Effective Co-betweenness Centrality Computation

Mostafa Haghir Chehreghani
Department of Computer Science, KU Leuven
Celestijnenlaan 200a - box 2402
3001 Leuven, Belgium
Mostafa.HaghirChehreghani@cs.kuleuven.be

ABSTRACT

Betweenness centrality of vertices is essential in the analysis of social and information networks, and *co-betweenness centrality* is one of two natural ways to extend it to sets of vertices. Existing algorithms for co-betweenness centrality computation suffer from at least one of the following problems: *i*) their applicability is limited to special cases like sequences, sets of size two, and *ii*) they are not efficient in terms of time complexity. In this paper, we present efficient algorithms for co-betweenness centrality computation of any set or sequence of vertices in weighted and unweighted networks. We also develop effective methods for co-betweenness centrality computation of sets and sequences of edges. Results of this paper, provide a clear and extensive view about the complexity of co-betweenness centrality computation for vertices and edges in weighted and unweighted networks. Finally, we perform extensive experiments on real-world networks from different domains including social and information, to show the empirical efficiency of the proposed method.

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*

General Terms

Theory

Keywords

Social networks, network analysis, centrality, betweenness centrality, co-betweenness centrality, algorithm, complexity.

1. INTRODUCTION

Centrality is a structural property of vertices in a network which determines the importance of a vertex (or a set of vertices) within the network [14]. *Betweenness centrality* of a vertex, introduced by Linton Freeman [13], is defined as the number of shortest paths (geodesic paths) from all vertices to all others that pass through that vertex. He used it as a measure for quantifying the control of a human on the communication between other humans in a social network [13].

There are several applications which benefit from a definition of centrality used for sets of vertices rather than individual vertices [12]. Betweenness centrality of vertices is extended to betweenness centrality of sets of vertices in two ways:

- *group betweenness centrality* of a set [12], which is defined as the number of shortest paths that pass through at least one of the vertices in the set, and
- *co-betweenness centrality* of a set [20], which is defined as the number of shortest paths that pass through all vertices in the set.

It has been shown that these two notions are intimately related [20]. The focus of this paper is *co-betweenness centrality*. In [20], the authors illustrated the utilization of co-betweenness centrality of sets of vertices in different social and communication networks and showed its interesting features in different real world networks. They also showed that co-betweenness centrality allows one to identify certain vertices which act as important actors in the relaying and destinating information in the network.

The following computation problems are defined for co-betweenness centrality.

PROBLEM 1. *Vertex co-betweenness centrality computation: given an unweighted graph (network)* $\mathbf{G}$ *and a set* $K \subseteq V(\mathbf{G})$*, compute co-betweenness centrality of* K.

Parameters of Problem 1 are n, m and $|K|$, where n and m are the number of vertices and the number of edges of the network, respectively.

PROBLEM 2. *Edge co-betweenness centrality computation: given an unweighted graph (network)* $\mathbf{G}$ *and a set* $W \subseteq E(\mathbf{G})$*, compute co-betweenness centrality of* W.

Similar to Problem 1, parameters of Problem 2 are n, m and $|W|$. The problems can be defined similarly, for sequences (where the order in which vertices in K (or edges in W) must be met by shortest paths is fixed), as well as for sets and sequences in weighted (valued) graphs.

Recently, a number of complexity results have been reported for co-betweenness centrality computation. For example,

- In [20], the authors presented an algorithm for vertex co-betweenness centrality computation. However, their method works only for sets of size 2 and its worst case time complexity is a $O(n^3)$.

WSDM'14, February 24–28, 2014, New York, New York, USA.

http://dx.doi.org/10.1145/2556195.2556263.

- The authors of [32], as a part of their algorithm for computing successive group betweenness centrality of sets of vertices, showed that co-betweenness centrality of a sequence consisting of two vertices is computable in $O(mn)$ time.
- In [11], the authors introduced the Routing Betweenness Centrality (RBC) index, which is a generalization of betweenness centrality. In this work, they studied complexity of RBC computation of sequences.

However, several problems like co-betweenness centrality computation of vertices, co-betweenness centrality computation of edges and co-betweenness centrality computation in weighted graphs, either have not been studied yet, or complexity of proposed algorithms is too high. In this paper [1]:

- We present an $O(nm-|K|m+n|K|\log|K|-|K|^2\log|K|)$ time algorithm for co-betweenness centrality computation of a set K of vertices in unweighted graphs. We show that it is possible to compute co-betweenness centrality of K in weighted graphs in $O(nm-|K|m+n^2\log n-|K|n\log n)$ time.
- We show that co-betweenness centrality of a sequence K of vertices is computable in $O(nm-|K|m)$ time in unweighted graphs and in $O(nm-|K|m+n^2\log n-|K|n\log n)$ time in weighted graphs.
- We present an $O(nm+n|W|\log|W|)$ time algorithm for co-betweenness centrality computation of a set W of edges in unweighted graphs. We show that it is possible to compute co-betweenness centrality of W in weighted graphs in $O(nm+n^2\log n)$ time.
- We show that co-betweenness centrality of a sequence W of edges is computable in $O(nm)$ time in unweighted graphs and in $O(nm+n^2\log n)$ time in weighted graphs.
- We perform extensive experiments on real-world networks from different domains including social, information, and collaboration, and show the empirical efficiency of the proposed method.

We show that both vertex co-betweenness centrality computation and edge co-betweenness centrality computation can reduce to betweenness centrality computation of a single vertex. As a result, increasing the size of the set does not necessarily increase complexity of co-betweenness centrality computation, and for sequences, computation of co-betweenness centrality for larger sequences is always easier than smaller ones.

A shortcoming of our proposed algorithms is that unlike the Brandes algorithm [7] which can calculate betweenness centrality of all vertices simultaneously, they need the set of vertices/edges as an input. However, we think this shortcoming is not important since checking all sets of vertices is usually needed when someone wants to find the minimum set of vertices with the highest betweenness centrality. While finding such a *prominent set* is crucial for group betweenness centrality [32], it is not of high importance for co-betweenness centrality. The reason is that as the size of the set increases, co-betweenness centrality decreases so that betweenness centrality of single vertices is always higher than co-betweenness centrality of their super-sets. In practice, rather than considering all sets of vertices, it might be more desirable to compute co-betweenness centrality of only a few sets, like core vertices of communities in social/information networks, or hubs in communication networks.

