# Application of Group Testing in Identifying High Betweenness Centrality Vertices in Complex Networks

Vladimir Ufimtsev
Computer Science Department
University of Nebraska at Omaha
vufimtsev@unomaha.edu

Sanjukta Bhowmick
Computer Science Department
University of Nebraska at Omaha
sbhowmick@unomaha.edu

## ABSTRACT

Group testing is a mathematical technique that uses superimposed code theory to find a specified number of distinct units among a large population, using the fewest number of tests. In this paper, we investigate the applicability of group testing in finding vertices with high betweenness centrality. Betweenness centrality (BC) is a widely applied network analysis objective, for identifying important vertices in complex networks. Most algorithms for computing BC compute the values for all the vertices in the network. However, in practice, only the vertices with the highest BC values are used in analyzing the network, and even then we only need the identities of the vertices–not the exact values.

We demonstrate that Latin square based group testing is effective in finding the top two highest BC nodes of most networks. We also show that the instances where group testing fails to obtain the top BC nodes are networks where slight perturbation of the edges can change the ranking of the vertices. An additional benefit of group testing is that it allows us to decompose the betweenness centrality computation into a trivially parallelizable algorithm with high scalability.

## Keywords

betweenness centrality, group testing, network perturbation

## 1. INTRODUCTION

Many systems of interacting entities, such as those appearing in biology [26], epidemiology [6] and social sciences [2], can be modeled as networks. In a network model, the entities are represented by vertices and the interaction between each pair of entities as edges. Analysis of the properties of networks can help us understand the characteristics of the underlying system.

One important analytical property of networks is the *betweenness centrality* of the vertices. Betweenness centrality (BC) measures the importance (centrality) of a vertex with respect to the flow of information in a network, based on the number of shortest paths that pass through that vertex [3], [4, 5, 15, 18]. The popular Brandes method [5] for obtaining BC, cumulatively computes the values for *every vertex* in the network. This method has a complexity of $O(|V| \cdot |E|)$ for unweighted networks, where $|V|$ is the number of vertices and $|E|$ is the number of edges. Although the algorithm has polynomial complexity, the execution time is still prohibitive for large-scale networks.

However, most applications of betweenness centrality, such as the Girvan-Newman community detection method using the divisive method [18], require only the top few vertices in the network with *high* BC. Moreover, the BC values for most real-world networks, with strong community structure, show an exponential decay. In most cases, only a small fraction of vertices have distinctive high BC values, and the values then quickly fall with a bulk of vertices having zero betweenness centrality.

Based on these observations we posit that the analysis algorithm should focus on identifying only the top-highest BC vertices in the network, and it is the *identity not the actual BC value* of the vertex that is important. We present a group testing based method that identifies the highest BC vertices in a network, without explicitly computing their individual values. We study how well group testing is suited for finding high BC vertices in complex networks, along with solutions to deal with the issue of interacting entities. Our experiments demonstrate that group testing is most successful on networks that are stable under edge perturbations, i.e. they retain the same ranking of high BC vertices. We also show that group testing can be translated into an easily parallelizable algorithm requiring minimum communication.

Group testing is a mathematical technique employed in the design of screening experiments [9] to identify items that have a given special characteristic from a population of items. It originated during World War II for efficiently testing potentially infected blood samples of millions of draftees. Since then, group testing has been used in many applications[21, 22] including finding counterfeit coins, patterns in data and DNA library screening. The central idea of group testing is that if there is a small percentage of defective units in the population, it is more efficient (requires less tests) to test the units in carefully selected groups, rather than testing each unit separately. For example, if the total population is $n$, using group testing, two defective units can be identified in $O(\sqrt{n})$ tests.

In our approach, the defective units correspond to vertices with high betweenness centrality. We group samples of vertices and approximate the betweenness centrality of each


*Eleventh Workshop on Mining and Learning with Graphs.* Chicago, Illinois, USA

sample to find which groups have BC values higher than a given threshold. These groups contain vertices with high betweenness centrality values. We apply a Latin Square-based strategy for composing the groups. This method guarantees that we find at least two defective units in $3\lceil\sqrt{n}\rceil$ tests from a sample of $n$ units.

To date, there has been very limited implementation of group testing in graph theory. Examples include finding broken links in optical networks [19] and congested links in wireless sensor networks (WSNs) [7].To the best of our knowledge, this is the first application of group testing in identifying sensitive vertices in complex networks. Our previous work on this topic and preliminary results can be found in [25]. There is however previous work on identifying the vertices in a network with the highest closeness centrality, a different centrality measure. For example in [23] ranking and approximation algorithms are used to obtain the ranking of the $k$ highest closeness centrality vertices in a network. These methods widely differ from our group testing approach though the end goal is very similar to ours i.e. identifying important vertices in a network.

# 2. BACKGROUND

## 2.1 Terminology for Network Analysis

A network (or graph) $G = (V, E)$ is defined as a set of vertices $V$ and a set of edges $E$. An edge $e \in E$ is associated with two vertices $u, v$ which are called its *endpoints*. A vertex $u$ is a *neighbor* of $v$ if they are joined by an edge. A graph is undirected if $(u, v) \in E$ also implies that $(v, u) \in E$ and is unweighted if there are no values associated with the edges, that is all edges have the same importance. In this paper we will be dealing only with undirected and unweighted networks.

