Finding k-Highest Betweenness Centrality Vertices in Graphs

Min-Joong Lee
Department of Computer Science, KAIST
291 Daehak-ro, Yuseong-gu
Daejeon, Korea
mjlee@islab.kaist.ac.kr

Chin-Wan Chung
Division of Web Science and Technology &
Department of Computer Science, KAIST
291 Daehak-ro, Yuseong-gu
Daejeon, Korea
chungcw@kaist.edu

ABSTRACT

The betweenness centrality is a measure for the relative participation of the vertex in the shortest paths in the graph. In many cases, we are interested in the k-highest betweenness centrality vertices only rather than all the vertices in a graph. In this paper, we study an efficient algorithm for finding the exact k-highest betweenness centrality vertices.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms, Path and circuit problems*; E.1 [**Data**]: Data structures—*Graphs and networks*

Keywords

Graph; Betweenness centrality; Top-k

1. INTRODUCTION

The centrality is one of the essential concepts for the analysis of networks, and the betweenness centrality is one of the most prominent measures among several centrality measures. The betweenness centrality problem has been extensively studied in the literature since the idea of the betweenness centrality is defined by Anthonisse et al. [1]. The fastest known algorithm to compute exact betweenness centralities for all the vertices is proposed by Brandes et al. [2]. Recently, starting from the work by Lee et al. [3], a few works address an updating problem in dynamic graphs.

However, in many cases, we are only interested in the k-highest betweenness centrality vertices rather than the betweenness centralities of all the vertices in a graph. This is reasonable since a vertex with a higher centrality is viewed as a more important vertex than a vertex with a lower centrality, and we are only interested in the important vertices in many applications such as finding influencers in a social network, and locating bottlenecked junctions/routers in a transportation network/the Internet. Moreover, a community detection application, one of the most prominent application of the betweenness centrality, utilizes the highest centrality vertex only. Despite the above evident applications, the k-highest betweenness centrality problem is hardly discussed in the literature. In this paper, we propose an efficient algorithm for finding the exact k-highest betweenness centrality vertices in a graph. We utilize the novel properties of biconnected components to compute the betweenness centralities of a part of vertices in a graph, and employ an idea of the upper-bounding to compute the relative partial order of the vertices with respect to their betweenness centralities.

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2577358.

2. PRELIMINARY

Definition 1 (Betweenness Centrality). A graph is represented by $G = (V, E)$. The betweenness centrality of a vertex $v_j \in V$ is defined as follow. $c(v) = \sum_{i,j \in V} (\sigma_{i,j}(v)/\sigma_{i,j})$ where $v, i, j \in V$, $v \neq i, i \neq j, j \neq v$, $\sigma_{i,j}(v)$ is the number of shortest paths between i and j that include v, and $\sigma_{i,j}$ is the number of shortest paths between i and j.

Definition 2 (Biconnected Component and Articulation Vertex). A *biconnected graph* is a inseparable graph by removing an any vertex in a graph, and the *biconnected component* is a maximal *biconnected subgraph* of the graph. Any connected graph can be decomposed into *biconnected components* that are connected to each other through a vertex called *articulation vertex*. An example is shown in Figure 1.

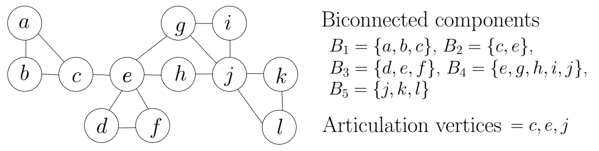
a b c d e f g h i j k l
Biconnected components
$B_1 = \{a, b, c\}$, $B_2 = \{c, e\}$,
$B_3 = \{d, e, f\}$, $B_4 = \{e, g, h, i, j\}$,
$B_5 = \{j, k, l\}$
Articulation vertices $= c, e, j$

Figure 1: Graph Example.

3. SOLUTION

We can efficiently find the k-highest betweenness centrality vertices by solving the following two sub-problems.

