A Faster Algorithm to Update Betweenness Centrality after Node Alteration

Keshav Goel[2], Rishi Ranjan Singh[1], Sudarshan Iyengar[1], and Sukrit[3]

[1] Department of Computer Science and Engineering, Indian Institute of Technology, Ropar, Punjab, India
{rishirs,sudarshan}@iitrpr.ac.in
[2] Department of Computer Engineering, National Institute of Technology, Kurukshetra, India
keshavgoel1993@gmail.com
[3] Department of Computer Science and Engineering, PEC University of Technology, Chandigarh, India
sukritkumar.becse11@pec.edu.in

Abstract. Betweenness centrality is a centrality measure that is widely used, with applications across several disciplines. It is a measure which quantifies the importance of a vertex based on its occurrence in shortest paths between all possible pairs of vertices in a graph. This is a global measure, and in order to find the betweenness centrality of a node, one is supposed to have complete information about the graph. Most of the algorithms that are used to find betwenness centrality assume the constancy of the graph and are not efficient for *dynamic networks*. We propose a technique to update betweenness centrality of a graph when nodes are added or deleted. Our algorithm experimentally speeds up the calculation of betweenness centrality (after updation) from 7 to 412 times, for real graphs, in comparison to the currently best known technique to find betweenness centrality.

Keywords: Betweenness Centrality, Minimum Cycle Basis, Bi-connected Components.

1 Introduction

Network Centrality measures are used to quantify the intuitive notion of nodes' importance in a network. There are several application centric definitions of network centrality measures, the popular ones being *degree centrality, closeness centrality, eigenvector centrality* and *betweenness centrality*. For background and description of centrality measures please refer to the excellent books by Newman [20] and Jackson [13].

There are a number of centrality indices based on the *shortest path lengths* (closeness centrality [22], graph centrality [10]) and the *number of shortest paths* (stress centrality [24], betweenness centrality [7, 1]) in a graph. Each centrality measure signifies a particular characteristic that is exhibited by a node. *Closeness centrality* of a vertex indicates the distance of a vertex from other vertices.

A. Bonato, M. Mitzenmacher, and P. Prałat (Eds.): WAW 2013, LNCS 8305, pp. 170–184, 2013.

Graph centrality denotes the difference between closeness centrality of the vertex under consideration and the vertex with the highest closeness centrality. *Stress centrality* simply denotes the total number of shortest paths passing through a vertex.

The idea of betweenness centrality was proposed by Freeman [7] and Anthonisse [1]. Betweenness centrality of a node v is defined as $BC(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} is the total number of shortest paths from vertex s to vertex t and $\sigma_{st}(v)$ is the total number of shortest paths from vertex s to vertex t passing through vertex v.

Betweenness centrality insinuates a more *global* characteristic unlike the degree centrality which considers the number of links from a node - which is clearly a local characteristic. Betweenness centrality has found many important applications in diverse fields. It has been used in the identification of sensitive nodes in biological networks[18]. Similarly, it can be used in electronic communication system networks, public transit system networks, gas pipeline networks, waste-water disposal system networks, etc. In protein-protein interaction (PPI) networks, essential proteins can be identified by their high betweenness centrality [14]. This characteristic of proteins can be used in selecting suitable drug targets [28] for various ailments including cancer[27], tuberculosis [23], zoonotic cutaneous leishmaniasis [6], etc. Betweenness centrality score of a person, on popular social networking sites, like 'Facebook' or 'Twitter', is being used by advertisers to choose him/her as an ambassador for their organization. Betweenness centrality is also used to identify nodes which are crucial for information flow in a brain network [12] where different regions of the brain represent nodes in the network and white matter represents the links.

Brandes [3] suggested an algorithm to calculate betweenness centrality that reduced the time complexity from $O(|V|^3)$ to $O(|V||E|)$ for unweighted graphs. Since, real world networks tend to be large and transient, such algorithms are found to be impractical if one requires to compute the betweenness centrality of nodes in a dynamic network. Work has been done by Lee et al. [15] and Green et al. [9] to find out betweenness centrality for edge updation in a graph. Most of the literature are found only for edge updation case. They assume that deletion of a node from a graph is equivalent to deleting all edges incident on that node. It is easy to analyze that algorithm proposed here is $deg(v)$-times faster than the algorithms with above mentioned concept for updation after node deletion where $deg(v)$ is the degree of the deleted vertex. The ranking of vertices on the basis of betweenness centrality is of use in various applications which are mentioned in the subsequent sections. To ascertain that the order of vertices in terms of their betweenness centralities before updation of graph is not the same as after updation, we performed experiments and got positive results. We present situations which demand a better way of updating betweenness centrality in changing networks.