A *path*, of length $l$, in a graph $G$ is an alternating sequence of $v_0, e_1, v_1, e_2, \ldots, e_l, v_l$ vertices and edges, such that for $j = 1, \ldots, l$; $v_{j-1}$ and $v_j$ are the endpoints of edge $e_j$, with no edges or internal vertices repeated.

The centrality of a vertex is a parameter which specifies the importance of the vertex in a network. The betweenness centrality of vertex $v$ is defined as [15]: $BC(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where $\sigma_{st}$ is the total number of shortest paths in $G$ between nodes $s$ and $t$, and $\sigma_{st}(v)$ is the total number of shortest paths in $G$ between $s$ and $t$ that pass through $v$. The most commonly used algorithm for computing betweenness centrality is the Brandes algorithm [5], which cumulatively computes the BC of *every* vertex while traversing the entire network. The Brandes algorithm has complexity $O(|V| \cdot |E|)$ on unweighted networks. Faster algorithms include methods for approximating BC scores [8, 17] and sampling to obtain the BC of a single vertex [3]. Parallel algorithms for computing BC algorithms include [4, 14]. All these methods still focus on computing the exact or nearly exact BC value of all the vertices.

## 2.2 Group Testing

Group testing [9] is a mathematical technique to find a specified number of defective units among a large population of units using the fewest number of tests. Given a population of $n$ entities, units are termed as "defective" if they have a characteristic that is not present in the the other "non-defective" units. In group testing, the items of the population are combined in carefully selected groups. For a given group and a given threshold, the *presence* or *absence* of the defective characteristic can be established by *exactly one test*. If a defective unit is present in the group then the result of the test is said to be *positive* (1), otherwise it is a negative (0) result. After $N$ tests on sufficient number of groups, we can exactly identify the defective units. An important research topic in group testing is designing algorithms to select the composition of the groups, such that the fewest number of tests are required to find the defective units. Superimposed code theory is used to design efficient composition of groups.

## 2.3 Superimposed Code

A superimposed code is a set of binary vectors such that no vector in the set can be covered by a Boolean-OR of a certain number of other vectors in the set. A superimposed code of length (number of elements in the set) $N$ and size (size of the vectors) $n$ can be represented as a binary $N \times n$ matrix, $\mathbf{X}$. Let $x_{i,j} \in \{0, 1\}$ denote the element in row $i$ and column $j$ of $\mathbf{X}$ and let $\mathbf{x_j}$ denote the $j^{th}$ *column* of $\mathbf{X}$. The Boolean-OR sum of any $k$ columns $\mathbf{x_{j_1}}, \mathbf{x_{j_2}}, ..., \mathbf{x_{j_k}}$ is;

$$f(\mathbf{x_{j_1}}, \mathbf{x_{j_2}}, ..., \mathbf{x_{j_k}}) = \begin{bmatrix} x_{1,j_1} \vee x_{1,j_2} \vee \ldots \vee x_{1,j_k} \\ x_{2,j_1} \vee x_{2,j_2} \vee \ldots \vee x_{2,j_k} \\ \vdots \\ x_{N,j_1} \vee x_{N,j_2} \vee \ldots \vee x_{N,j_k} \end{bmatrix}$$

where $\vee$ is the Boolean-OR operation i.e. $0 \vee 0 = 0, 0 \vee 1 = 1, 1 \vee 0 = 1, 1 \vee 1 = 1$.

A column $\mathbf{x_j}$ *covers* column $\mathbf{x_i}$ if $f(\mathbf{x_j}, \mathbf{x_i}) = \mathbf{x_j}$. Code $\mathbf{X}$ has strength $d$ if and only if the Boolean-OR sum of any $d$ columns does not cover any other column [11]. Superimposed code theory can be applied to group testing as follows;

Given a population $n$, we can construct $N$ tests each represented by binary vectors of length $n$. The vector $\mathbf{x_i}$ is created such that, $\mathbf{x_{i,j}} = \mathbf{1}$ if the $j^{th}$ element is included in the group for the $i^{th}$ test. Based on this construction it is easy to see that the groups and the tests can be represented as a superimposed code. If the strength of the code is $d$, then at most $d$ defective units can be identified in $N$ tests. The goal is to select the binary vectors to maximize $d$.

The *weight*, $w(\mathbf{x_j})$, of column $\mathbf{x_j}$ is the number of ones in the column. The *minimum* weight $w = \min_{1 \leq j \leq n} w(\mathbf{x_j})$. The *intersection*, $\lambda(\mathbf{x_j}, \mathbf{x_i})$, between two columns $\mathbf{x_j}, \mathbf{x_i}$ is the number of positions in which both $\mathbf{x_j}$ and $\mathbf{x_i}$ have a 1. The *maximum* intersection $\lambda = \max_{1 \leq i \neq j \leq n} \lambda(\mathbf{x_j}, \mathbf{x_i})$. The Kautz-Singleton Bound [20] states that $d \geq \lfloor \frac{w-1}{\lambda} \rfloor$; that is, just by knowing the minimum weight $w$ and maximum intersection $\lambda$ we are able to obtain a lower bound on the strength $d$ of the code. It has been shown by D'yachkov and Rykov ([11],[12],[13]) that as $n \to \infty$ and $d \to \infty$ with $d \leq \log_2 n$, the *minimum* number of tests $N$ is bounded by: $\Omega\left(\frac{d^2}{\log_2 d} \log_2 n\right) \leq N \leq O\left(d^2 \log_2 \frac{n}{d}\right)$.

