A Scalable Group Testing Based Algorithm for Finding $d$-highest Betweenness Centrality Vertices in Large-Scale Networks

Vladimir V. Ufimtsev
Department of Computer Science, University of Nebraska at Omaha

1. Problem and Motivation

Systems of interacting entities from diverse disciplines, such as biology (correlation among genes) and sociology (collaboration networks), can be modeled as networks. Networks (also termed as graphs) consist of a set of vertices that correspond to the entities in the system, and a set of edges that correspond to the interaction between a pair of entities. The importance of a vertex is evaluated based on centrality metrics. Betweenness centrality (BC) is a popular centrality metric that measures the importance with respect to the flow of information in a network [1], [2], [3], [11]. Vertices (or edges) with high betweenness centrality are the most vulnerable points in an information flow network and also are used in divisive algorithms to detect communities [12].

Current algorithms for finding high BC vertices, first cumulatively find the values for all vertices in the network and then sort them to obtain the ones with the highest values. However most real-world networks are massive. For example, currently Facebook has over 845 million vertices. Thus even polynomial time algorithms can be computationally very expensive. Our goal is to lower the execution time for finding the highest BC vertices. We note that only the few highest BC vertices are required for most applications, and even then we need only the identity of the vertices and not their values. This observation inspired us to apply group testing (GT). The primary idea of group testing is to identify highly sensitive entities from a collection of objects. The idea originated in the spring of 1942, during World War II, by Robert Dorfman and David Rosenblatt [5] to efficiently test blood samples for millions of draftees. Since then, group testing has been used in many applications including finding counterfeit coins, finding patterns in data [15] and DNA library screening [16].

2. Background and Related Work

Graph Theory: 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 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 BC of vertex $v$ is defined as [11]: $BC(v) = \sum_{x \neq y \neq \emptyset \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, 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 $v$.

Betweenness centrality is generally computed cumulatively for every vertex in the network. The popular Brandes method has a complexity of $O(|V| \cdot |E|)$ time [3]. Faster algorithms include iterative methods based on pivots values for approximating BC scores [17] and sampling to obtain the BC of a single vertex [1]. Parallel BC algorithms have also been designed, although they too compute the values of every vertex [2].

**Group Testing:** Group testing (GT) [6] is a mathematical technique to find a specified number of defective units among a large population of units with the fewest number of tests. Given a population of $n$ units, units are termed as "defective" if they have a characteristic that is not present in the other "non-defective" units. For a sample of tests. Given a population of $n$ number of defective units among a large population of units with the fewest number $N$ tests on a sufficient number of groups, we can exactly identify the defective units. In designing an efficient group testing scheme, the goal is to carefully select the composition of the groups.

**Superimposed Code Theory** Take the binary $N \times n$ matrix (code) $X$. Let $x_{i,j} \in \{0, 1\}$ denote the element in row $i$ and column $j$ of $X$ and let $x_j$ $j = 1, 2, ..., n$ denote the $j^{th}$ column of $X$. The Boolean-OR sum of any $k$ columns $x_{j_1}, x_{j_2}, ..., x_{j_k}$ is:

$$f(x_{j_1}, x_{j_2}, ..., x_{j_k}) = \left[\begin{array}{c} x_{1,j_1} \lor x_{1,j_2} \lor \ldots \lor x_{1,j_k} \\ x_{2,j_1} \lor x_{2,j_2} \lor \ldots \lor x_{2,j_k} \\ \vdots \\ x_{N,j_1} \lor x_{N,j_2} \lor \ldots \lor x_{N,j_k} \end{array}\right]$$

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

A column $x_j$ covers column $x_i$ if $f(x_j, x_i) = x_j$. Code $X$ has strength $d$ if and only if the Boolean-OR sum of any $d$ columns does not cover any other column [8]. If $X$ is an $N \times n$ matrix then the code has length $N$ and size $n$. The weight: $w(x_j)$ of column $x_j$ is the number of non-zero elements in the column. The minimum weight $w = \min_{1 \leq j \leq n} w(x_j)$.

The intersection: $\lambda(x_j, x_i)$ between two columns $x_j, x_i$ is the number of positions in which both $x_j$ and $x_i$ have a 1. The maximum intersection $\lambda = \max_{1 \leq i \neq j \leq n} \lambda(x_j, x_i)$. The Kautz-Singleton Bound [14] states that a lower bound for the value of the parameter $d$, the strength of a code $X$ with minimum weight $w$ and maximum intersection $\lambda$ is: $d \geq \lceil \frac{\log_2 \lambda}{2} \rceil$

Superimposed codes of length $N$, size $n$, and strength $d$ can be implemented as group testing designs for populations of size $n$ that have at most $d$ defective units. It has been shown by D’yachkov and Rykov ([8],[9],[10]) that as $n \to \infty$ and $d \to \infty$ with $d \leq \log_2 n$, the minimum number of tests $N$ is bounded by: $\Omega \left( \frac{d^2}{\log_2 n} \log_2 n \right) \leq N \leq O \left( d^2 \log_2 \frac{n}{d} \right)$.