1.1 Motivation

It is sometimes necessary to calculate betweenness centrality for a network at every stage of transition. With a large network and the current algorithms in use, recalculation becomes difficult. Some examples of such networks are given below.

- Complex communication networks are continuously growing and evolving. Each node in a communication network has a maximum capacity for carrying load[1], after which the node shuts down and its load is distributed among the remaining nodes. Due to increased load, other nodes may shut down and the network may become disconnected. This phenomena is commonly known as *cascading failure*. It has been found through experiments conducted by S. Narayanan [18] that breakdown of nodes with higher betweenness centrality causes greater harm. In such networks we can compute a sequence of nodes as following: We start with the given network. At each step, we delete the node with the highest betweenness centrality, add that node to the sequence and then repeat this process until the network becomes disconnected. This sequence can be used to decide the order in which security should be provided to the nodes in the network and that can ensure that if a node in the present network fails, the node with the highest betweenness centrality in the resulting network have enough security and resources. This requires repetitive calculation of betweenness centrality which when done with the conventional Brandes[3] algorithm will be highly inefficient. Similarly, points which have excess load in power grid systems and computer networks can be provided with more resources; stations with excess traffic in public transit systems can be provided with more measures to redistribute traffic and sewer lines with higher betweenness centrality can be provided with more frequent maintenance to prevent blockades. This exercise can also be done after the failure of some random node in a graph and appropriate actions on the nodes in the network may be taken thereafter.
- In social networking websites like 'Twitter' and 'Facebook', betweenness centrality of a node denotes the number of heterogeneous groups of nodes, the node under consideration links [25]. Since these nodes are involved in passing of information between heterogeneous groups of nodes, they're more important than a node with just a higher degree.[2] Also, we may want to determine the next important actor in case the current social network is altered. Such networks are highly dynamic due to the continuous addition and removal of actors.
- In a network composed of nations, the betweenness centrality of a nation describes its potential to act as information broker and provides information

[1] The amount of information flowing through a node in a communication network, is called its load.

[2] In the study conducted by A. Hanna [11] on the uprising in Egypt, where the social networking site 'Twitter' played an important role in the formation of public opinion against Mr. Hosni Mubarak (the then President of Egypt), it was found that these nodes played an important role in shaping public opinion.

about its overall activity level in the network. Thus nations can analyze the variation in their eminence with changes in network. They can analyze how other nations affect them and what actions will benefit or harm them. For example: Suppose in a network, countries are represented by nodes and a link signifies that there is a trade relation between the two countries. Countries may want to know what the effect of formation/removal of links or nodes of other countries with them or of other countries with other countries will be on their trade relations.
- A similar application can be in case a new actor wants to join a network. He will want to form links such that his prominence is maximum. He will have to form links to nodes accordingly. For example: A professional wants to join a network of other professionals in his area, he will try to connect with other actors considering what effect that will have on him. This requires repetitive calculation of betweenness centrality for a wide number of cases.

We tested our algorithm for both real and synthetic graphs and got positive results. For synthetic graphs, we achieved speedups ranging from 1.78 to 14 times and for real graphs we got speedups ranging from 7 times to 412 times, in comparison with Brandes algorithm [3]. In section 2, we present some basic definitions and concepts used in the paper. Section 3 contains the algorithm with explanation. Implementation and results are presented in section 4. We've further discussed the previous work conducted on betweenness centrality in section 5. We conclude in section 6.

2 Preliminary

In this section we define some terms which have been used throughout the paper. We also explain the basic concepts which provide basis for developing algorithm in section 3.

2.1 Terminology

We use following terms interchangeably throughout the paper; node or vertex and graph or network. A (simple) *path* in a graph is a sequence of edges connecting a sequence of vertices without any repetition of vertices. Thus a path between two vertices v_i and v_j (called terminal vertices) can be denoted as a sequence of vertices, $\{v_i, ..., v_j\}$ such that $v_i \neq v_j$ and no vertices in the sequence are repeated. The *length* of a path is the sum of the weights of edges in the path (edge weight is taken as one for unweighted graphs). A *shortest path* between two vertices is the smallest length path between them. An *end vertex* is a vertex with degree one. A graph is said to be connected if there exists a path between each pair of vertices. An *articulation vertex* is a vertex whose deletion will leave the graph disconnected. A *biconnected graph* is a connected graph having no articulation vertex. A *cycle* in a graph is a path having the same terminal vertices. A *cycle basis* of a graph is defined as a maximal set of linearly independent cycles. Weight of a cycle basis is the sum of the lengths