The rest of this paper is organized as follows. In Section 2, preliminaries and definitions related to co-betweenness centrality computation are given. In Section 3, we present an efficient algorithm for computing co-betweenness centrality of a set (or a sequence) of vertices in unweighted and weighted graphs. In Section 4, we investigate complexity of edge co-betweenness centrality computation for sets and sequences in weighted and unweighted networks. We empirically study the proposed methods in Section 5. In Section 6, we have a brief overview on related work. Finally, in Section 7, the paper is concluded.

2. PRELIMINARIES

In this section, we present definitions and notations widely used in the paper. We assume that the reader is familiar with basic concepts in graph theory. Throughout the paper, $\mathbf{G}$ refers to a graph (network). For simplicity, we assume that $\mathbf{G}$ is a connected graph without multi-edges. In Sections 3-3.1, $\mathbf{G}$ points to an unweighted graph and in Section 3.2, it points to a weighted graph with positive weights. $V(\mathbf{G})$ and $E(\mathbf{G})$ refer to the set of vertices and the set of edges of $\mathbf{G}$, respectively. Throughout the paper, n points to $|V(\mathbf{G})|$ and m points to $|E(\mathbf{G})|$. For an edge $e=(u,v)\in E(\mathbf{G})$, u and v are two end-points of e. A graph G' is a *subgraph* of G if $V(G')\subseteq V(G)$ and $E(G')\subseteq E(G)$. G' is an *induced subgraph* of G, if $V(G')\subseteq V(G)$ and $E(G')$ contains all edges of $E(G)$ which have both end-points in $V(G')$.

A *shortest path* (also called a *geodesic path*) between two vertices $u,v\in V(\mathbf{G})$ is a path whose size is minimum, among all paths between u and v. For two vertices $u,v\in V(\mathbf{G})$, we use $d_{\mathbf{G}}(u,v)$, to denote the size (the number of edges) of a shortest path connecting u and v. By definition, $d_{\mathbf{G}}(u,u)=0$ and $d_{\mathbf{G}}(u,v)=d_{\mathbf{G}}(v,u)$. For $s,t\in V(\mathbf{G})$, σ_{st} denotes the number of shortest paths between s and t. $\sigma_{st}(v)$ denotes the number of shortest paths between s and t that also pass through v. $\sigma_{st}(K)$, $K\subseteq V(\mathbf{G})$, denotes the number of shortest paths between s and t that also pass through all vertices in K. We have $\sigma_s(v)=\sum_{t\in V(\mathbf{G})\setminus\{s,v\}}\sigma_{st}(v)$ and $\sigma_s(K)=\sum_{t\in V(\mathbf{G})\setminus\{s\}\setminus K}\sigma_{st}(K)$.

Betweenness centrality of a vertex v is defined as:

$$\mathcal{B}(v)=\sum_{s,t\in V(G)\setminus\{v\}}\sigma_{st}(v) \tag{1}$$

Co-Betweenness centrality of a set $K\subset V(\mathbf{G})$ is defined as:

$$\mathcal{CB}(K)=\sum_{s,t\in V(G)\setminus K}\sigma_{st}(K) \tag{2}$$

We note that the number of shortest paths passing through a set of vertices can be exponential in terms of n.

A notion which is widely used for counting the number of shortest paths in a graph is the directed acyclic graph (DAG) containing all shortest paths starting from a vertex s (see e.g. [7]). In this paper, we refer to it as the *shortest-path-DAG*, or *SPD* for short, rooted at s. For every vertex s in a graph G, the *SPD* rooted at s is unique, and it can be computed in $O(m)$ time for unweighted graphs and in $O(m+n\log n)$ time for weighted graphs [7]. In the SPD D rooted at s, the *depth* of $v\in V(D)$, denoted $dep_D(v)$, is defined as $d_D(s,v)$.

[1]The definition of betweenness centrality and as a result co-betweenness centrality considered here is slightly different from the definition considered in [20] and [32]. While similar to [21] and [31], we define co-betweenness centrality as the *number* of shortest paths passing through a given set, in [20] and [32] it is defined as the *ratio* of shortest paths passing through a given set.

In [7], the authors introduced the notion of the *dependency score* of a vertex $s \in V(\mathbf{G})$ on a vertex $v \in V(\mathbf{G}) \setminus \{s\}$, which is defined as:

$$\delta_{s\bullet}(v) = \sum_{t \in V(\mathbf{G}) \setminus \{v,s\}} \sigma_{st}(v) \quad (3)$$

and then:

$$\mathcal{B}(v) = \sum_{s \in V(\mathbf{G}) \setminus \{v\}} \delta_{s\bullet}(v) \quad (4)$$

As discussed in [8], given the SPD rooted at s, $\delta_{s\bullet}(v)$ can be computed in $O(m)$.

3. COMPUTING CO-BETWEENNESS CENTRALITY OF A SET OF VERTICES

In this section, we study Problem 1. In [20], the authors introduced the problem and extended an $O(n^3)$ time algorithm for the special case of $|K| = 2$. We present a $O(nm)$ time algorithm for co-betweenness centrality computation of a set K of vertices. K can be a set of any size. First, in Lemma 1, we present a property for every SPD D containing at least one shortest path which passes through all the vertices in K. Then, we present a *pruned* subgraph of D with respect to K, called $\rho_K(D)$, and show that co-betweenness centrality computation of K can be reduced to betweenness centrality computation of a single vertex in $\rho_K(D)$.

LEMMA 1. *Let $s \in V(\mathbf{G})$, $K \subseteq V(\mathbf{G}) \setminus \{s\}$, and D be the SPD rooted at s. There exists a shortest path from s to some vertex $t \in V(\mathbf{G}) \setminus \{s\} \setminus K$ passing through all vertices in K, if for any two vertices $u, v \in K$, either u is an ancestor of v or v is an ancestor of u in D.*

PROOF. Let $s \to t$ be a shortest path passing through all vertices in K and without loss of generality, assume that in the SPD rooted at s, $s \to t$ passes through u before v. Let w be the vertex which is met by $s \to t$ after u. Obviously, w can not be a parent of u. On the other hand, in every SPD, two vertices with the same distance from the root are not connected to each other. Therefore, w is a child of u, and similarly, the next vertex will be a child of w, etc. After finite steps, $s \to t$ will reach v. Therefore, v is a descendant of u. □

Lemma 1 expresses the necessary condition for a source vertex s, to have at least one shortest path to other vertices in the graph passing through all vertices in K. This condition applies a total ordering on the members of K according to their distance from s which can be expressed in terms of the α operator introduced in Definition 1.

