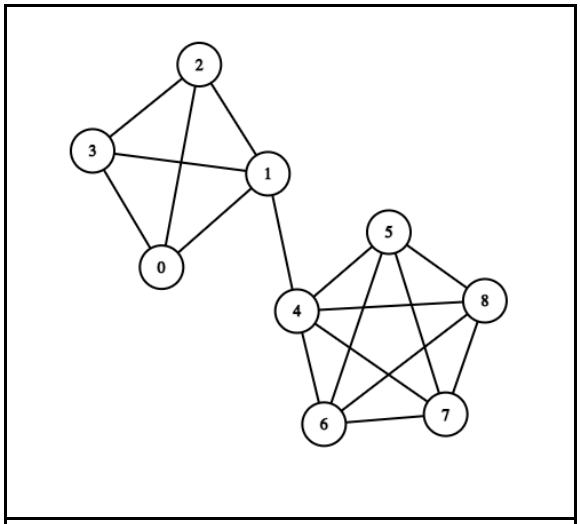
<Table 5>

Table 5 shows the top 3 node scores of each centrality of the graph in **Figure 8**. Only nodes B, C, G, F, and H were in the top 1, 2, or 3; specifically, G was either the top 1 node or the top 3 node, and B was either the top 1 node or the top 2 node. Most centralities reported G as the most efficient/most influential node, while C and F were either the top 2 nodes or the top 3 nodes. The closeness and Katz centralities reported identical scores – the eigenvector centrality also did, except the top 3 was node H, not node C. For disparity, the Katz centralities

had little disparity in the top 3; on the other hand, the degree centralities had lots of disparity in the top 3. An interesting finding is that node C in the closeness centrality, and node G in the degree centrality had the same score. Similarly, there was only a 0.001 difference in Nodes F and H scores in the eigenvector centrality.

Bridge Detection

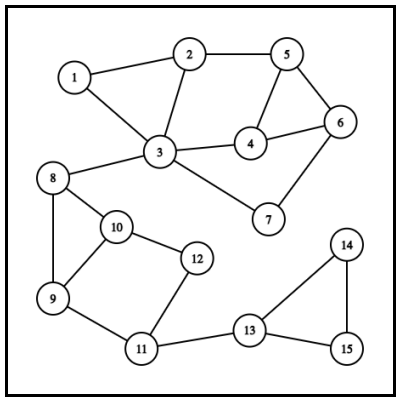
Bridges are important connections between certain clusters of nodes in a graph. When two graphs have no elements in common, they are called subsets. If the removal of a certain edge causes a graph to split into sub-sets, then that edge is considered a bridge.



<Figure 9>

Example 8: In Figure 9, edge 1-4 is considered a bridge because splitting it results in two subgraphs: 0-1-2-3 and 4-5-6-7-8.

It is essential to identify bridges to prevent the time complexity of a graph from growing to high amounts as the graph complexity of a graph rises exponentially. **Depth-first search** (DFS) is an algorithm used to find tree-like structures in graphs and find as many nodes along certain branches (as deep as possible) before **backtracking** is required.



<Figure 10>

Example 9: **Table 6** shows the discovery order of the 15 nodes of **Figure 10** using DFS.

Table 6.

Node	**Score**
1	6/1
2	5/1
3	4/1
4 (Source Node)	1/1
5	7/1
6	2/1
7	3/1
8	8/8
9	9/8
10	12/8
11	10/8
12	11/8
13	13/13
14	14/13
15	15/13

<Table 6>

As seen, there are only three different upper bounds. Namely, these are 1, 8, and 13. Therefore we can conclude that the nodes with certain upper bounds fit into subsets: a subset of nodes 1 to 7, a subset of nodes 8 to 12, and a subset of nodes 13-15. Now, we know that the edges connecting these subsets are bridges. Therefore, the bridges are the 3-8 edge and the 11-13 edge.

Community Detection

Communities are sets of dense groups of nodes with strong connections with other nodes in their communities but weak connections with other nodes. For example, many groups of friends exist on social media platforms that have strong bonds with one another in the friend group. These communities can also overlap, similar to how someone may be in multiple social circles.

Community detection is essential when analyzing a network or a graph. In graphs of immense scales, there may be millions of nodes and edges. Therefore, we try to identify significant communities to aid us in deciding which nodes are more influential than others. There are two primary community detection methods: **agglomerative methods** and **divisive methods**.