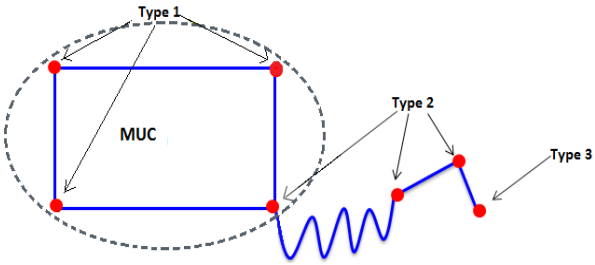

Fig. 1. Type 1: Vertex belongs to a MUC but is not an articulation vertex. Type 2: Vertex is an articulation vertex. Type 3: Vertex does not belong to a MUC and is an end vertex.

of all cycles in the cycle basis. A cycle basis of minimum total weight is called *minimum cycle basis (MCB)*.

The set achieved by repetitive merging (taking union) of all the elements of the MCB that have at least one vertex in common is called as *MUCset*. Each element of MUCset is termed as *Minimum Union Cycle (MUC)*. Thus, two MUCs can not have any vertex in common. A *connection vertex* c in a MUC (say MUC_i) is an articulation vertex such that it is adjacent to a vertex which does not belong to MUC_i. On removal of the connection vertex c, the graph will become disconnected and the components that are disconnected from MUC_i are together termed as *disconnected subgraph* G_c.

2.2 Basic Concept

Throughout the paper, we have considered only the case of vertex deletion in undirected unweighted connected graphs. For the case when a vertex is added, all lemmas, observations, and results hold with a slight modification. On the basis of the method used for updation of betweenness centrality after deletion of a vertex, we can categorize the vertices of the graph into three groups as mentioned in Figure 1.

In this paper, we explain the updation process after alteration of vertices of Type 1. Deletion of a vertex of Type 2 will leave the graph disconnected and concept of betweenness centrality will no longer be valid for the graph. So, we can not consider this case for updation. After deletion of vertices of Type 3, we can use a procedure similar to Algorithm 2 to update the centality scores. Now, we define few more terminologies, give lemmas and establish a theorem which provides basis to develop our algorithm.

Pair dependency of a pair of vertices (s,t) on a vertex v is defined as: $\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$ where σ_{st} is the number of shortest paths from vertex s to vertex t and $\sigma_{st}(v)$ is the number of shortest paths from vertex s to vertex t passing through vertex v. Betweenness centrality of a vertex v can be defined in terms of pair dependency as: $BC(v) = \sum_{s \neq v \neq t \in V} \delta_{st}(v)$. Let BFT_r denotes the breadth-first

traversal (BFT) of the graph rooted on vertex r. [3] *Dependency* of a vertex s on a vertex v is defined as: $\delta_{s\bullet}(v) = \sum_{t \in V \setminus \{s,v\}} \delta_{st}(v)$. Let us define a set $P^s(w)$ $= \{v : v \in V, w \text{ is a successor of } v \text{ in } BFT_s\}$. Brandes [3] proved that:

$$\delta_{s\bullet}(v) = \sum_{w:v \in P^s(w)} \frac{\sigma_{sv}}{\sigma_{sw}}(1 + \delta_{s\bullet}(w)). \tag{1}$$

Let $SP(v_i, v_j)$ be the set of all shortest paths from vertex v_i to v_j. Let MUC_U be the MUC where alteration has been made. Let G_i be the subgraph made of the components that will be disconnected from MUC_U after removal of connection vertex $c_i \in MUC_U$. Let $V(G_i)$ denote the set of vertices in subgraph G_i. Then we can establish following lemmas and theorem.

Lemma 1. *If v lies on all shortest paths between s and t, where $s \neq t \neq v \in V$, then:*

$$\sigma_{st} = \sigma_{sv}.\sigma_{vt}.$$

Lemma 2. *For $u \in V(G_k)$ and $v \in MUC_U$, every element of $SP(u, v)$ must contain c_k.*

Proof. Since c_k (connection vertex) is the only vertex that links G_k with MUC_U. So, every path between vertices in G_k and MUC_U must pass through c_k. □

Lemma 3. *Betweenness centrality of a vertex v can be changed only due to the shortest paths that had the altered vertex as one of their terminal vertices, where $v \in V \setminus MUC_U$.*

Proof. Assume $s, t \in V$ and $s \neq t \neq v$. Betweenness centrality of vertex v, $BC(v)$ will be influenced by following types of shortest paths:

1. Shortest paths that do not pass through the MUC_U. They start and end outside MUC_U, without passing through it. Naturally when an alteration is made in a graph, these paths remain unchanged. An example is shown in Fig. 2.
2. Shortest paths that have one terminal vertex in one disconnected subgraph G_i and the other in a disconnected subgraph G_j: These shortest paths may change. For one such shortest path, suppose the terminal vertices s and t are in different disconnected subgraphs, which pass through connection vertices c_1 and c_2, respectively (v lies in the disconnected subgraph where s lies). According to Lemma 1 and Lemma 2, $\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}} = \frac{\sigma_{sc_1}(v).\sigma_{c_1c_2}.\sigma_{c_2t}}{\sigma_{sc_1}.\sigma_{c_1c_2}.\sigma_{c_2t}} = \frac{\sigma_{sc_1}(v)}{\sigma_{sc_1}}$. We can observe that the shortest paths from s to c_1 remain same after node deletion and so this factor doesn't change.
3. Shortest paths that have one terminal vertex in one disconnected subgraph G_i and the other in MUC_U: Out of these shortest paths, the paths where deleted vertex is not a terminal vertex, we can get a relation similar to the one obtained above. When the deleted vertex is a terminal vertex (i.e. either s or t is deletion vertex), a factor of $\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$ should be deleted from

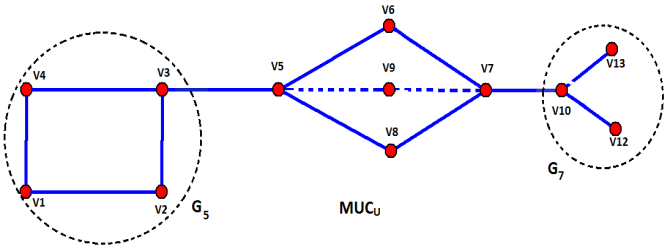

Fig. 2. Vertex $V9$ deleted. No effect on the shortest paths starting and ending in G_5 or G_7 itself. Other shortest paths may be altered.

betweenness centrality score of vertex v. This is because existing shortest paths from s to t are nonexistent now.

Thus, only one type of shortest paths (with altered vertex as one of terminal vertex) can change the betweenness centrality of the vertex v. □

Theorem 1. *Let v_d be the vertex to be deleted. Let $BC(v)$ be the betweenness centrality of the vertex v and dependency of the vertex v_d on the vertex $v \in V \setminus MUC_U$ is $\delta_{v_d\bullet}(v)$. Then the updated betweenness centrality of the vertex v after deletion of the vertex v_d can be calculated as:*

$$BC'(v) = BC(v) - 2\delta_{v_d\bullet}(v)$$

Proof. By the definition of dependency, $\delta_{v_d\bullet}(v)$ gives the effect of all shortest paths starting at vertex v_d in the betweenness centrality of node v. According to Lemma 3, shortest paths with v_d as terminal vertex (start vertex or end vertex on the path) are only affecting the change in centrality of vertices outside the MUC_U. After deletion of v_d, all such shortest paths will be deleted. Since the graph is undirected, $\sigma_{v_d t} = \sigma_{t v_d}$, so, we will subtract the dependency $\delta_{v_d\bullet}(v)$ twice. □

3 Algorithm

After deletion of a vertex which belonged to a MUC and was not an articulation vertex, we will update the betweenness centrality in different ways for the two types of vertices: vertices outside MUC_U and vertices in MUC_U. We use Theorem 1 to update betweenness centrality for vertices outside MUC_U and the algorithm is explained in detail in section 3.2. When vertices in MUC_U are considered, we observe that several shortest paths that were passing through altered vertex changed after deletion. So we recompute betweenness centrality using the idea given by Lee et al. [15] which is explained in brief in section 3.3. We assume that the betweenness centrality score of all vertices is available before proceeding with the preprocessing step of our algorithm.

Algorithm 1. Preprocesssing Step: Calculating MUCs in the Graph

```
1: Use Tarjan's Algorithm to calculate a set of biconnected components, C.
2: for each C_i ∈ C do
3:    if |C_i| = 2 then
4:       Remove C_i from C.
5:    end if
6: end for
7: while ∃ C_i, C_j ∈ C where C_i and C_j have at least one common vertex do
8:    Remove C_i and C_j from C.
9:    Insert C_i ∪ C_j in C.
10: end while
11: MUCset ← C
12: for each MUC_j ∈ MUCset do
13:    Find all the connection vertices and corresponding disconnected subgraphs.
14: end for
```

3.1 Preprocessing Step

Every time a change is made in the graph, updating the MUCset becomes necessary. We can do it in two ways, either by updation of MUCset (approach used in [15]) or by recalculation of MUCset. Approach for updation of MUCset takes longer time than recalculation. So, instead of updating MUCset, we recalculate it using the output of Tarjan's biconnected components algorithm[26] (commonly known as Tarjan's algorithm) as explained in algorithm 1. The time complexity for recalculation is $O(|V| + |E|)$. We use the following procedure for calculating MUCset.