Definition 1. The operator $\alpha : K \to \{1 \dots |K|\}$ is defined with respect to a SPD D and maps an integer $i \in \{1 \dots |K|\}$ to a vertex $v \in K$, such that:

- $\forall i, j \in \{1, \dots, |K|\} : i \neq j \Leftrightarrow \alpha_{D,K}(i) \neq \alpha_{D,K}(j)$, and
- $\forall i, j \in \{1, \dots, |K|\} : i < j \Leftrightarrow \alpha_{D,K}(i)$ is an ancestor of $\alpha_{D,K}(j)$.

Definition 2. Let D be the SPD rooted at $s \in V(\mathbf{G})$ and $K \subset V(\mathbf{G})$. A vertex $v \in V(\mathbf{G})$ is *markable* in D if $v \notin K$ and at least one of the following conditions holds:

1. there exists a vertex $u \in K$ such that $dep_D(v) = dep_D(u)$,
2. if vertices satisfying condition (1) are removed from $\mathbf{G}$, in the resultant graph, v and s are not in the same component.

For example, in Figure 1.(b), vertices distinguished with a red '+' are *markable* vertices. Vertices $'3'$, $'7'$ and $'15'$ are markable because they satisfy condition (1) of Definition 2, and vertices $'6'$, $'9'$, $'10'$ and $'11'$ are markable because they satisfy condition (2) of Definition 2.

Definition 3. Let D be the SPD rooted at $s \in V(\mathbf{G})$ and $K \subset V(\mathbf{G})$. The *pruned DAG* of D with respect to K, denoted by $\rho_K(D)$, is the subgraph of D induced by $\{v \in V(D) | v \text{ is not markable}\}$.

Figure 1.(c) shows the pruned DAG of the SPD presented in 1.(b), with respect to $K = \{'2', '8', '14'\}$.

LEMMA 2. *Given a SPD D rooted at $s \in V(\mathbf{G})$ and $K \subseteq V(\mathbf{G})$, $\rho_K(D)$ is computable in $O(m)$.*

The following properties can be shown for $\rho_K(D)$:

- if for a set K of vertices, the condition of Lemma 1 is satisfied, none of the members of K are markable, and therefore, $K \subseteq V(\rho_K(D))$.
- if for a set K of vertices, the condition of Lemma 1 is satisfied, for every $i : 1 \leq i \leq |K|$, $\alpha_{D,K}(i) = \alpha_{\rho(D),K}(i)$. We sometimes write $\alpha_{D,K}(i)$ or $\alpha_{\rho(D),K}(i)$ simply as $\alpha_K(i)$.
- For $v \in V(D) \cap V(\rho(D))$, $dep_D(v) = dep_{\rho(D)}(v)$.

In the rest of this paper, we might talk about computing dependency scores either in a SPD D, or in its pruned form $\rho(D)$. To distinguish these two situations, we use $\delta_{s\bullet}(A, D)$ for dependency score computation in D and $\delta_{s\bullet}(A, \rho(D))$ for dependency score computation in $\rho(D)$. s is a vertex in the network and A is either a 'vertex' or 'a set of vertices' or 'a sequence of vertices'.

LEMMA 3. *Let $s \in V(\mathbf{G})$, $K \subseteq V(\mathbf{G}) \setminus \{s\}$, and D be the SPD rooted at s.*

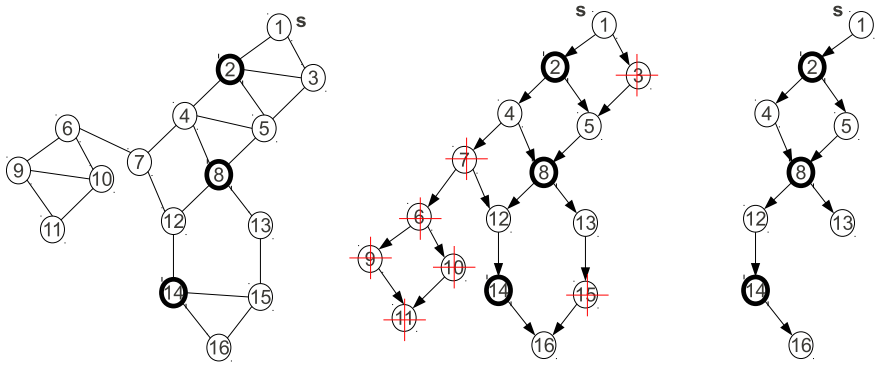
1. *if members of K are not in a single path of D, then: $\delta_{s\bullet}(K, D) = 0$*
2. *if all members of K are in a single path of D, then:*

$$\delta_{s\bullet}(K, D) = \delta_{s\bullet}(\alpha_K(|K|), \rho(D)) \quad (5)$$

i.e. $\delta_{s\bullet}(K, D)$ is equal to the number of shortest paths in $\rho(D)$ from s to other vertices which pass through $\alpha_K(|K|)$.

PROOF. Case (1) is directly resulted from Lemma 1. To prove case (2):

- First, we prove that for every shortest path in D starting form s and passing through all vertices in K, there exists a shortest path in $\rho(D)$ which starts from s and passes through $\alpha_K(|K|)$. Let $S = s_1 s_2 \dots s_l$ be a shortest path from $s_1 (= s)$ to s_l in D which passes through all vertices in K. As a contradiction, for any $1 < i \leq l$, suppose $s_i \notin V(\rho_K(D))$. Let $s_j \in S$ be the ancestor of s_i with the largest depth which belongs to $V(\rho_K(D))$. s_j always exists because s_1 is an ancestor of s_i and $s_1 \in V(\rho_K(D))$. s_j is not equal to s_l because otherwise, S is a path in $\rho_K(D)$. Now, consider s_{j+1}. s_{j+1} is not *markable* because:
  - there does not exist a $v \in K$ such that $dep_D(v) = dep_D(s_{j+1})$ because S passes through all members of K (and S can not pass through two vertices with the same depth). Therefore the first condition of Definition 2 does not hold for s_{j+1}.



a) **G** b) The SPD D rooted at s c) $\rho(D)$

Figure 1: Part (a) shows a network G. K is $\{'2', '8', '14'\}$, which are depicted with bold circles. Part (b) shows the SPD rooted at $s =' 1'$. In this graph, vertices with red '+' are *markable* vertices with respect to K. We have: $\alpha(1) =' 2'$, $\alpha(2) =' 8'$, $\alpha(3) =' 14'$. Part (c) shows $\rho_K(D)$. As Lemma 3 says, the number of shortest paths in G starting from s and passing through all $'2'$, $'8'$ and $'14'$, is equal to the number of shortest path in $\rho_K(D)$ starting from s and passing through $'14'$.