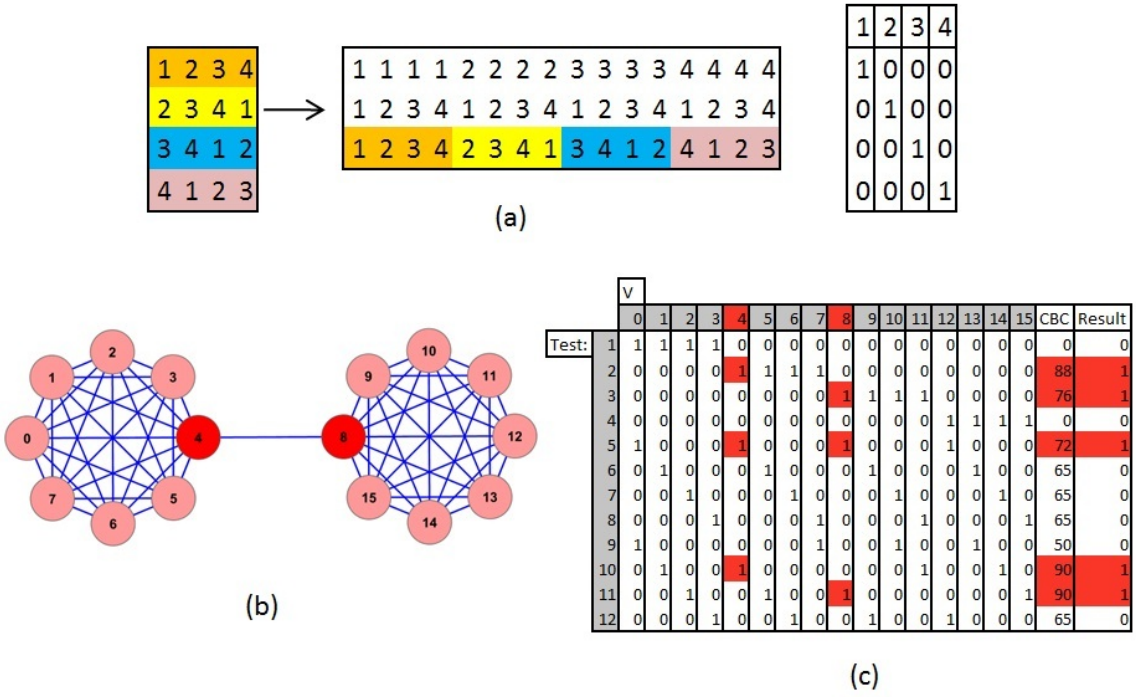


**Figure 1: Example of Group Testing Using Latin Squares. (a): Construction of coding matrix using a 4 by 4 Latin square. (b): A sample graph of two-8-cliques connected by one edge. The final matrix given in (c), allows at most 16 units to be tested.The threshold is set to 65. Clearly vertices 4 and 8 are the ones with highest BC, as given by the Results column in (c).**

## 3. GROUP TESTING FOR IDENTIFYING HIGH BETWEENNESS CENTRALITY

We now present our main contribution on how group testing can be applied to find high betweenness centrality vertices in a network. The central idea in our method is that when high BC vertices are part of a group of vertices ( henceforth referred to as the supervertex) then the BC of the supervertex should also be high. We combine the selected vertices in a group as follows; given a network $G = (V, E)$, for each test, the group of selected vertices are combined into one "supervertex" whose set of neighbors is the union of the sets of neighbors of its constituent vertices. For example, assume that in test $i$ we have the set of vertices to be grouped and tested: $T_i = \{v_1, v_2, ..., v_g\} \subset V$. Let $v_{T_i}$ denote the supervertex made by combining all vertices in $T_i$ and let $N_{v_j} = \{u \in V : (v_j, u) \in E\}$ be the set of neighboring vertices for vertex $v_j$. Then the set of neighboring vertices of the supervertex corresponding to $T_i$ is simply $N_{T_i} = \{u \in V : \exists v_j \in T_i, (v_j, u) \in E\} = \bigcup_{j=1}^{g} N_j$.

### 3.1 Latin Square Based Group Testing

In our experiments, we used a group testing strategy based on Latin Squares, which is guaranteed to find at least 2 defective units. The superimposed code $X$ is created as follows: given a population of $n$ units, create finite set of contiguous integers $\{1, 2, ..., l\}$, where $l = \lceil \sqrt{n} \rceil$. Then create a $l \times l$ Latin square is a matrix $A$, where $A_{i,j} \in \{1, 2, ..., l\}$ such that each element from $\{1, 2, ..., l\}$ appears exactly once in any given row and column (left diagram in Figure 1(a)).