Every graph can be decomposed into a set of biconnected components, C, where the elements of C are denoted by C_i using Tarjan's algorithm. Let $|C_i|$ denote the number of vertices in the biconnected component C_i. Each biconnected component contains at least one edge (two vertices) and may share vertices (articulation vertex) with other biconnected components. We remove components which contain only one edge because single edge can not form a MUC. Since, the elements of a MUCset are disjoint, we take repetitive union of the components that have atleast one vertex in common. We form the MUCset in this fashion. Then for each MUC, we calculate connection vertices and disconnected subgraph(s) associated with each connection vertex.

3.2 Calculating Changes in Betweenness Centrality for Vertices outside MUC_U

Effect of the altered vertex on betweenness centrality of vertices outside MUC_U can be found by forming breadth-first traversal (BFT) for the vertex that was deleted. The BFT can be calculated with a time complexity of $O(|E|)$. We then calculate the dependency of each vertex with respect to deletion vertex, starting from the vertices in the bottom level and recursively calculate the dependency for vertices in subsequent higher levels using equation 1. Then we use Theorem 1

to update the centrality values. The complete procedure is shown in Algorithm 2. In case of vertex addition, we will add the dependency to the betweenness centrality scores of each vertex outside MUC_U.

3.3 Calculating Betweenness Centrality for Vertices in MUC_U

This section briefly describes the idea suggested by Lee et al. [15] for recomputation of betweenness centrality for vertices in MUC_U. In the disconnected subgraph G_j, let $V(G_j)$ denote the vertex set and let $|V(G_j)|$ denote the number of vertices. Let $|SP(u,v)|$ denote the number of shortest paths between vertex u and vertex v. Here, we will explain the basic steps of the algorithm, in brief. For detailed concept and used algorithm, please refer to the QUBE algorithm [15]. Let betweenness centrality of vertex v, $BC(v)$ for all $v \in MUC_U$ be initialized with 0. Let c_j be a connection vertex of MUC_U and G_j be the corresponding disconnected subgraph. Now we calculate the betweenness centrality of vertex v by calculating and adding the effect of following three types of shortest paths on vertex v:

1. The shortest paths with both source and destination in MUC_U: For counting the effect of these paths, we use the algorithm suggested by Brandes [3] for only the vertices in MUC_U and compute local betweenness centrality $BC_0^{MUC_U}(v)$, for all vertices $v \in MUC_U$.
2. The shortest paths with either source or destination (but not both) in MUC_U: Let $< s, .., t >$ be a shortest path from $s \in V(G_j)$ to $t \in MUC_U$. In this case, $\frac{\sigma_{st}(v)}{\sigma_{st}} = \frac{\sigma_{c_j t}(v)}{\sigma_{c_j t}}$. So, to calculate the total effect of such paths, for each shortest path $< c_j, ..., t >$, we add the following factor to $BC(v)$:

$$BC_1^{<c_j,...,t>}(v) = \begin{cases} \frac{|V(G_j)|}{|SP(c_j,t)|}, & \text{if } v \in < c_j, ..., v_t > \setminus \{v_t\} \\ 0, & \text{otherwise} \end{cases}$$

3. The shortest paths with neither source nor destination in MUC_U: Let $< s, .., t >$ be a shortest path from $s \in V(G_j)$ to $t \in V(G_k)$ where $j \neq k$. In this case, $\frac{\sigma_{st}(v)}{\sigma_{st}} = \frac{\sigma_{c_j c_k}(v)}{\sigma_{c_j c_k}}$. So, to calculate the total effect of such paths, for each shortest path $< c_j, ..., c_k >$, we add the following factor to $BC(v)$:

$$BC_2^{<c_j,...,c_k>}(v) = \begin{cases} \frac{|V(G_j)||V(G_k)|}{|SP(c_j,c_k)|}, & \text{if } v \in < c_j, ..., c_k > \\ 0, & \text{otherwise} \end{cases}$$

When either of the subgraphs is disconnected, an additional factor:

$$BC_3^i(c_i) = \begin{cases} |V(G_i)|^2 - \sum_{l=1}^{x}(|V(G_i^l)|^2), & \text{if } G_i \text{ is disconnected} \\ 0, & \text{otherwise} \end{cases}$$

is added to the betweenness centrality calculations (c_i is a connection vertex). Where ${G_j}^l$ is the lth component of G_i and x is the number of connected components in G_i.