Agglomerative methods begin with a graph of nodes and without edges. Then, edges are gradually added to the graph from strongest to weakest. In **divisive methods**, the opposite is done. Edges are removed from a complete graph with the highest weight starting first.

Modularity

One issue with community detection is that there is no guaranteed way to find the most optimized partition of a graph into communities. Numerous algorithms have been developed that attempt to balance both accuracy and efficiency of graph splitting. The effectiveness of a partition is calculated in a measurement called modularity (also called a "score"), which is a constant ranging from -1 to 1. The formula of modularity (in a graph with weighted paths) is given by:

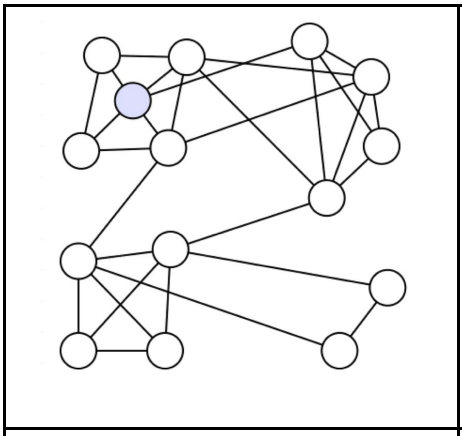
$$Q = \frac{1}{2m} \sum_{i,j} [A_{ij} - \frac{k_i k_j}{2m}] \, \delta(c_i, c_j)$$

$$A_{nm} \rightarrow \text{weight of edge between node } n \text{ and node } m$$
$$k_n \rightarrow \text{sum of weights of edges attached to node } n$$
$$c_n \rightarrow \text{community that node } n \text{ is in}$$
$$\delta(c_i, c_j) \quad \text{returns } 1 \text{ if } c_i = c_j, \text{and } 0 \text{ in any other case}$$

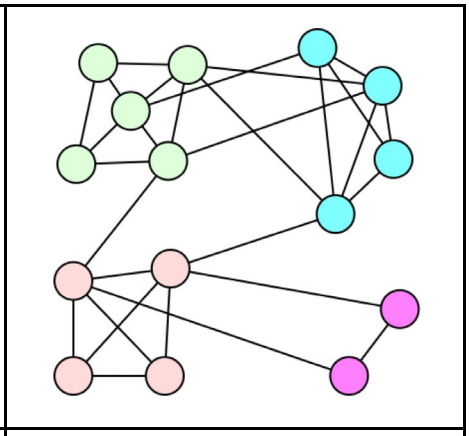
Finding Communities

The modularity function above can be utilized to create a graph with optimal partitions. There are three phases to detect community:

Phase 1: Partition



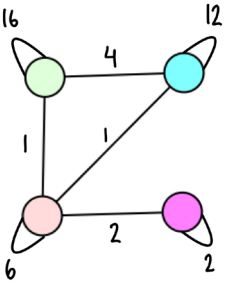
<Figure 11>



<Figure 12>

In **Figure 11**, a source node is chosen to begin the partitioning process. The communities partitioned using the score function are shown in **Figure 12**.

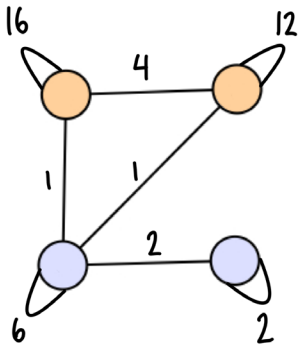
Phase 2: Aggregation

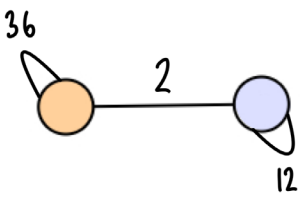


<Figure 13>

In **Figure 13**, the communities are aggregated, and each community is represented by one node. Node and edge weights are given by the number of connections within a community and from one community to another.

Phase 3: Repetition





<Figure 14>	*<Figure 15>*

The method is done several times until the graph cannot be split into any more communities (in this case, a graph with two nodes in **Figure 15**). The modularity of the partition can then be calculated feasibly. The complexity of a graph can also be seen by the number of times a "pass" (perform partition and aggregation) must be done.