We construct a coding matrix $\mathbf{X}$ from a Latin square $\mathbf{L}$ in the following way; the first 2 positions in any given column in $\mathbf{X}$ are coordinates (row and column) in $\mathbf{L}$ and the 3rd position is the element in $\mathbf{L}$ at those coordinates (middle diagram in Figure 1(a)). We then encode each integer in $\mathbf{X}$ in its binary form. Each integer is coded as a binary vector of length $l$, where for integer $i$, the vector has zero in all positions, except at position $i$ which has one. In other words, the binary representation of integer $i$ is the $i^{th}$ row (or column) of an $l \times l$ identity matrix (right hand diagram in Figure 1(a)). Once the elements of the coding matrix are transformed to their binary form, the total number of tests is equal to the number of rows in the $\mathbf{X}$ matrix, which is $3\lceil \sqrt{n} \rceil$, for a population of $n$ units (Figure 1(c)).

Due to the Latin square construction, any two columns in $\mathbf{X}$ can intersect in *at most* one position. Defective units are those whose value is more than the user selected threshold. We find the minimum weight $w = 2$ and maximum intersection $\lambda = 1$ then use the Kautz-Singleton Bound to get a guaranteed value for the strength parameter $d \geq \lfloor \frac{w-1}{\lambda} \rfloor = \lfloor \frac{2}{1} \rfloor = 2$. Although ideally in non-interactive cases the final results of the Latin-Squares method should not vary, the composition of the groups can change according to how the vertices are ordered.

An example of how Latin square group testing can be used on a small network is given in Figure 1. The network consists of 16 vertices, composed of two cliques of 8 vertices each. The Latin Square is therefore created for a 4 by 4 matrix by right shifting the numbers {1,2,3,4} in each row. The superimposed code has 16 columns and 12(3*4 rows),

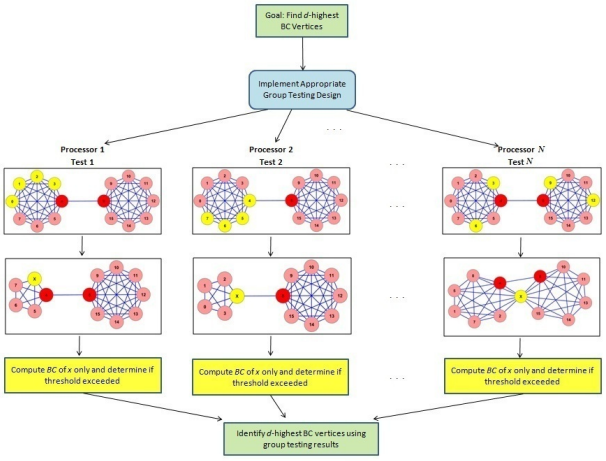


**Figure 2: Parallel Group Testing For Finding High BC vertices. This figure shows the parallel implementation of our group testing algorithm. Each group is sent to a different processor which calculates the value of the associated test.**

where the first row gives the row in the Latin Square, the second row gives the column and the third row gives the value. This matrix is then expanded to its binary form. Each row denotes a test, presence of a 1 indicates that a vertex will be included in the test. For example, test 2 contains only the vertices {4,5,6,7}, which are combined as a supervertex. The threshold is set to 65 and all tests where the BC value of the supervertex is higher than 65 are marked as positive (colored red). The result vector has 1 for positive tests and 0 for negative ones. Note that the Boolean-OR of the columns 4 and 8 is exactly the same as the resultant vector. Therefore vertices 4 and 8 are the high BC vertices.

## 3.2 Implementation Details.

As described above, group testing for identifying high BC vertices consists of three steps. The first step is to generate the coding matrix based on Latin Square. We do not create the Latin square or the coding matrix explicitly. Based on the number of vertices, we can then determine the positions of the ones at each row, hence the group of vertices for each test.

The second step is to compute the BC values for the group of vertices per test. We add a new vertex to the network that is connected to the neighbors of all the vertices in the group and compute the BC value of this supervertex.

In the third step, once the BC values of the supervertex from all tests are collected, we set the threshold to be $\tau$% of the highest value. If the BC value of a test is equal to or higher than the threshold it is positive, otherwise it is negative. Again, we do not explicitly compute the boolean-OR of the columns of the coding matrix. Instead, we initially mark all vertices to be in the high ranking set. If a vertex belongs to a group that tests negative, we remove it from the high ranking set. At the end of this process only the vertices whose groups always tested positive remain. These are identified as the high ranking vertices. The pseudo code for the algorithm is given in Algorithm 1.

*Parallel Algorithm.* This process of identifying high BC vertices can be easily parallelized. We implemented a MPI-based master-worker parallelization scheme. The master processor determines the groups of vertices per row (test), and sends each test to a worker. Each worker processor retains a copy of the network, and based on the group obtained from the master, creates the new network and then computes the BC value. The BC value and the associated group is returned to the master. Once all the BC values are received, the master then performs the final step of identifying the high BC nodes. A schematic diagram of this process is given in Figure 2. Note that the parallelization is on the group testing process and *not* the calculation of BC. Each processor has a group assigned to it and it calculates the BC of that group sequentially.