– s_j is not markable (since it belongs to $V(\rho_K(D))$) and s_{j+1} is connected to it, therefore, the second condition of Definition 2 does not hold for s_{j+1}.

Therefore, a contradiction occurs because s_{j+1} is an ancestor of s_i which belongs to $V(\rho_K(D))$ and $dep_D(s_{j+1}) > dep_D(s_j)$. So, s_i belongs to $V(\rho_K(D))$.

- Second, we prove that for every shortest path S in $\rho(D)$ which starts from s and passes through $\alpha_K(|K|)$, there exists a shortest path in D which starts form s and passes through all vertices in K. Since $V(\rho_K(D)) \subseteq V(D)$, vertices of S exist in D and we only need to show that S passes through all vertices in K. As a contradiction suppose that in D, S does not pass through at least one vertex $v \in K$. Then, there will exist a vertex u such that $u \notin K$, $dep_D(u) = dep_D(v)$ and S passes through u. However, according to the first condition of Definition 2, u is markable and it can not be in $\rho(D)$ and therefore, S can not pass through it. □

□

Lemma 3 says that if we check the property expressed in Lemma 1 for K, and then, prune the network in the way defined in Definition 3, instead of co-betweenness centrality computation of K, we can compute betweenness centrality of a single vertex. This means that co-betweenness centrality computation of larger sets is not necessarily more difficult that co-betweenness centrality computation of smaller ones.

For example, in Figure 1, $K = \{'2', '8', '14'\}$, vertices $'2'$,$'8'$ and $'14'$ are in a single path of the SPD rooted at $s =' 1'$ and $\alpha_K(1) =' 2'$, $\alpha_K(2) =' 8'$ and $\alpha_K(3) =' 14'$. $\alpha_K(|K|) =' 14'$ and We have: $\delta_{'1'\bullet,D}(\{'2', '8', '14'\}) = \delta_{'1'\bullet,\rho(D)}('14')$.

Algorithm 1 shows the high level pseudo code for co-betweenness centrality computation of a set K. For a vertex $s \in V(\mathbf{G}) \setminus K$:

- First, the SPD D rooted at s is formed. Using e.g. the method of [7], this step can be done in $O(m)$ time.
- Second, it is checked whether members of K are in a single path of D. To do so, the following steps are done:
  1. For every $v \in K$, $dep_D(v)$ is computed. This can be done in $O(m)$ time.
  2. Vertices in K are sorted increasingly, according to their *depth* in D. Using sorting algorithms like *merge sort* [19], this step can be done in $O(|K| \log|K|)$ time.
  3. If two or more vertices have the same *depth*, they are not in a single path. Time complexity of this check is $O(|K|)$.
  4. Let $v_1 \dots v_{|K|}$ be the members of K which are sorted according to their *depth* in D. For every j, $1 \leq j \leq |K|-1$, the following procedure is performed: the subgraph of D induced by

     $$\{v_j\} \cup \{u \in V(D) | u \textit{ is a descendant of } v_j\}$$

     is traversed in BFS, until either (*i*) v_{j+1} is met or, (*ii*) the traversal is finished. In case (*i*), v_{j+1} is a descendant of v_j and the procedure is performed for $j+1$. In case (*ii*), v_{j+1} is not a descendant of v_j and therefore, all members of K are not in a single path of D. Members of K are in a single path of D, if and only if for all j, case (*i*) occurs. This procedure is done in $O(m)$ time.

  Therefore, it can be decided in $O(m + |K| \log|K|)$ time whether all members of K are in a single path of D.
- Third, $\rho(D)$ is computed. As Lemma 2 says, it can be computed in $O(m)$.
- Forth, the dependency score of s on $\alpha_{D,K}(|K|)$ in $\rho(D)$ is computed. Using e.g. the method of [7], this step can be done in $O(m)$ time.

The above mentioned procedure is performed for every $s \in V(\mathbf{G}) \setminus K$. Therefore, we have the following theorem.

THEOREM 1. *Algorithm 1 computes co-betweenness centrality of a set $K \subset V(\mathbf{G})$ in $O(nm - |K|m + n|K| \log|K| - |K|^2 \log|K|)$ time.*

Algorithm 1 High level pseudo code of the algorithm of computing vertex co-betweenness centrality.

```
1: VertexCoBetweenness
2: Require. A network (graph) G, a set K ⊆ V(G).
3: Ensure. The co-betweenness centrality of K.
4: CB ← 0;
5: for all s ∈ V(G) \ K do
6:    Generate the SPD D rooted at s;
7:    if all vertices in K are in a single path of D then
8:       Compute ρ(D)
9:       cb ← δ_{s,•}(α_{D,K}(|K|), ρ(D));
10:      CB ← CB + cb;
11:   end if
12: end for
13: return CB;
```

3.1 Computing co-betweenness centrality of sequences

In this section, we investigate how the algorithm proposed for computation of co-betweenness centrality of sets can be revised to compute co-betweenness centrality of sequences.

Let $K = v_1, \ldots, v_k$ be the sequence of vertices for which we want to compute co-betweenness centrality. Let $s \in V(\mathbf{G})$ and D be the SPD rooted at s. There exists a shortest path from s to some other vertex $t \in V(\mathbf{G}) \setminus \{s\} \setminus K$ passing through all vertices in K in order (i.e. first v_1, second v_2 etc), if in addition to the condition presented in Lemma 1 (i.e. all v_is are in a single path of D), the following holds:

$$\forall i \in \{1, \ldots, k\} \Rightarrow v_i = \alpha_{D,K}(i) \tag{6}$$

which means the order of vertices in K corresponds to the order applied by the condition presented in Lemma 1. This condition makes computation of co-betweenness centrality easier; it is possible to check both conditions in $O(m)$ time as follows:

Let D be the SPD rooted at $s \in V(\mathbf{G}) \setminus K$. For every v_i, $i \in \{1, \ldots, |K| - 1\}$, the subgraph of D induced by $\{v_i\} \cup \{u \in V(D) : u \textit{ is a descendant of } v_i\}$ is traversed in BFS, until either (*i*) v_{i+1} is met, or (*i*) the traversal is finished. In case (*i*), v_{i+1} is a descendant of v_i. In case (*ii*), v_{i+1} is not a descendant of v_i, and therefore, the conditions do not hold. The conditions presented in Lemma 1 and Equation 6 hold if and only if for all v_is, case (*i*) occurs. The procedure can be done in $O(m)$ time, therefore it is possible to decide in $O(m)$ time whether the conditions are satisfied. In fact, for sequences we do not need to compute the depth of vertices in D and sort members of K according to their depth.