Results

The two methods I have chosen balance time complexity and accuracy well: **Closeness centrality with community detection** and **Katz centrality with bridge detection.** Under "Centrality Score Comparison," we can see that nodes G, F, and C were chosen to be the most influential three nodes (in that order) by both closeness centrality and Katz centrality. These two methods also had the lowest time complexities in **Figure 19**.

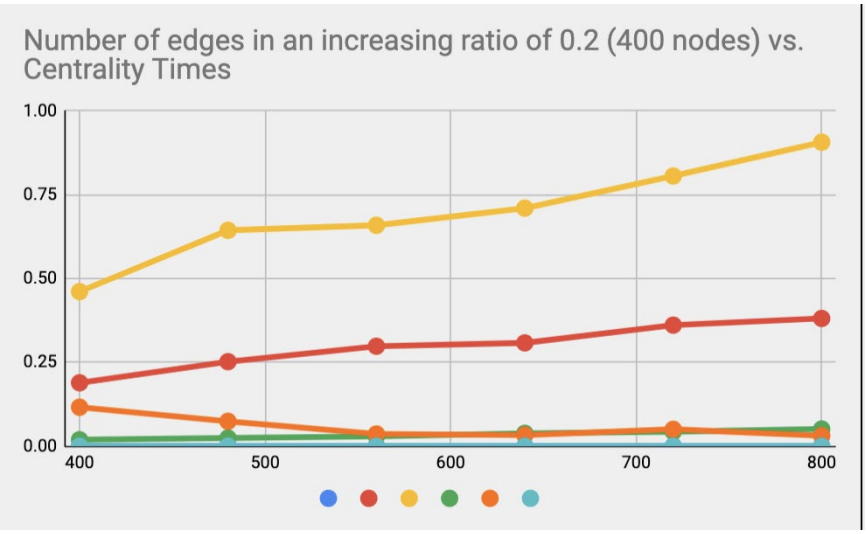
Time Complexity of Centralities

Density

Node	Edge	Ratio	Degree	Close	Betw.	Katz	Eig.
400	400	1:1	0.0001	0.189	0.461	0.019	0.116
400	480	1:1.2	0.0002	0.252	0.644	0.024	0.074
400	560	1:1.4	0.0002	0.298	0.659	0.029	0.036
400	640	**1:1.6**	0.0001	0.308	0.710	0.038	0.033
400	720	1:1.8	0.0002	0.361	0.806	0.043	0.050
400	800	1:2	0.0003	0.381	0.907	0.051	0.031

<Table 7>

I wanted to test the 1:1.6 ratio in the evaluation because it had the least time-to-ratio proportion (as shown in **Table 7**), which meant it was most efficient in the first test. I managed to achieve similar results in the second experiment.



Blue = Degree, **Red** = Close, **Yellow** = Between, **Green** = Katz, **Orange** = Eig

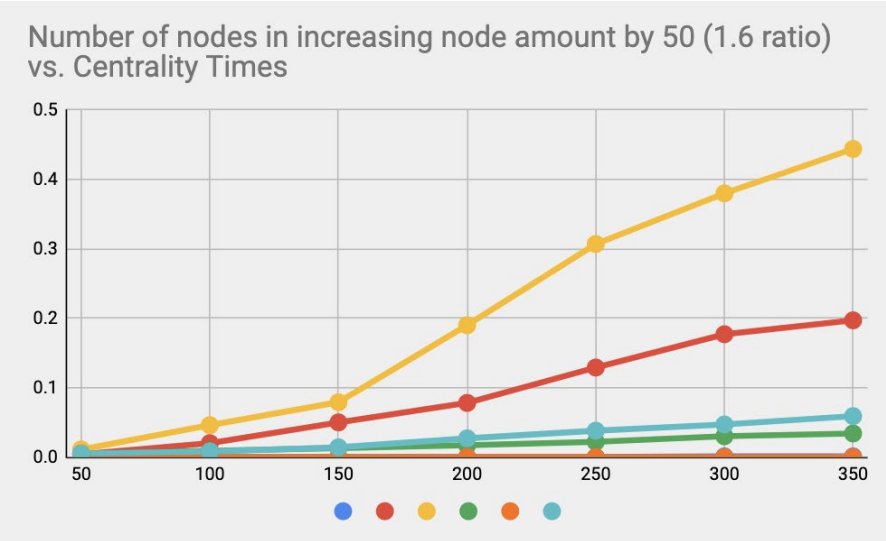
<Figure 17>

Figure 17 shows the graph of the time complexities of graphs with a constant number of nodes and an increasing ratio of nodes-to-edges. Therefore, this experiment is based on increasing density. The closeness and betweenness centralities increased the most as the density of the graph increased. The degree and Page Rank centralities stayed constant through the increasing density. The Katz centrality increased by a small amount. Interestingly, the eigenvector centrality first decreased, then gradually increased at the same rate as the Katz centrality.