### 3. Approach and Uniqueness

In our problem, the $d$-highest BC vertices are taken to be the defective units in the set of all vertices $V$. 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. When
Figure 1: **Examples of Group Testing**: Figure (a): GT using All Binary Columns. Out of the eight units, unit 6 is defective as shown by the result vector. Figure (b): Construction of coding matrix using a 4 by 4 Latin square. The final matrix is given Figure (d), allows at most 16 units to be tested. Figure (c) shows a sample graph of two-8-cliques connected by one edge. The threshold is set to 65. Clearly vertices 4 and 8 are the ones with highest BC, as given by the results in Figure (d).

performing this grouping of vertices we are essentially compressing the original graph $G$ into a smaller one on which we perform the BC calculations.

**Using All Binary Columns.** This is a simple group testing design for finding only one defective unit. We create a matrix that specifies the groups as follows: each column represents the binary value of the number of the unit (numbered 0 through $n - 1$) and the $i^{th}$ position in the row indicates if the unit $s_i$ is either part of the group to be tested (1) or not (0). Each test has a result component which indicates the presence(1) or absence(0) of an infected unit. The binary value of the result column matches the binary value of the defective unit. This way we are able to identify the defective unit among $n$ units in exactly $\lceil \log_2 n \rceil$ tests.

**Using Latin Squares.** We used Latin squares for finding 2 defective units. Given a finite set of integers $\{1, 2, ..., l\}$, an $l \times l$ Latin square is a matrix $A$, where $A_{i, j} \in \{1, 2, ..., l\}$ such that each element from appears exactly once in any given row and column. We construct a coding matrix $X$ from a Latin square $L$ in the following way; the first 2 positions in any given column in $X$ are coordinates in $L$ and the 3rd position is the element in $L$ at those coordinates. We then encode each integer in $X$ in its binary form.

Due to the Latin square construction, any two columns in $X$ 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 $w = 3$ and maximum intersection $\lambda = 1$ then use the Kautz-Singleton Bound to get a guaranteed value for the strength parameter $d \geq \left\lfloor \frac{w-1}{\lambda} \right\rfloor = \left\lfloor \frac{2}{1} \right\rfloor = 2.$
Figure 2: Results of Applying Group Testing: Top Table: Networks on which group testing was successful. Bottom Table: Networks on which group testing was not as successful. Our method is most successful when there exist outstandingly high BC vertices. If the BC values are very close, then group testing fails to identify defective vertices.

Uniqueness There has been limited implementation of GT in graph theoretical contexts. Examples include finding optical network broken link identification [13], and congested links in wireless sensor networks (WSNs) [4]. However both these approaches set constraints to building the groups to be tested. Our algorithm, in contrast, does not have any constraints in group selection and we can use general combinatorial group testing designs. Our approach also allows for several tests to be performed simultaneously and is therefore easily parallelizable. Finding key vertices is an important operation in network analysis. Most algorithms focus on obtaining the values and then sorting the results. Our approach is one of the first that applies different strengths of GT to this problem and focuses on identifying the vertices rather than computing their centrality values.

4. Results and Contributions
We implemented group testing on real world networks examples given in [7]. For networks with only one highly distinguished BC vertex we used the "all binary columns" design which requires $\lceil \log_2 n \rceil$ tests. For networks with 2 highly distinguished BC vertices we used the Latin square method which requires $3 \lceil \sqrt{n} \rceil$ tests. Note that when selected vertices are grouped for each test, the network is compressed and the BC calculation is done on a smaller network, thus also reducing computation time. For example, in the the C. Elegans network, our compressed graphs were of 100 vertices and an average of 1500 edges, much smaller than the original size of 453 vertices and 4050 edges. Additionally, we were able to identify the vertex with the highest BC in only 9 tests (as opposed to
453 if testing each vertex individually). We can also "parallelize" by running the tests simultaneously across different processors. In the case of the C. Elegans network using parallel runs gives 34% reduction in execution time. The experiments were performed on a 600 core AMD cluster with 8 GB per node.

Contributions. We applied group testing in a novel context of finding the d- highest BC vertices in networks, where d ranges from 1 (binary columns) to 2 (Latin squares). Our method is most successful when there are outstandingly high values vertices, and our results can also help classify networks based on the distribution of their high centrality vertices.

We are currently investigating methods to select appropriate thresholds. Since each test relies on the computation of the BC of a single supervertex, we plan to develop an efficient method to compute the BC of exactly one specified vertex. We will also extend our algorithm to identify vertices for other values of d, beyond just 1 or 2 using superimposed codes constructed from Reed-Solomon codes.

Acknowledgements. The author would like to thank Dr. S. Bhowmick and Dr. V. Rykov for their help, insight, and support.

References