Similar to sets, in order to compute co-betweenness centrality of a sequence K, for every $s \in V(\mathbf{G}) \setminus K$, the following steps are done:

1. Form the SPD D rooted at s,
2. Check whether members of K satisfy the conditions presented in Lemma 1 and Equation 6,
3. Compute $\rho_K(D)$ (if the conditions hold),
4. Calculate $\delta_{s\bullet}(v_k, \rho_K(D))$ (if the conditions hold).

Co-betweenness centrality of K is the sum of the calculated dependency scores. All steps 1-4 can be done in $O(m)$ time, and they are performed for $n - |K|$ times, therefore, co-betweenness centrality of a sequence of vertices is computable in $O(nm - |K|m)$ time:

THEOREM 2. *Co-betweenness centrality of a sequence K of vertices in an unweighted graph can be computed in $O(nm - |K|m)$ time.*

3.2 Computation of co-betweenness centrality in weighted graphs

Algorithms proposed for unweighted graphs can be used to compute co-betweenness centrality in weighted graphs. However, compared to unweighted graphs, time complexity of steps like forming SPDs, is higher for weighted graphs. As Corollary 4 of [7] says, for a weighted graph $\mathbf{G}$, given a source $s \in V(\mathbf{G})$, the number of all shortest paths to other vertices can be determined in time $O(m + n \log n)$.

For vertex co-betweenness centrality computation, this time complexity overcomes time complexity of checking whether all members of the set (or the sequence) are in a single path. This means that for weighted graphs, it is possible to compute co-betweenness centrality of a set (or sequence) K of vertices in $O((n - |K|) \times (m + n \log n)) = O(nm - |K|m + n^2 \log n - |K|n \log n)$ time.

THEOREM 3. *For weighted graphs, co-betweenness centrality of a set (or a sequence) K of vertices is computable in $O(nm - |K|m + n^2 \log n - |K|n \log n)$ time.*

4. CO-BETWEENNESS CENTRALITY COMPUTATION OF SETS OF EDGES

Betweenness of edges, defined in terms of the number of shortest paths passing through an edge, has many interesting applications in network analysis. For example, Girvan and Newman [16] used it to detect communities in complex networks. Very recently, Cuzzocrea et al. [10] proposed a new topology-control algorithm in wireless sensor networks, called edge betweenness centrality, which is based on edge betweenness centrality.

Co-betweenness centrality of a set or a sequence of edges (*edge co-betweenness centrality*), is a generalization of this notion to a set or a sequence of edges. It has several applications in traffic network analysis, social networks etc. For example, an important problem in traffic data analysis is finding *hot routes* in a road network. A *hot route* is a sequence of edges which share (almost) the same amount of traffic. This problem has applications in city planning, real estate developing, etc and allows city-holders to better direct traffic or analyze congestion resources. Efficient algorithms for addressing this problem cluster sets of edges based on the density of common traffic they share [25]. With the assumption that movements are done through shortest paths, the basic element of this problem is *edge co-betweenness centrality* computation. A weight can be assigned to every edge (road) to reflect the number of trajectories passing through that edge. In the rest of this section, we first briefly formulate edge co-betweenness centrality, and then, we discuss about effective edge co-betweenness centrality computation.

4.1 Edge co-betweenness centrality

For $e \in E(\mathbf{G})$, $\sigma_{st}(e)$ denotes the number of shortest paths between s and t that also pass through e. We note that in this definition, s and t are allowed to be an end-point of e. $\sigma_{st}(W)$, $W \subseteq E(\mathbf{G})$, denotes the number of shortest paths between s and t that also pass (in any order) through all members of W. We have

$$\sigma_s(e) = \sum_{t \in V(\mathbf{G}) \setminus \{s\}} \sigma_{st}(e)$$

and

$$\sigma_s(W) = \sum_{t \in V(\mathbf{G}) \setminus \{s\}} \sigma_{st}(W)$$

Betweenness centrality of an edge e is defined as

$$\mathcal{B}(e) = \sum_{s,t \in V(G)} \sigma_{st}(e)$$

Co-Betweenness centrality of a set $W \subseteq E(\mathbf{G})$ is defined as:

$$\mathcal{CB}(W) = \sum_{s,t \in V(G)} \sigma_{st}(W)$$

We note that the number of shortest paths passing through a set of edges can be exponential in terms of n.

4.2 Computation of edge co-betweenness centrality

Lemma 4 expresses the necessary condition for a source vertex s, to have at least one shortest path to other vertices in the network which pass through all edges in W:

LEMMA 4. *Let $s \in V(\mathbf{G})$, $W \subseteq E(\mathbf{G})$ and D be the SPD rooted at s. There exists at least one shortest path from s to some other vertex in the network which pass through all edges in W, if vertices in L are in a single path of D, where L is*

$$\{v \in V(\mathbf{G}) \setminus \{s\} | v \text{ is an end-point of at least one } e \in W\} \quad (7)$$

PROOF. The proof is omitted because it is similar to the proof of Lemma 1. □

As a result of Lemma 4, we can define an ordering operator α, in a way similar to Definition 1, which applies a total ordering on the members of L, and therefore, on the members of W. Dependency score of $s \in V(\mathbf{G})$ on $W \subseteq E(\mathbf{G})$, i.e. the number of shortest paths from s to other vertices in the network which pass through all edges in W, is defined as $\delta_{s\bullet}(W) = \sum_{t \in V(\mathbf{G}) \setminus \{s\}} \sigma_{st}(W)$, and then $\mathcal{CB}(W) = \sum_{s \in V(\mathbf{G})} \delta_{s\bullet}(W)$.

Lemma 5 shows how 'edge co-betweenness centrality' of W is related to 'betweenness centrality' of a single vertex in L.

LEMMA 5. *Let $s \in V(\mathbf{G})$, $W \subseteq E(\mathbf{G})$ and D be the SPD rooted at s.*

1. *if $W \subseteq E(D)$ and members of L are in a single path of D, then:*

$$\delta_{s\bullet}(W, D) = \delta_{s\bullet}(\alpha_L(|L| - 1), \rho_L(D)) \quad (8)$$

   where L is defined in Equation 7.

2. *otherwise: $\delta_{s\bullet}(W, D) = 0$*

We note that in $\rho_L(D)$, $\alpha_L(|L| - 1, \rho_L(D))$ has always exactly one child which is $\alpha_L(|L|, \rho_L(D))$ and $\alpha_L(|L|, \rho_L(D))$ has always exactly one parent which is $\alpha_L(|L| - 1, \rho_L(D))$.