So, we have the following formula to calculate the betweenness centrality score of a vertex in MUC_U:

$$BC(v) = BC_0^{MUC_U}(v) + 2\sum_{G_j,t}\sum_{x\in SP(c_j,t)} BC_1^x(v) + \sum_{G_j,G_k(j\neq k)}\sum_{y\in SP(c_j,c_k)} BC_2^y(v) \quad (+\, BC_3^i(c_i) \quad if\ v = c_i).$$

4 Implementation and Results

We have implemented the algorithm for deletion of vertices, however it can be easily modified to implement node addition. The algorithm can work faster for updating betweenness centrality, since it forms a subset of vertices of the graph, MUC_U, for which recalculation of betweenness centrality is to be done. For the rest of vertices, algorithm 2 updates the betweenness centrality in negligible time as compared to recalculation step. The recalculation procedure includes local Brandes algorithm so it is directly proportional to number of vertices inside MUC_U. We have compared our results with Brandes algorithm [3] since that is the best known algorithm, according to our knowledge, for calculation of betweenness centrality after vertex updation. The experiments were performed on an Intel i5-2450M C.P.U. with 2.5 GHz clock speed and 4 G.B. main memory.

We use a similar measure used by authors in [15] termed as *proportion* to compare the algorithms. Proportion can be calculated as:

$$\left(\frac{Number\ of\ vertices\ in\ MUC_U}{Total\ number\ of\ vertices\ in\ the\ graph}\right).100$$

Proportion is a direct function of the number of vertices in MUC_U and thus speedup achieved by our algorithm is directly affected by the proportion. So, a smaller proportion would mean betweenness centrality for lesser number of nodes will have to be recomputed and so this should achieve greater speed-up which we achieved in our experimental result. We considered the following strategy to compute the *average proportion* of a graph. We randomly start deleting vertices from the graph till either the graph becomes disconnected or k vertices are deleted. Then we take the average of proportion values for each deletion. The *average speed-up* is calculated in a similar way. We consider $k = 500$ for real networks. In general, average speed-up on a graph or average proportion of a graph depends on the number of MUCs formed by biconnected components and the fraction of total number of vertices belongs to these MUCs. If a graph consists of most of the MUCs with small number of vertices with respect to the total number of vertices in the graph, the average proportion will be small and thus average speed-up on that graph will be large.

4.1 Results for Synthetic Graphs

We derived three groups of synthetic graphs with 1000, 2000 and 3000 nodes. In each group we generated graphs with proportion x, for each $x \in A$, where $A = \{10, 20, 30, 40, 50, 60, 70, 80\}$. These graphs were random graphs based on the Erdös Rényi graph model. To implement the algorithm, we initially calculated the betweenness centrality of each vertex using Brandes [3] algorithm. Then we ran the preprocessing step (Algorithm 1) to calculate the MUCs, connection

Algorithm 2. Calculating BFT and updating the vertices outside MUC_U accordingly

v_d: Vertex to be deleted.
Input: $BC[v]$ of each vertex of original graph ($v \in V$).
$S \leftarrow$ Empty Stack
$P[w] \leftarrow$Empty List, $w \in V$
$\sigma[t] \leftarrow 0$, $t \in V$, $\sigma[v_d] = 1$
$d[t] \leftarrow -1$, $t \in V$, $d[v_d] = 0$
$Q \leftarrow$ Empty Queue
Enqueue $v_d \rightarrow Q$
while Q not empty **do**
  Dequeue $v \leftarrow Q$
  push $v \rightarrow S$
  for each neighbour of v **do**
    if $d[w] < 0$ **then**
      enqueue $w \rightarrow Q$
      $d[w] \leftarrow d[v] + 1$
    end if
    if $d[w] = d[v] + 1$ **then**
      $\sigma[w] \leftarrow \sigma[w] + \sigma[v]$
      append $v \rightarrow P[w]$
    end if
  end for
end while
$\delta[v] \leftarrow 0$, $v \in V$
while S not empty **do**
  pop $w \leftarrow S$
  for $v \in P[w]$ **do**
    $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]}(1 + \delta[w])$
  end for
end while
for $v \in V \setminus MUC_U$ **do**
  $BC[v] \leftarrow BC[v] - 2.\delta[v]$
end for

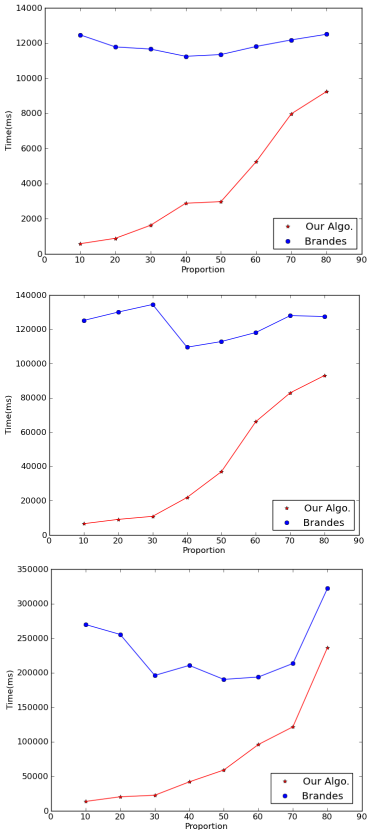

Fig. 3. Comparison of ours and Brandes' result on synthetic graphs with 1000, 2000, and 3000 vertices respectively

vertices and disconnected subgraphs for our graphs. 50 vertices were removed from each x proportion graph of each group and average updation times were calculated for different proportions.

