Information Sciences 326 (2016) 278–296

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Efficient algorithms for updating betweenness centrality in fully dynamic graphs

Min-Joong Lee [a], Sunghee Choi [a], Chin-Wan Chung [a,b,*]

[a] The School of Computing, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea
[b] The Chongqing Liangjiang KAIST International Program, Chongqing University of Technology (CQUT), Chongqing, China

ARTICLE INFO

Article history:
Received 1 January 2015
Revised 25 April 2015
Accepted 23 July 2015
Available online 6 August 2015

Keywords:
Betweenness centrality
Update algorithm
Biconnected component
Dynamic graph
Community detection

ABSTRACT

Betweenness centrality of a vertex (edge) in a graph is a measure for the relative participation of the vertex (edge) in the shortest paths in the graph. Betweenness centrality is widely used in various areas such as biology, transportation, and social networks. In this paper, we study the update problem of betweenness centrality in fully dynamic graphs. The proposed update algorithm substantially reduces the number of shortest paths which should be re-computed when a graph is changed. In addition, we adapt a community detection algorithm using the proposed algorithm to show how much benefit can be obtained from the proposed algorithm in a practical application. Experimental results on real graphs show that the proposed algorithm efficiently update betweenness centrality and detect communities in a graph.

1. Introduction

Centralities are one of the essential concepts for the analysis of networks, and betweenness centrality [22] is one of the most prominent measures among several centrality measures. Betweenness centrality of a vertex (edge) in a graph is a measure for the participation of the vertex (edge) in the shortest paths in the graph. It represents the relative importance of a vertex (edge) in the graph, and allows an understanding of the extent to which a vertex (edge) contributes in the flow of information.

Motivations and applications. Betweenness centrality is widely used in diverse applications across many different disciplines.

It is used to find the most prominent vertices in a complex network, whether they are individuals in a social network [15], elements in a biological network [19], intersections or junctions in a transportation network [20], physical elements in a computer network [18], or documents in a hyper-link network [50]. For example, in a social network, an individual with a higher centrality can be viewed as a more influential individual than an individual with a lower centrality. The importance and applications of betweenness centrality and its several variants are well explained in [13].

Although betweenness centrality problem has been extensively studied in the literature, most of existing studies did not address the problem of updating betweenness centrality. Currently, many real graphs such as social network graphs change over time [39]. In addition, although a graph is static itself, some algorithms iteratively alter the graph to achieve their objectives. For example, a community detection algorithm using betweenness centrality calculates betweenness centralities of edges in a graph iteratively, and removes the edge with the highest centrality until the graph is disconnected. Therefore, the need for updating betweenness centrality is evident. The difficulty on updating betweenness centrality was addressed by several researchers such

* Corresponding author. Tel.: +82 42 350 3537; fax: +82 42 350 3510.
E-mail addresses: mjlee@islab.kaist.ac.kr (M.-J. Lee), sunghee@kaist.edu (S. Choi), chungcw@kaist.edu (C.-W. Chung).

http://dx.doi.org/10.1016/j.ins.2015.07.053
0020-0255/

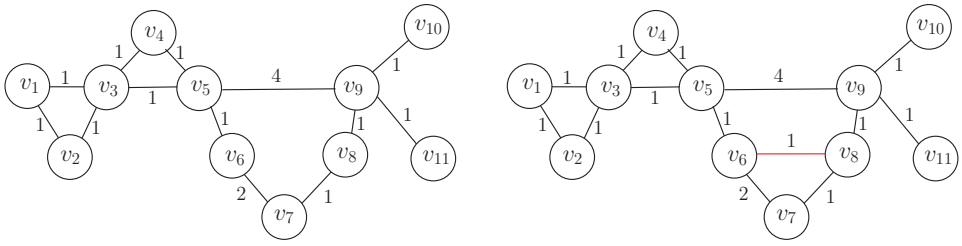

Fig. 1. Graph update example (numbers are edge weights).