**Data**: Network $C$, threshold $\tau$
**Result**: High BC vertices from $C$
$n$ = number of vertices in $C$;
$T = 3\lceil\sqrt{n}\rceil$ (number of tests);
$Result[T]$ = vector of length $T$ ;
Create $G$, a binary $3\lceil\sqrt{n}\rceil \times n$ group testing matrix based on a $\lceil\sqrt{n}\rceil \times \lceil\sqrt{n}\rceil$ Latin square;
**for** $i = 1$ **to** $T$ **do**
  Group vertices indicated by row $i$ of matrix $G$ into supervertex $v_i$;
  Compute BC value of $v_i$;
  $Result[i] = BC(v_i)$;
**end**
MAX $= max_{1\le i\le T}(Result[i])$;
**for** $i = 1$ **to** $T$ **do**
  **if** $Result[i] \ge \tau MAX$ **then**
    $Result[i] = 1$;
  **else**
    $Result[i] = 0$;
  **end**
**end**
Use binary *Result* vector to determine which columns of $G$ are covered by it and output the vertices corresponding to the covered columns;
**Algorithm 1:** Group Testing for High BC Algorithm

*Network Perturbation.* Networks collected from real-world applications inherently contain some noise, i.e. false positive edges that are added, but should not be and false negative or missing edges. We perturb the networks to see whether the ranking of the high BC vertices is maintained under perturbation. We used the Erdős-Rényi random graph based perturbation model developed in [1]. In this model, for a given parameter $\epsilon$, $0 \le \epsilon \le |V|$, an edge that is present in the original network has a probability of $\frac{\epsilon}{|V|}$ of being removed, and an edge that is not part of the original network has a probability of $\frac{\epsilon}{|V|}$ of being added. Note that if the network is fairly sparse then the perturbed network will have more edges added than removed which will result in a network with more edges than the original.

Using this model we analyze the effects perturbations have on the ranking of the top 10 BC vertices. We first compute the BC of every vertex in each network, using the exact Brandes algorithm and find the top 10 ranked vertices. Then we run the perturbation model on each network at various values of $\epsilon$ to obtain new networks for which we again find the top 10 BC vertices. We then compare the new set with the original set and measure their similarity using the Jaccard index. Given two sets $A$ and $B$, the Jaccard index is

defined to be: $JI(A,B) = \frac{|A \cap B|}{|A \cup B|}$ where $|A \cap B|$ is the size of their intersection and $|A \cup B|$ is the size of their union. For two sets that are identical, the Jaccard index is 1 and for two disjoint sets the value is 0. Thus the closer the index is to 1 the lower the effect of the perturbation.

We have observed that group testing is most effective in finding the high BC values when the vertex rankings are maintained under perturbation. If the vertex rankings alter this indicates that the high ranked vertices are not significantly higher than their competitors and this leads to false positives in the group testing method as discussed in the next section.

## 3.3 Issues in Group Testing on Networks

Although group testing seems a natural match for finding important or sensitive vertices in a network, there exist several issues (some related to group testing itself, and some due to special properties of the network) that can affect the accuracy and performance of the method. Some of the issues and our solution to them are discussed below.

*False Positive Results Due to Interaction Among Vertices:* Group testing was originally designed for non-interacting samples–that is samples that do not combine to change their characteristics. For example, two non-infected blood samples when combined, do not create an infected sample. But this is not the case for vertices in a network. Two (or more) vertices with low to medium BC values can together combine in a sample to create a group with very high BC.

We have observed two types of false positives; in the first type, the vertices with very low betweenness centrality can be falsely identified if due to their placement in the superimposed code, they are always in the same group with at least one very high BC vertex. These false positives can be eliminated by permuting the vertices in the graph to create a new set of groups. Alternatively, we can use known topological characteristics, for example vertices with clustering coefficient 1 have 0 BC, to cull out some of these false positives.

The second type is when groups of vertices with medium BC values combine together to give positive results. There is a much lower chance of this happening because the positive results also have to cover the appropriate columns of the vertices. Therefore if both high and medium BC vertices get covered then there would be an avalanche of positively marked vertices which would signal the presence of many false positives. In our experiments, we select thresholds to restrict the number of positive results to around 7 vertices.

There are also cases where only the vertices with medium BC values show up as positive, their group BC outranking the vertices with individually high BCs. This happens when the original high BC vertices are by themselves not high enough to counterbalance the cumulative effect of mid-ranked vertices. Our method cannot distinguish false positives in this case. However, we have also observed that in these problem instances, the rankings of the high BC vertices change under small perturbations to the network. Thus, for these problems, the ranking is not very stable, and identifying ranking the vertices by their BC values would be more academic than utilitarian.

*Selection of Threshold:* The threshold value for a given group testing design determines which tests are classified as positive (above threshold and thus contain high defective units) and which are negative (below threshold). Selection of correct threshold is a problem inherent to group testing itself, and even more important when applied to finding high betweenness centrality vertices, due to the probability of achieving false positive results.