Node

Node	Edge	Ratio	Degree	Close	Betw.	Katz	Eig.
50	80	1.6	0.000	0.004	0.011	0.005	0.005
100	160	1.6	0.000	0.020	0.046	0.009	0.008
150	240	1.6	0.000	0.050	0.079	0.013	0.014
200	320	1.6	0.000	0.078	0.190	0.017	0.027
250	400	1.6	0.000	0.129	0.307	0.022	0.038
300	480	1.6	0.001	0.177	0.380	0.030	0.047
350	560	1.6	0.001	0.197	0.444	0.034	0.059

<Table 8>

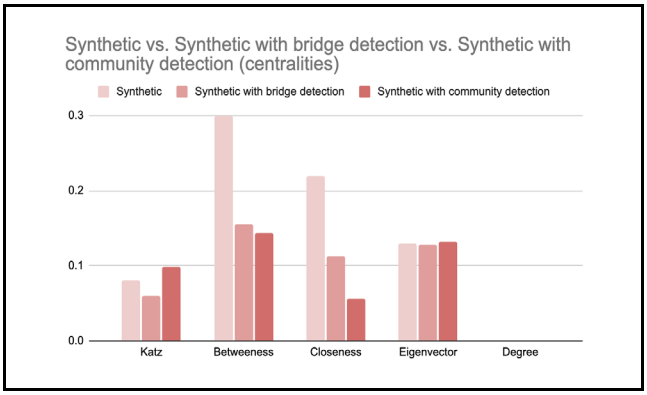


Blue = Degree, **Red** = Close, **Yellow** = Between, **Green** = Katz, **Orange** = Eig

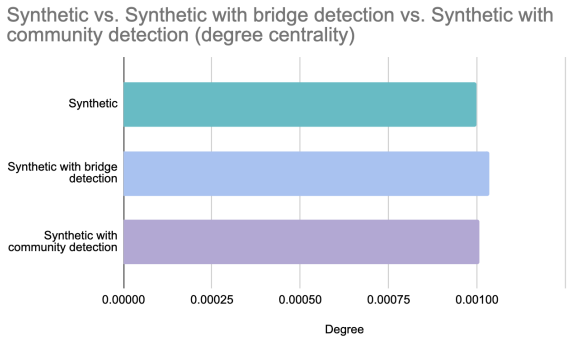
<Figure 18>

Figure 18 shows the graph of the time complexities of graphs with constant node-to-edge ratios (date from **Table 8**). Therefore, this experiment is based on the number of nodes in a graph. The duration of the betweenness and closeness centralities increase rapidly. Hence, these centralities can be said to have exponentially increasing time complexities. The Page Rank centrality is linear. The eigenvector centrality is constant, and while the Katz centrality appears to have taken longer than the eigenvector centrality, this is negligible as the Katz centrality is merely a variation of the eigenvector centrality. Therefore, we can still say the Katz centrality is constant. Lastly, although the degree centrality looks to be constant, it increases in a very slight amount. Compare the time complexity of the degree centralities of two graphs: one of 2 nodes, and one of 1000 nodes. By intuition, one can know that the graph with 1000 nodes will take longer to calculate its degree centrality. So the degree centrality time complexity is linear nonetheless.

Time Complexities of Bridge and Community Detection



<Figure 19>



<Figure 20>

Figure 19 shows a comparison between the time complexities of node centralities measured with a synthetic graph, a synthetic graph with bridge detection, and a synthetic graph with community detection. **Figure 20** shows a closer look at the time complexities of the degree centrality, which is too small to be seen effectively in **Figure 19**.