As an example, consider Figure 2. All members of W which is $\{('1', '2'), ('4', '8'), ('12', '14')\}$, are in a single path of the SPD D rooted at s. Furthermore, $W \subseteq E(D)$. Therefore, $\delta_{s\bullet}(W, D) = \delta_{s\bullet}('12', \rho(D))$

Algorithm 2 shows the high level pseudo code of co-betweenness centrality computation for a set of edges. It differs from Algorithm 1 in two aspects. First, we need to check whether $W \subseteq E(D)$ (Line 7 of Algorithm 2). This step can be done in $O(|W|)$ time since we can check in $O(1)$ time whether there exists an edge between two vertices of a graph G. Second, we need to perform the different steps (generating SPDs, checking whether $W \subseteq E(D)$, checking whether all members of K are in a single path of D, and computing dependency scores) for n times (the loop in Lines 5-14). We note that the second case increases time complexity of edge co-betweenness centrality, compared to vertex co-betweenness centrality computation.

THEOREM 4. *Algorithm 2 computes co-betweenness centrality of a set $W \subset E(\mathbf{G})$ in $O(nm + n|W| \log|W|)$ time.*

Algorithm 2 High level pseudo code of the algorithm of computing co-betweenness centrality of a set of edges.

```
1: EDGECOBETWEENNESS
2: Require. A network (graph) G, a set W ⊆ E(G).
3: Ensure. The co-betweenness centrality of W.
4: CB ← 0;
5: for all s ∈ V(G) do
6:    Generate the SPD D rooted at s;
7:    if W ⊆ E(D) then
8:       {let L be {v ∈ V(G) \ {s} | v is an end-point of at least one e ∈ W}}
9:       if all vertices in L are in a single path of D then
10:         cb ← δ_{s•}(α_{D,L}(|L| − 1), ρ_L(D));
11:         CB ← CB + cb;
12:      end if
13:   end if
14: end for
15: return CB;
```

Co-betweenness centrality of a sequence W of edges can be computed, as the following. For every vetex s in the graph:

1. form the SPD D rooted at s,
2. check whether $W \subset E(D)$,
3. check whether members of L are in a single path of D,
4. the condition expressed in item (3), applies a total ordering on the edges in W. Check whether this total ordering is consistent with the order of edges in the sequence.
5. calculate $\delta_{s\bullet}(\alpha_{D,L}(|L| - 1), \rho_L(D))$, if the conditions hold (L is defined in Equation 7).

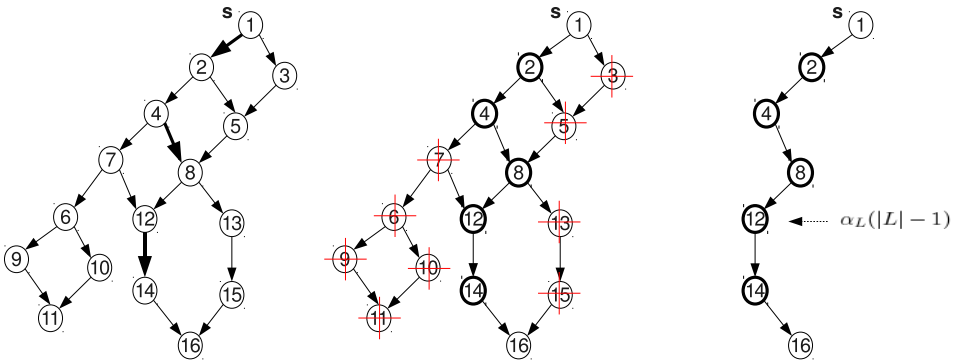
Similar to our discussion for vertex co-betweenness centrality computation, we can show that every step 1-5 can be done in $O(m)$ time. Therefore, we have the following theorem:

THEOREM 5. *Co-betweenness centrality of a sequence W of edges in an unweighted graph can be computed in $O(nm)$ time* [2].

Finally, for co-betweenness centrality computation of a set (or a sequence) of edges in weighted graphs, we present the following theorem, where the proof is omitted because it is similar to the case of vertex co-betweenness centrality computation in weighted graphs.

THEOREM 6. *For weighted graphs, co-betweenness centrality of a set (or a sequence) W of edges is computable in $O(nm + n^2 \log n)$ time.*

[2]A sequence of edges is also the sequence of vertices composing the edges. What increases time complexity of co-betweenness centrality computation for edges is the number of source vertices used to form SPDs. According to the definitions, a shortest path starting with (or ending to) a member of a sequence of vertices does not contribute to the co-betweenness centrality of the sequence. However, a shortest path starting with (or ending to) an end-point of a member of a sequence of edges contributes to co-betweenness centrality of the sequence.



b) The SPD D rooted at s b) The SPD D rooted at s c) $\rho(D)$

Figure 2: Part (a) shows the SPD rooted at $s =' 1'$. W is $\{('1','2'),('4','8'),('12','14')\}$ which are distinguished with bold lines. In part (b), members of $L = \{'2','4','8','12','14'\}$ are distinguished with bold circles and vertices with red '+' are *markable* vertices with respect to L. We have: $\alpha(1) =' 2'$, $\alpha(2) =' 4'$, $\alpha(3) =' 8'$, $\alpha(4) =' 12'$, $\alpha(5) =' 14'$. Part (c) shows $\rho_L(D)$.

Table 1: Summary of real-world networks.

Dataset	# vertices	# edges	Reference
Advogato	6,551	51,332	[27]
ego-facebook	2,888	2,981	[28]
caida	26,475	53,381	[22]
CA-HepTh	27,770	352,807	[23]
Eva	8,343	6,726	[30]
Odlis	2,909	18,419	[34]
Dolphins	62	159	[26]
dblp	12,591	49,793	[24]

5. EXPERIMENTAL RESULTS

We performed extensive experiments on real-world networks from different domains to assess the quantitative and qualitative behavior of the proposed algorithm. The experiments were done on one core of a single AMD Processor 270 clocked at 2.0 GHz with 8 GB main memory and 2×1 MB L2 cache, running Ubuntu Linux 12.0. The program was compiled by the GNU C++ compiler 4.0.2 using optimization level 3. Table 1 summarizes specifications of our real-world networks.

Due to lack of space, we only report the results obtained for co-betweenness centrality computation of a set of vertices. The results obtained for other problems, e. g. a set of edges, are very similar to the results reported here for a set of vertices. In our experiments, for every dataset, we select a random set P of vertices such that the subgraph induced by them is a path. Then, we start with a randomly chosen subset K of P (of size 2) and compute their co-betweenness centrality. Then, we add a randomly chosen vertex $v \in P \setminus K$ to K and compute co-betweenness centrality of the vertices in K. Adding new vertices to K is continued until co-betweenness centrality of all vertices in P is computed. We note that if the subgraph induced by K is a cyclic graph, its co-betweenness centrality will be 0.