1. Compute the betweenness centrality of a vertex faster than the betweenness centralities of all vertices.
2. Compute the partial order of vertices with respect to their betweenness centralities.

Sub-proplem 1. To compute the betweenness centrality of a single vertex v, $\sigma_{i,j}(v)/\sigma_{i,j}$ for all vertices pairs i, j in a graph should be computed by Definition 1. Therefore, solving this sub-problem seems impossible. However, we propose a novel idea to compute $\sigma_{i,j}(v)/\sigma_{i,j}$ without performing all-pairs shortest paths computation for all vertices pairs i, j in the graph.

Let $\mathbb{V}(B_v) = \{V^i, ..., V^k\}$ be a set of vertex sets. $V^i \in \mathbb{V}(B_v)$ is a set of vertices in the disconnected component which has a vertex i among the disconnected components induced by removals of all vertices in B_v. Then, the betweenness centrality of vertex v in B_v can be efficiently computed as follow.

$$c(v) = c_{B_v}(v) + \sum_{i \in B_v, V^j \in \mathbb{V}(B_v)} \frac{|V^j| \cdot \sigma_{i,j}(v)}{\sigma_{i,j}} + \sum_{V^i, V^j \in \mathbb{V}(B_v)} \frac{|V^i| \cdot |V^j| \cdot \sigma_{i,j}(v)}{\sigma_{i,j}}$$

where $i \neq j$. The first term $c_{B_v}(v)$ represents an amount of the betweenness centrality of v with respect to the shortest paths between vertices in B_v. Since the shortest paths between vertices in a biconnected component only include vertices in the component, we can compute this term by computing the betweenness centrality of v on B_v using any existing betweenness centrality computation algorithm. The second term represents an amount of the betweenness centrality with respect to the shortest paths between a vertex in B_v and a vertex in not B_v, and the third term represents an amount of the betweenness centrality with respect to the shortest paths between vertices not in B_v. Note that the above equation requires the shortest paths between pairs of vertices i, j in B_v only, and the betweenness centralities of all vertices in B_v can be computed also using the shortest paths between i, j pairs in B_v. The number of vertices in a biconnected component is fairly smaller than that in the entire graph. In Figure 1, $c(h) = \frac{1}{2} + (\frac{5 \cdot 1}{2} + \frac{2 \cdot 1}{2}) + (\frac{5 \cdot 2 \cdot 1}{2})$.

Sub-proplem 2. The betweenness centrality of any vertex in B_v can not exceeds $\mathcal{C}(|B_v|, 2)$, the number of ways of choosing two vertices from B_v. It is used to exclude biconnected components with a small vertices. The number of the shortest paths between a vertex in B_v and a vertex not in B_v is $(|V| - |B_v|) \cdot (|B_v| - 1)$. Among the shortest paths between vertices not in B_v, $\sum_{V^i, V^j \in \mathbb{V}(B_v)} |V^i| \cdot |V^j|$ shortest paths include at least one vertex in B_v. It is used to exclude outlier biconnected components. Using the above numbers, the upper-bound of the betweenness centralities of vertices in B_v, denoted as $\widehat{c}(B_v)$, can be computed as $\widehat{c}(B_v) = \mathcal{C}(|B_v|, 2) + (|V| - |B_v|) \cdot (|B_v| - 1) + \sum_{V^i, V^j \in \mathbb{V}(B_v)} |V^i| \cdot |V^j|$. The majority of cases, this upper-bound is an overestimate compared to the actual centralities. Despite the overestimate, it is enough to give a concrete theoretical reason for pruning small and outlier components.

Overall Process. The overall process of our proposed algorithm is as follows.

1. Decompose G into biconnected components. Let $\mathcal{B}$ be a set of the decomposed biconnected components.
2. Calculate $\widehat{c}(B_v)$, the upper-bound of the betweenness centralities of vertices in B_v, for each $B_v \in \mathcal{B}$.
3. Calculate and update the betweenness centralities of vertices in B_h. B_h is B_v with the highest $\widehat{c}(B_v)$ among $B_v \in \mathcal{B}$.
4. Update $\mathcal{B}$ to $\mathcal{B} \setminus \{B_h\}$.
5. Repeat 3-4 until the known k-th highest betweenness centrality is higher than the new $\widehat{c}(B_h)$.