Discussion

In the Katz centrality, bridge detection served moderately useful compared to the synthetic time, but the time complexity of the community detection graph clearly worsened. This can be explained because Katz centrality aims to count the number of paths that must be taken between two nodes to find their centralities. Bridge detection helps to count the number of "walks" it takes because it counts paths in the process of finding bridges. On the other hand, Katz centrality does not require community detection, and therefore, its time complexity increases. In the betweenness and closeness centralities, distances are being compared. Bridge detection and community detection allow a graph to be split into clusters, and thus, it is easier to find the closest paths. Usually, the time complexities become exponential (they multiply) as we check each node's distance, but with these methods, only each subgraph has to be checked to result in a much smaller time complexity (because we only have to add the subgraph time complexities). The time complexities were all nearly identical for the eigenvector centrality because eigenvector centrality is found using adjacency matrices, which do not need bridges or communities for them to be created. The only thing that must be counted for degree centrality is the number of edges connected to a node. Bridges and communities are not required for this task, so they merely hinder the time complexity. This is why the bridge detection and community detection time complexities for degree centrality are higher than the synthetic graph without any advanced method.

Conclusion

Graphs are very effective tools for engineering analysis of the real world. In this paper, the centrality score was selected to measure the influence of each node in the graph. As the method of calculating this score becomes more complex, accuracy is guaranteed, but time complexity increases simultaneously. In particular, when the score is a case where the relationship between nodes is important, the time complexity increases. Graphs representing the real world have a lot of nodes and edges in many cases, and experiments have found that if applied

as they are, the time efficiency will be extremely low. To compensate for this point, bridge detection and community detection, a method of dividing a large graph into several subgraphs at a low level, were applied to change the nested loop operation to a simple sum of operations in series. Furthermore, a model, which is the most appropriate combination, was proposed and experimentally proved in consideration of trade-offs. The reason for selecting the ratio of node and edge numbers to increase the experiment's credibility was also described.

In this study, synthetic data was generated and experimented with, but more realistic experimental results can be obtained because random characteristics become stronger when tested with actual data. Since this leads to a better methodology proposal, my future work is to experiment with real data and propose a more appropriate model.

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

[1] Zhao, X., Guo, S., & Wang, Y. (2017, July). The node influence analysis in social networks based on structural holes and degree centrality. In 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC) (Vol. 1, pp. 708-711). IEEE.

[2] Lv, L., Zhang, K., Zhang, T., Bardou, D., Zhang, J., & Cai, Y. (2019). PageRank centrality for temporal networks. Physics Letters A, 383(12), 1215–1222. https://doi.org/10.1016/j.physleta.2019.01.041

[3] Cohen, E., Delling, D., Pajor, T., & Werneck, R. F. (2014). Computing classic closeness centrality, at scale. in Proceedings of the Second ACM Conference on Online Social Networks, ser. COSN '14. ACM, 37-50.

[4] Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness. Sociometry, 40(1), 35. https://doi.org/10.2307/3033543

[5] Gao, S., Ma, J., Chen, Z., Wang, G., & Xing, C. (2014). Ranking the spreading ability of nodes in complex networks based on local structure. Physica A: Statistical Mechanics and Its Applications, 403, 130–147. https://doi.org/10.1016/j.physa.2014.02.032

[6] Sabidussi, G. (1966). The centrality index of a graph. Psychometrika, 31(4), 581–603. https://doi.org/10.1007/bf02289527

[7] Freeman, L. C. (1978). Centrality in social networks conceptual clarification. Social Networks, 1(3), 215–239. https://doi.org/10.1016/0378-8733(78)90021-7

[8] Corradini, E., Nocera, A., Ursino, D., & Virgili, L. (2020). Defining and detecting k-bridges in a social network: The Yelp case, and more. Knowledge-Based Systems, 195, 105721. https://doi.org/10.1016/j.knosys.2020.105721

[9] Ghosh, S., Halappanavar, M., Tumeo, A., Kalyanaraman, A., Lu, H., Chavarria-Miranda, D., Khan, A., & Gebremedhin, A. (2018). Distributed Louvain Algorithm for Graph Community Detection. 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS). https://doi.org/10.1109/ipdps.2018.00098

[10] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank Citation Ranking: Bringing Order to the Web. - Stanford InfoLab Publication Server. Stanford.edu. https://doi.org/http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

[11] Cadini, F., Zio, E., & Petrescu, C. A. (2008, October). Using centrality measures to rank the importance of the components of a complex network infrastructure. In International Workshop on Critical Information Infrastructures Security (pp. 155-167). Springer, Berlin, Heidelberg.