Two well-known existing methods in the literature are the algorithm of Kolaczyk et. al. [20] and the algorithm of Dolev et. al. [11]. As mentioned earlier, the first one can only computes co-betweenness centrality of sets of size 2. On the other hand, the algorithm of Dolev et. al. can only compute co-betweenness centrality of sequences. Therefore, both algorithms are improper for our comparisons since they consider limited cases. Thus, we only report the results of our algorithm. Table 2 reports the empirical results.

The first dataset studied here is the *Advogato* network[3]. It is the trust network of the Advogato online community. Vertices are users of Advogato and edges represent trust relationships [27]. Clearly, by increasing the size of K, co-betweenness centrality decreases, i.e. for a subset K' of K, co-betweenness centrality of K' is not less than co-betweenness centrality of K. This is reflected in our experiments. On the other hand, over Advogato, as the size of K increases, the time required to compute co-betweenness centrality decreases. It shows the efficiency of the SPDs pruning techniques.

The next dataset is the *ego-facebook* network[4]. This network contains Facebook user-user friendships. A vertex represents a user and an edge indicates a friendship relationship [28]. As depicted in Table 2, over this dataset, co-betweenness centrality always decreases as the size of K increases. On the other hand, the running time usually decreases, as the size of K increases. The only exception is that the time required to compute co-betweenness centrality of $\{428, 10\}$ is less than the time required to compute co-betweenness centralities of $\{2897, 428, 10\}$ and $\{3428, 2897, 428, 10\}$. In these situations, while compared to smaller sets, forming pruned SPDs with respect to larger sets is much more expensive, shortest path counting over pruned SPDs of larger sets is not much more efficient. In other words, the time required to prune SPDs dominates the speed-up resulted by counting shortest paths over pruned SPDs.

The next dataset comes from autonomous systems of the Internet[5]. The *caida* network is the undirected network of autonomous systems of the Internet connected with each other from the CAIDA project, collected in 2007. Vertices are autonomous systems (AS) and edges are communications [22]. Over this dataset, the running time does not stay consistent as the size of K changes. For example, the running time increases when vertex 14835 is added to $\{3891, 5379, 81, 1\}$. The next dataset is CA-HepTh[6]. This collaboration network covers scientific collaborations between authors papers submitted to High Energy Physics - Theory category. If an

[3] http://konect.uni-koblenz.de/networks/advogato
[4] http://snap.stanford.edu/data/egonets-Facebook.html
[5] http://snap.stanford.edu/data/as-caida.html
[6] http://snap.stanford.edu/data/ca-HepTh.html

Table 2: Empirical results of our proposed algorithm on different datasets. K contains the vertex identifiers of the chosen vertices.

Dataset	Vertices in the set K	Running time (sec)	Co-betweenness centrality
Advogato	$\{428, 1\}$	822.643	662982499
	$\{2209, 428, 1\}$	546.88	152106265
	$\{5235, 2209, 428, 1\}$	533.351	76053118
	$\{5234, 5235, 2209, 428, 1\}$	469.635	38026559
ego-facebook	$\{428, 10\}$	238.184	1683935329
	$\{2897, 428, 10\}$	500.259	23371727
	$\{3428, 2897, 428, 10\}$	383.021	4475940
	$\{3999, 3428, 2897, 428, 10\}$	349.836	106570
	$\{4000, 3999, 3428, 2897, 428, 10\}$	361.781	85256
	$\{4001, 4000, 3999, 3428, 2897, 428, 10\}$	307.447	63942
	$\{4002, 4001, 4000, 3999, 3428, 2897, 428, 10\}$	303.489	42628
	$\{4003, 4002, 4001, 4000, 3999, 3428, 2897, 428, 10\}$	302.32	21314
caida	$\{5379, 81, 1\}$	6079.36	979172539
	$\{3891, 5379, 81, 1\}$	5897.89	4198841
	$\{14835, 3891, 5379, 81, 1\}$	6015.15	1050967
	$\{14834, 14835, 3891, 5379, 81, 1\}$	5998.22	263472
	$\{20972, 14834, 14835, 3891, 5379, 81, 1\}$	5237.81	16866
CA-HepTh	$\{5622, 7907, 5784, 2930\}$	1836.33	3793691022
	$\{3125, 5622, 79075784, 2930\}$	1284.3	3762578814
	$\{9457, 3125, 5622, 7907, 5784, 2930\}$	1219.95	3306837476
	$\{7289, 9457, 3125, 5622, 7907, 5784, 2930\}$	1071.67	2227068862
	$\{840, 7289, 9457, 3125, 5622, 7907, 5784, 2930\}$	1046.52	1115518431
Eva	$\{165, 0\}$	280.202	162635430
	$\{159, 165, 0\}$	332.485	118993976
	$\{2, 159, 165, 0\}$	220.67	8610082
	$\{287, 2, 159, 165, 0\}$	201.197	158516
	$\{4356, 287, 2, 159, 165, 0\}$	299.393	29440
	$\{4853, 4356, 287, 2, 159, 165, 0\}$	314.575	7728
	$\{4584, 4853, 4356, 287, 2, 159, 165, 0\}$	367.678	7544
	$\{4859, 4584, 4853, 4356, 287, 2, 159, 165, 0\}$	380.485	5152
	$\{4660, 4859, 4584, 4853, 4356, 287, 2, 159, 165, 0\}$	401.938	4600
	$\{4554, 4660, 4859, 4584, 4853, 4356, 287, 2, 159, 165, 0\}$	302.847	920
	$\{6239, 4554, 4660, 4859, 4584, 4853, 4356, 287, 2, 159, 165, 0\}$	407.922	368
Odlis	$\{2530, 117\}$	116.878	233058
	$\{2531, 2530, 117\}$	197.293	232872
	$\{1541, 2531, 2530, 117\}$	238.806	230268
	$\{1550, 1541, 2531, 2530, 117\}$	196.662	155310
	$\{339, 1550, 1541, 2531, 2530, 117\}$	189.471	144336
	$\{481, 339, 1550, 1541, 2531, 2530, 117\}$	54.8989	10602
	$\{270, 481, 339, 1550, 1541, 2531, 2530, 117\}$	46.7788	4278
	$\{155, 270, 481, 339, 1550, 1541, 2531, 2530, 117\}$	46.482	4092
	$\{2025, 155, 270, 481, 339, 1550, 1541, 2531, 2530, 117\}$	48.6372	2790
	$\{2430, 2025, 155, 270, 481, 339, 1550, 1541, 2531, 2530, 117\}$	51.9214	2604
Dolphins	$\{40, 0\}$	0.0431	13348
	$\{7, 40, 0\}$	0.033	355
	$\{54, 7, 40, 0\}$	0.0308	233
	$\{13, 54, 7, 40, 0\}$	0.0308	83
	$\{32, 13, 54, 7, 40, 0\}$	0.0302	38
dblp	$\{2, 135\}$	3812.74	3242271388
	$\{2, 135, 1377\}$	3556.46	1823783366

author i co-authored a paper with author j, an edge is drawn between vertices i and j [23]. Over this dataset, the running time always decreases, as the size of K increases.