The results are plotted with average updation time (in ms) at the y-axis and the proportion of the graphs at the x-axis for each group and are shown in Fig. 3. We get different speed-ups for different proportion graphs in each group. For synthetic graphs with 1000 nodes, we achieve a speedup of 13.26, 3.90 and 2.25 for graph with proportion 20, 40 and 60 respectively. For synthetic graphs with 2000 nodes, we achieve a speedup of 14.27, 5.01 and 1.78 for graph with proportion 20, 40 and 60 respectively. Similarly, for synthetic graphs with 3000 nodes, we achieve a speedup of 12.54, 5.02 and 2.02 for graph with proportion 20, 40 and 60 respectively as shown in Fig. 3.

4.2 Results for Real Graphs

We also tested our algorithm on real world networks. We chose different types of real datasets to depict the flexibility of our algorithm. We converted directed graphs into undirected graphs. Collaboration networks generally depict the collaborations an author has made while writing research papers. Interaction networks are networks where nodes are connected due to their features. An ownership network suggests transfer of resources between two nodes. Trust network is a network of individuals with kindred interests and connections. The yeast protein-protein interaction network was also taken as an input.[3]

For such various real networks like Geom[3], Erdos02[4], Erdos972[5], etc., information about networks, average proportion and average speed-ups over Brandes algorithm are summarized in Table 1.

Table 1. Figures for Real Data sets

Name of Dataset	Type	$\|V\|$	$\|E\|$	Avg. Proportion	Avg. Speed-up
YeastL	Interaction	2361	6646	31.16	64.88
YeastS	Interaction	2361	6646	28.05	72.76
Geom	Collaboration	7343	11898	17.08	7.63
Erdos02	Collaboration	6927	11850	7.13	323.75
Edros972	Collaboration	5488	8972	9.49	411.835
ODLIS[21]	Dictionary data	2909	16377	67.68	18.52
Wiki-vote[16]	Trust	8297	100762	38.452	28.472

5 Related Work

The idea of betweenness centrality was first introduced by Freeman [7] and Anthonisse [1]. Newman [19] defined another measure that considered random

[3] Dataset available at `http://vlado.fmf.uni-lj.si/pub/networks/data`

[4] Available at `http://www.cise.u.edu/research/sparse/matrices/Pajek/`

[5] `http://www.cise.ufl.edu/research/sparse/matrices/`

walks on any arbitrary length rather than just shortest paths between two vertices. Brandes [5] considered other types of betweenness centrality like edge betweenness and group betweenness and algorithms to compute them efficiently. Betweenness centrality was earlier calculated by finding the number and length of shortest paths between two vertices and then adding up pair dependencies for all pairs. Brandes [3] suggested an algorithm which introduces a recursive way to sum the dependencies in graphs. Although the algorithm proposed by Brandes[3] was faster than the one used previously, it was still too costly for large graphs. So, several approximation algorithms were proposed by Bader et al. [2], Brandes et al. [4], Geisberger et al. [8] and Makarychev [17]. Real world networks tend to be large and transient. Work has been done by Lee et al. [15] and Green et al. [9] to find out betweenness centrality after updation in a graph. The algorithm suggested by [15] selects a subset of vertices whose betweenness centrality is updated. However, it works only in the case of edge removal and addition. The algorithm suggested by [9] takes into consideration different instances that may arise due to edge addition in a graph and speeds up the algorithm in these cases. This algorithm works only for streaming graphs i.e. only in case of edge additions.

6 Conclusion

In this paper, we formulated an algorithm that efficiently calculates betweenness centrality when vertices in a graph are updated. We did not consider the traditional way for updating betweenness centrality after node alteration which considers a node alteration event as a series of edge alteration event. We achieved the speedups by calculating two sets of vertices; one for which we need to update betweenness scores and the other for which we need to recompute the betweenness score. We achieve an average speedup of 4.5 for a proportion of 40 for synthetic graphs as compared to Brandes algorithm. For real graphs, we get an average speedup of around 133 for a proportion of 29. The speedup will increase futher when proportion decreases.