In our experiments, we start by selecting the threshold to be 85% of the highest BC value obtained by group testing and keep on decreasing the threshold by units of 5% until we obtain enough positive results to cover a column corresponding to a vertex. This strategy is currently based on trial and error and we are working on a theoretical method to select the threshold a priori. The Latin Square method guarantees two defective units and we have seen that by slightly decreasing the threshold we can obtain 3-4 high BC vertices. We therefore keep on decreasing the threshold until about 7 columns are covered. Some of these columns are false positives, but they can be eliminated using the techniques described above.

The value of the threshold does not affect the complexity of our algorithm, only the final results. Thus, even if we select a very low threshold, the runtime will be the same but the final result will have too many identified vertices (many false positives).

# 4. EXPERIMENTAL RESULTS

We now present empirical results demonstrating how group testing can be used to identify high BC vertices. Our experiments were performed over a set of ten networks collected from the DIMACS Implementation Challenge Set [10] and the Stanford Network Analysis Project [24]. The networks we used are:
(i) Karate, a social network of friendships between 34 members of a karate club,
(ii) Chesapeake, the ecosystem network in the Chesapeake Bay,
(iii, iv) AS20000101 and AS20000102, communication networks, two instances of an autonomous system comprised of internet routers,
(v) Caida, autonomous systems network,
(vi) C. Elegans, the metabolic network of *C. elegans* species,
(vii) LesMis , coappearance network of characters in the same chapter of the novel Les Miserables,
(viii) GrQc, collaboration network of authors submitting to Arxiv in the general relativity and quantum cosmology category,
(ix) HepTh, collaboration network of authors submitting to Arxiv in the high energy physics theory category,
(x) Power Grid, network representing the topology of the Western States Power Grid of the United States.

*Accuracy of Results.* In order to evaluate the accuracy, we compare how many of the nodes identified to be high ranking using group testing also have high rank when the exact BC values are computed. We deem the group testing method successful if group testing is successful in correctly identifying the top 2 vertices. The results are given in Table 1. Out of the ten networks, group testing was successful in six networks (top six rows of the table), and found low ranked (below rank 10) vertices for the other four (the last four rows of the table).

We further perturbed the networks using 8 different $\epsilon$ values ranging from 0.05 to 2.5. We computed the BC values of the vertices using the exact Brandes algorithm. We calculated the Jaccard index between the top ten BC vertices

**Table 1: Finding High BC Vertices Using Group Testing on Real-World Networks. The best threshold and the vertices obtained using that threshold are given. The vertices are represented by their rank, as per their BC values obtained using the Brandes method.**

| Name | Vertices | Edges | # of Tests | Threshold | High BC Vertices |
|---|---|---|---|---|---|
| Karate | 34 | 156 | 18 | 55% | **1st, 2nd** |
| Chesapeake | 39 | 340 | 21 | 30% | **1st, 2nd** |
| AS20000102 | 6474 | 13233 | 243 | 12% | **1st, 2nd, 3rd** |
| AS20000101 | 3570 | 7391 | 180 | 16% | **1st, 2nd** |
| Caida | 16301 | 65910 | 384 | 21% | **1st, 2nd, 3rd** |
| C. Elegans | 453 | 4050 | 66 | 35% | **1st, 2nd, 4th 10th** + 3 low ranked |
| Les Mis. | 77 | 508 | 27 | 45% | **1st, 10th**, +3 low ranked |
| GrQc | 5242 | 28980 | 219 | 80.3% | 20 low ranked |
| HepTh | 9877 | 51971 | 300 | 76% | 6 low ranked |
| Power Grid | 4941 | 13188 | 213 | 84% | 6 low ranked |

from the original set and the new sets and took the mean of the Jaccard indices over multiple runs over the same $\epsilon$ value. As shown in Figure 3 some networks have a very stable top 10 BC set (Jaccard index very high) while some show increasing instability as the level increases. Power grid is an extreme example of instability where the top BC vertices almost never match after perturbations.

The top figure contains results for the networks on which group testing was successful. Note that the ranking is stable with high Jaccard index, or the value gradually declines. In contrast, Jaccard index values in the bottom figure, where group testing did not work, do not exhibit any specific pattern and even small $\epsilon$ values have very low Jaccard index.

One exception is the Les Mis dataset where group testing was successful in finding the top vertex and then identified the vertex with rank 10 as the next highest. Since this is the network of interactions between characters in the novel Les Miserbales, we could further investigate why group testing failed. The top vertex corresponds to Jean Valjean, the protagonist in the novel. The second highest BC vertex as computed by the Brandes method corresponds to Bishop Myriel, who only appears in 27 chapters out of a total 365. The Bishop is one of the few characters in the early chapters of the book who interacts with the protagonist and therefore is a vertex that links the characters in the first chapters with the rest of the novel. This character is almost an articulation point in the network, and therefore has high BC value.

Our group testing method, however, failed to find this second highest BC vertex and got the tenth highest vertex instead. This vertex corresponds to Tholomyes, a character who appears in only 9 chapters in the entire novel. This vertex is also an almost articulation point, and is connected with the network not by the protagonist but by a supporting character Fantine (BC rank 6).