as Brandes [13], Koschützki et al. [34] and Newman and Girvan [44]. However, none of them proposed a solution for fully dynamic graphs.[1]

Solution overview. Consider an edge insertion example in Fig. 1. We want to figure out which shortest paths are changed due to the insertion of edge $e(v_6, v_8)$. Let G be the original graph and G^U be the updated graph of G. It is trivial to see that the shortest path between v_6 and v_8 is changed. In addition, the shortest paths, which include the original shortest path between v_6 and v_8, are changed (e.g., the shortest path between v_5 and v_8). Let us assume that we have an index structure which stores all the shortest paths in the graph. Then we may easily identify such shortest paths. However, some shortest paths in the graph are changed although they did not include the original shortest path between v_6 and v_8. The shortest path between v_1 and v_{11} in G did not include the former shortest path between v_6 and v_8, but it is changed to include the inserted edge $e(v_6, v_8)$. Such shortest paths cannot be easily identified even if we store all the shortest paths in a graph in an index.

However, we observe that there exist vertices (edges) whose betweenness centralities are not changed although the shortest paths, which include the vertices (edges), are changed. In the above example, betweenness centrality of v_3 remains the same although the shortest path between v_1 and v_{11}, which includes v_3, is changed. This is because, only a part (v_5, v_9) of the shortest path not including v_3 is changed due to the update. In Fig. 1b, betweenness centralities of v_5, v_6, v_7, v_8 and v_9 are changed, while those of the other vertices are not changed.

Based on the above observation, we propose a new algorithm for updating betweenness centrality. The key idea of the proposed update algorithm is to perform betweenness centrality computation on a subgraph with vertices and edges whose betweenness centralities should be updated. We first find a subgraph of vertices and edges of which betweenness centralities can be changed due to the graph update. In Fig. 1b, betweenness centralities of vertices and edges in a subgraph induced by a set of vertices $\{v_5, v_6, v_7, v_8, v_9\}$ can be changed while those of other vertices and edges are not changed. Such a subgraph is called the *re-calculation subgraph*. However, computing the new betweenness centralities of vertices and edges of a *re-calculation subgraph* using the *re-calculation subgraph* only is insufficient since the following shortest paths are not yet considered. 1) The shortest paths of each of which the source or the target is not in the subgraph, and 2) the shortest paths which go through the subgraph. We propose a novel approach to identify a *re-calculation subgraph* and to compute amounts of increase in betweenness centralities of vertices and edges of a *re-calculation subgraph* due to the shortest paths 1) and 2) above.

In case of a vertex insertion, although it affects all the vertices and edges in the graph, we can effectively identify the amounts of betweenness centrality changes of all the vertices as follows. First, we separate the vertex to be inserted and its incident edges into a unit vertex, which has only one incident edge, and remaining edges. Second, insert the unit vertex to the graph. The amounts of betweenness centrality changes for inserting the unit vertex can be calculated by priority first traversing the graph from the unit vertex. This is because, all and only the newly created shortest paths in the graph are the single source shortest paths from the unit vertex. Then, insert remaining edges to the graph. The vertex deletion can be handled in a similar way.

Contributions. The contributions of this paper are as follows.

- We devise a novel algorithm for updating betweenness centrality in fully dynamic graphs. To the best of our knowledge, the proposed algorithm is the first work which deals with fully dynamic graphs.
- Our proposed algorithm efficiently updates betweenness centralities without re-computing all-pairs shortest paths in the entire graph. Moreover, this is the only work which does not require any structure to be maintained or pre-processed for updating betweenness centrality.
- Based on the proposed update algorithm, we also devise the highest betweenness centrality edge finding algorithm and a community detection algorithm.
- We conduct experiments on several real graphs. The experimental results show that the proposed algorithm outperforms existing algorithms for all update operations.

[1] A graph is fully dynamic if there are no limits on graph updates, i.e., all insertions and deletions of edges and vertices, and incremental and decremental edge weight changes are possible. In contrast, a graph is partially dynamic if only some of graph updates are possible.