Acknowledgement. The authors would like to thank the anonymous reviewers for comments that helped with the clarification of some concepts. They would also like to thank M.J. Lee and Yayati Gupta for their invaluable help.

References

[1] Anthonisse, J.M.: The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam (1971)
[2] Bader, D.A., Madduri, K.: Parallel algorithms for evaluating centrality indices in real-world networks. In: Proceedings of the 2006 International Conference on Parallel Processing, ICPP 2006, pp. 539–550 (2006)
[3] Brandes, U.: A Faster Algorithm for Betweenness Centrality. Journal of Mathematical Sociology 25(2), 163–177 (2001)

[4] Brandes, U., Pich, C.: Centrality estimation in large networks. International Journal of Bifurcation and Chaos 17(7), 2303 (2007)
[5] Brandes, U.: On variants of shortest-path betweenness centrality and their generic computation. Social Networks 30(2), 136–145 (2008)
[6] Florez, A.F., Park, D., Bhak, J., Kim, B.C., Kuchinsky, A., Morris, J.H., Espinosa, J., Muskus, C.: Protein network prediction and topological analysis in Leishmania major as a tool for drug target selection. BMC Bioinformatics 11, 484 (2010)
[7] Freeman, L.A.: set of measures of centrality based on betweenness. Sociometry 40, 35–41 (1977)
[8] Geisberger, R., Sanders, P., Schultes, D.: Better approximation of betweenness centrality. In: Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX), pp. 90–100. SIAM (2008)
[9] Green, O., McColl, R., Bader, D.A.: A fast algorithm for streaming betweenness centrality. In: 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust (2012)
[10] Hage, P., Harary, F.: Eccentricity and centrality in networks. Social Networks 17, 57–63 (1995)
[11] Hanna, A.: Revolutionary Making and Self-Understanding: The Case of #Jan25 and Social Media Activism. Presented at Meeting of the International Studies Association, San Diego, CA (2012)
[12] Iturria-Medina, Y., Sotero, R.C., Canales-Rodríguez, E.J., Alemán-Gómez, Y., Melie-García, L.: Studying the human brain anatomical network via diffusion-weighted MRI and Graph Theory. NeuroImage 40(3), 1064–1076 (2008)
[13] Jackson, M.O.: Social and Economic Networks. Princeton University Press (2010)
[14] Joy, M.P., Brock, A., Ingber, D.E., Huang, S.: High-Betweenness Proteins in the Yeast Protein Interaction Network. J. Biomed. Biotechnol. 2, 96–103 (2005)
[15] Lee, M.J., Lee, J., Park, J.Y., Choi, R.H., Chung, C.W.: QUBE: a Quick algorithm for Updating BEtweenness centrality. In: Proceedings of the 21st International Conference on World Wide Web, pp. 351–360 (2012)
[16] Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. ACM Trans. Knowl. Discov. Data 1(1), 2 (2007)
[17] Makarychev, Y.: Simple linear time approximation algorithm for betweenness. Operations Research Letters 40(6), 450–452 (2012)
[18] Narayanan, S.: The Betweenness Centrality of Biological Networks. M. Sc. Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University (2005)
[19] Newman, M.E.J.: A measure of betweenness centrality based on random walks. Social Networks 27(1), 39–54 (2005)
[20] Newman, M.: Networks: An Introduction. Oxford University Press, Oxford (2010)
[21] Reitz, J.M.: ODLIS: Online Dictionary of Library and Information Science (2002)
[22] Sabidussi, G.: The centrality index of a graph. Psychometrika 31, 581–603 (1966)
[23] Shanmugham, B., Pan, A.: Identification and Characterization of Potential Therapeutic Candidates in Emerging Human Pathogen Mycobacterium abscessus: A Novel Hierarchical In Silico Approach. PLoS ONE 8(3), e59126 (2013), doi:10.1371/journal.pone.0059126
[24] Shimbel, A.: Structural parameters of communication networks. Bulletin of Mathematical Biophysics 15, 501–507 (1953)

[25] Spiliotopoulos, T., Oakley, I.: Applications of Social Network Analysis for User Modeling. In: International Workshop on User Modeling from Social Media / IUI 2012 (2012)
[26] Tarjan, R.: Depth-First Search and Linear Graph Algorithms. SIAM J. Comput. 1(2), 146–160 (1972)
[27] Website of National Cancer Institute, http://www.cancer.gov
[28] Yu, H., Kim, P.M., Sprecher, E., Trifonov, V., Gerstein, M.: The Importance of Bottlenecks in Protein Networks: Correlation with Gene Essentiality and Expression Dynamics. PLoS Comput. Biol. 3(4), e59 (2007), doi:10.1371/journal.pcbi.0030059