In the group testing, if the Bishop vertex was often combined with other vertices that were also connected to the protagonist, then these other vertices duplicated the contribution of the Bishop vertex to the high BC value. On the other hand, the tenth highest vertex since it was not connected via the protagonist could contribute more to the BC value because it showed more unique connections. This is an example where combining several medium ranked BC vertices that cover different regions of the networks can lead to a higher values than the actual high BC vertex covering the same region.

*Scalability Results* We also present the scalability results for the parallel group testing implementation. The experiments were performed on the Firefly cluster at the Holland computing centre. The cluster consists of 280 nodes running two AMD Quad core processors and 871 nodes running two dual core Opteron processors [16]. We used MPI to exchange the jobs and information between the master and the worker processors.

We executed the group testing in parallel using a master-worker model, where each worker executes one test at a time and communicates the results to the master (as described in Section 3.2). Given below are the strong scalability results of four of the larger networks from our test suite. The results show that the parallel implementation is very scalable. Furthermore the parallel algorithm is designed such that we obtain the same results regardless of the number of processors used.

# 5. DISCUSSION AND FUTURE WORK

We have presented a novel way of identifying high betweenness centrality vertices using group testing. Our group testing method is particularly effective in networks that have stable BC ranking under small edge perturbations. This work is but a preliminary study to demonstrate the potential of group testing in network algorithms, and there are many research directions that can be pursued from these initial studies. Here we list some of them.

Although our current method focuses on finding only the top two betweenness centrality vertices, there exists other superimposed codes, such as those constructed from Reed-Solomon codes and randomly generated superimposed codes, that can be designed to find $d$ top betweenness centrality vertices for a specified value of $d$. There also exist simpler codes based on binary columns for finding only the top BC vertex. The variety of superimposed codes provide an interesting compromise between execution time and information, and we plan to experiment with other strategies to experiment this trade-off further. Alternatively we can also use the Latin Square method to recursively divide the network

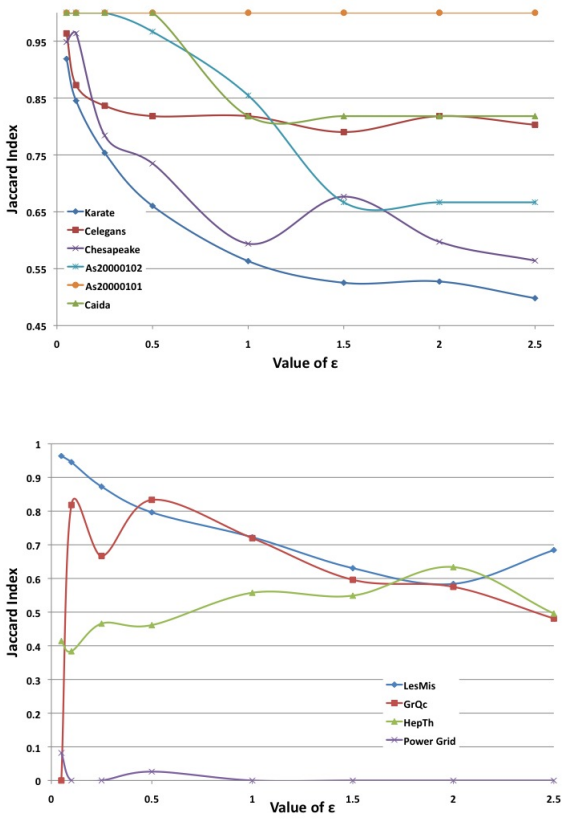


**Figure 3: Effect of perturbation on the ranking of the high BC vertices. Top: The ranking is stable or has a smooth decline. Bottom: Ranking is easily disrupted or has an erratic decline (except LesMis).**

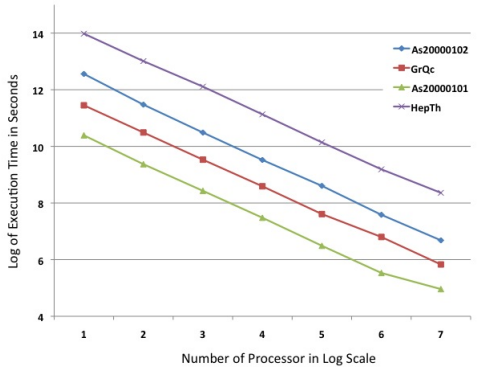


**Figure 4: Strong Scalability Results of Group Testing.**

across high BC vertices and then find the next higher BC vertices for the smaller networks.

We have seen that in some networks, the middle ranking vertices can combine to give false positive results. However, this phenomena can be utilized to identify alternate groups of important vertices. For example, if the goal is to disrupt the network and eliminating the highest BC vertex is disadvantageous, we can use group testing to find alternate groups of vertices that are not as high ranked, but together can be effective in disrupting the network.

Each test in group testing methods requires computing the value of only the supervertex. Currently, our implementation follows the more expensive method of using Brandes method to compute the BC for all vertices in the compressed network, and then use the value of the supervertex for group testing. However, recall that we only need to order the supervertices of each test by their relative and not exact value. Thus group testing can be made much more efficient if we use an approximation algorithm such as the one described in [3] to estimate the BC of the supervertex only. We plan to investigate further to determine such fast algorithms and levels of approximations that would be allowable, and thus create faster algorithms for computing betweenness centrality.