Optimization. We can further optimize the above algorithm by computing and utilizing the lower-bound of the betweenness centrality of an articulation vertex after Step 2. Let $\mathbb{V}(a) = \{V_1, ..., V_m\}$ be a set of vertex sets. $V_i \in \mathbb{V}(a)$ is consist of vertices in each disconnected component induced by the removal of an articulation vertex a. The lower-bound of the betweenness centrality of an articulation vertex a is denoted and computed as $\check{c}(a) = \sum_{V_i, V_j \in \mathbb{V}(a)} |V_i| \cdot |V_j|$ where $V_i \neq V_j$. Only the amount of the betweenness centrality, with respects to the shortest paths between vertices in each biconnected component which has a, is not considered in the lower-bound of the centrality of a. Thus, it is a fairly tight bound, and articulation vertices generally have higher betweenness centralities than non-articulation vertices. Therefore, we can prune vertices in many biconnected components which have smaller upper-bounds than the lower-bound of an articulation vertex.

4. EXPERIMENTAL RESULTS

Since an algorithm for finding the exact k-highest betweenness centrality does not exist, we compare our algorithm to the Brandes algorithm which is the fastest known algorithm for the exact betweenness centrality computation. Note that all existing algorithms require all-pairs shortest paths even for computing the centrality of one vertex in a graph. We show the experimental results with the summary of graphs in Table 1. The speed-up shows how much improvement is achieved by our algorithm compared to the Brandes algorithm. For example, our algorithm is 3312 times faster in 'eva' dataset. Moreover, our algorithm is scalable with respect to k since it compute the centralities of vertices in a biconnected component simultaneously.

Table 1: Experiments on Real Graphs

Graph	Type	$\|V\|$	$\|E\|$	Speed-Up				
				k=1	k=5	k=10	k=50	k=100
disease	Ownership	516	2376	18.22		18.09	17.38	17.12
eva	Ownership	8343	6726	3312.37		3312.13		
erdos972	Collab.	5822	14750	37.43				
geon	Collab.	9072	13567	6.33			6.31	
CAGrQc	Social	5242	14496	3.321			3.313	
power	Web link	4941	13188	4.021				4.020
pgp	Web link	4680	24340	21.61				
CAhep	Collab.	9877	25998	3.02				
contact	Social	11604	88806	4.24				

5. CONCLUSION AND FUTURE WORKS

In this paper, we propose an efficient algorithm for finding the exact k-highest betweenness centrality vertices. To the best of our knowledge, this is the first work which addresses the exact k-highest betweenness centrality problem. The proposed algorithm utilizes the novel properties of biconnected components, and outperforms the existing algorithm for finding the k-highest betweenness centrality vertices.

As future works, we plan to improve the upper-bound computation. Currently, an upper-bound is computed by only using the number of vertices in a biconnected component. However, we expect a proper utilization of other graph measures, such as the number of edges and the diameter, will tighten the upper-bound. Also, we plan to employ our idea for computing the exact betweenness centralities for all vertices. This is also promising since our idea provides a key for the divide and conquer computation.

Acknowledgments. This work was supported by the National Research Foundation of Korea grant funded by the Korean government (MSIP) (No. NRF-2009-0081365).

6. REFERENCES

[1] J. M. Anthonisse. The rush in a directed graph. *Stichting Mathematisch Centrum. Mathematische Besliskunde*, (BN 9/71):1–10, 1971.

[2] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(1994):163–177, 2001.

[3] M.-J. Lee, J. Lee, J. Y. Park, R. H. Choi, and C.-W. Chung. Qube: a quick algorithm for updating betweenness centrality. In *Proceedings of the 21st international conference on World Wide Web*, pages 351–360. ACM, 2012.