Finally it should be noted that our approach would also work for directed and weighted networks. The grouping of vertices remains exactly the same and only the BC calculation for each test is different. In the worst case we can use the Brandes algorithm version which is for directed and weighted networks though a better approach would involve using a BC algorithm that can compute the BC value of a specific vertex (as opposed to all vertices in the network).

## 6. ACKNOWLEDGEMENTS

The authors would like to thank V.S. Anil Kumar and A. Adiga for their help and support with the perturbation models, and V. Rykov for his support with group testing. This work was supported by F.I.R.E. (University of Nebraska at Omaha Sponsored Programs), and College of IS&T (University of Nebraska at Omaha).

## 7. REFERENCES

[1] A. Adiga, V.S. Anil Kumar, How robust is the core of a network?, (Pre-print from personal communication), 2012.

[2] Barabasi, A.L., Jeong, H., Ravasz, E., Neda, Z., Schuberts, A., Vicsek, T. Evolution of the social network of scientific collaborations. *Physica. A.* 311, 590-614 (2002)

[3] D. Bader, S. Kintali, K. Madduri, M. Mihail Approximating Betweenness Centralityâ *WAW2007*, 4863, 134 (2007)

[4] D. Bader and K. Madduri, Parallel Algorithms for Evaluating Centrality Indices in Real-World Networks, *ICPP*, (2006)

[5] U. Brandes , Faster Algorithm for Betweenness Centralityâ *J. Math. Sociol.*, 25, 163 (2001)

[6] Boguna, M., Pastor-Satorras, R., Vespignani: Epidemic spreading in complex networks with degree correlations. *Statistical Mechanics of Complex Networks.* Lecture Notes in Physics, vol. 625, pp. 127-147 (2003)

[7] M. Cheraghchi, A. Karbasi, S. Mohajer, V. Saligrama, Graph-Constrained Group Testing. *ISIT 2010* 1913-1917 (2010)

[8] W.H. Chong, W.S.B. Toh, L.N. Teow, Efficient Extraction of High-Betweenness Vertices, *ASONAM*, 31, 286, (2010)

[9] D. Du and F.K.Hwang, Combinatorial Group Testing and its Applicationsâ *World Scientific* (1993)

[10] DIMACS 10th Implementation Challenge http://www.cc.gatech.edu/dimacs10/archive/clustering.shtml (2011)

[11] A.G. D'yachkov and V.V. Rykov, Bounds on the Length of Disjunctive Codes, *Prob.Pered.Inform* 18(3), 7 (1982)

[12] A.G. D'yachkov and V.V. Rykov âSuperimposed Distance Codesâ*Prob.Cont.Inform.Theory*, 18(4) 237 (1989)

[13] A.G. D'yachkov, A.J. Macula, V.V. Rykov, New Constructions of Superimposed Codes, *IEEE Transactions on Information Theory*, 46(1) , 284-290, Jan 2000

[14] Nick Edmonds, Torsten Hoefler, and Andrew Lumsdaine. A Space-Efficient Parallel Algorithm for Computing Betweenness Centrality in Sparse Networks. *Indiana University Tech Report.* 2009.

[15] L.C. Freemanâ A Set of Measures of Centrality Based on Betweennessâ *Sociometry*, 40, 35 (1977)

[16] Firefly - Holland Computing Center http://hcc.unl.edu/firefly/

[17] R. Geisberger, P. Sanders, D. Schultes: Better Approximation of Betweenness Centrality. ALENEX 2008: 90-100

[18] M. Girvan and M.E.J. Newman, Community Structure in Social and Biological Networksâ *PNAS*, 99, n12, 7821 (2002)

[19] N. J. A. Harvey, M. Patrascu, Y. Wen, S. Yekhanin, V. W. S. Chan, Non-adaptive Fault Diagnosis for All-Optical Networks via Combinatorial Group Testing on Graphs *Proceedings of the 26th Annual IEEE Conference on Computer Communications (INFOCOM)*, pp.697-705, (2007)

[20] W.H. Kautz, R.C. Singleton, Nonrandom Binary Superimposed Codes,*IEEE Trans. Inform. Theory*, vol. 10, no. 4, pp. 363-377, (1964)

[21] A.J. Macula, L.J. Popyack , A group testing method for finding patterns in data, *Discrete Appl. Math.* 144, no. 1-2, 149–157, (2004)

[22] A.J. Macula, Probabilistic nonadaptive group testing in the presence of errors and DNA library screening, *Combinatorics and Biology* (Los Alamos, NM, 1998). Ann. Comb. 3, no. 1, 61–69, (1999)

[23] K. Okamoto, W. Chen, X.-Y. Li, Ranking of Closeness Centrality for Large-Scale Social Networks. FAW 2008: 186-195

[24] Stanford Network Analysis Project (SNAP) http://snap.stanford.edu/index.html

[25] V. V. Ufimtsev, A Scalable Group Testing Based Algorithm for Finding d-highest Betweenness Centrality Vertices in Large Scale Networks,Poster, *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011

[26] K. Voevodski, S.H.Teng, Y. Xia.: Finding local communities in protein networks. *BMC Bioinformatics* 10(10), 297 (2009)