Eva is a prototype system for extracting, visualizing, and analyzing corporate ownership information as a social network. Applying the system to the telecommunications and media industries, Norlen et al. [30] constructed an ownership network with 6,726 relationships among 8,343 companies[7]. In this network, an edge from company u to company v is drawn iff the company u is an owner of company v. Since we do not have ownership relationships for all companies, there will be companies without edges. Analysis shows that this network is highly clustered, with over 50% of all companies connected to one another in a single component. We tested our proposed algorithm over this ownership network. In this dataset, co-betweenness value of a set shows its impact on the ownership relations in the network. Our proposed algorithm scales well on this dataset, as sets with different sizes have close running times. By increasing the size of the set, its co-betweenness score decreases.

Odlis [34] is a hypertext reference system for users like library and information science professionals and university students and faculty. Users can add a new term to the system if they expect to encounter it in future or if they require to know its meaning. In the network representing this system, vertices are terms and an edge from term u to term v exists iff in the Odlis dictionary the term v is used to describe the meaning of term u [8]. As reflected in Table 2, over this network, in most cases, by increasing the size of the set the time required to compute its co-betweenness centrality decreases. In this network, the co-betweenness score of a set K shows the number of pairs of terms which use all terms in K to describe their meaning.

We also tested the proposed algorithm on a small dataset. The dolphins dataset [26] consists of an undirected graph of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand [9]. In this dataset, the co-betweenness value of a set K of dolphins shows the importance of all members in K with together in associations between dolphins. As depicted in Table 2, over this dataset, sets with different sizes have almost the same running time.

Finally, we performed experiments on the dblp dataset[10]. Each vertex in this network is a publication (paper, book, etc), and each edge represents a citation of a publication by another publication [24]. Using the proposed algorithm, we calculated co-betweenness scores of two sets of vertices $\{2, 135\}$ and $\{2, 135, 1377\}$. As presented in Table 2, it takes a shorter time to compute co-betweenness score of the larger set.

6. RELATED WORK

Betweenness centrality is widely used as a precise indicator for the information flow controlled by a vertex in social and information networks [35]. It assumes that information flow is done through shortest paths. Brandes [7] introduced new algorithms for computing betweenness centrality of a vertex, which is performed in $O(nm)$ and $O(nm + n^2 \log n)$ time for unweighted and weighted networks, respectively. Holme [17] showed that betweenness centrality of a vertex is highly correlated with the fraction of time that the vertex is occupied by the traffic of the network. Barthelemy [3] showed that many scale-free networks [1] have a power-law distribution of betweenness centrality. There are also several notions of betweenness centrality which are used to determine the structural prominence of Web pages [18] and [9].

There are several variations of betweennes centrality which are not designed for shortest path routing. *Flow betweenness* [15] equally considers all paths for routing. Borgatti [4] and [5] considered betweenness for all possible paths, as well as all possible trails, as well as walks (weighted inversely by length). He used numerical simulation to estimate the betweenness values. Newman [29] proposed *Random walk betweenness* which prefers shorter paths over the longer ones. In this work, Newman provided closed-form equations for the case of random traversal via walks. Borgatti [5] proposed a dynamic model-based view of centrality that focuses on the outcomes of vertices in a graph.

In [13], Freeman defined group betweenness centrality as a measure of homogeneity of betweenness of members. Everett and Borgatti [12] defined group betweenness centrality as a natural extension of betweenness centrality for sets of vertices. The other natural extension of betweenness centrality is *co-betweenness centrality* [20]. The authors of [20] presented an algorithm for individual co-betweenness centrality computation. However, this method works only for sets of size 2 and its time complexity is $O(n^3)$. Puzis et al. [32] proposed a $O(|K|^3)$ time algorithm for computation of successive group betweenness centrality. In [33], the authors presented two algorithms for finding the most prominent group. The first algorithm is based on heuristic search and the second is based on iterative greedy choice of vertices. In [11], the authors defined the Routing Betweenness Centrality (RBC) measure and presented algorithms for computing RBC of single vertices in the network and algorithms for computing group RBC of sets (or sequences) of vertices. Ballester et al. [2] discussed the importance of finding the key group in a criminal network. Borgatti elaborated in [6] on a Key Player Problem (KPP) that is strongly related to the cohesion of a network. He defined two problems: KPP-Pos and KPP-Neg. The solution of the first problem is a group maximally connected to all other vertices in a graph and the solution of the second is a group maximally disrupting the network.

Table 4: Time complexity of edge co-betweenness centrality computation ($|W|$ is the size of the set (sequence)).

	Unweighted graphs	Weighted graphs
Set	$O(nm + n\|W\| \log\|W\|)$	$O(nm + n^2 \log n)$
Sequence	$O(nm)$	$O(nm + n^2 \log n)$

7. CONCLUSION

In this paper, we developed efficient algorithms for vertex and edge co-betweenness centrality computation in different settings. Tables 3 and 4 summarize time complexity of the proposed methods. Our methods are based on reducing *vertex co-betweenness centrality* and *edge co-betweenness centrality* of a set (or a sequence) to *betweenness centrality* of a single vertex. They are not limited to special cases and they are more efficient than existing methods. We empirically evaluated the efficiency of the proposed methods and showed their high performance over different real-world networks.

8. REFERENCES

[1] A.-L. Barabasi and R. Albert, Emergence of scaling in random networks, Science, 286, 509-512, 1999.

[7] http://vlado.fmf.uni-lj.si/pub/networks/data/econ/Eva/Eva.htm

[8] http://vlado.fmf.uni-lj.si/pub/networks/data/dic/odlis/Odlis.htm

[9] http://www-personal.umich.edu/~mejn/netdata/

[10] http://konect.uni-koblenz.de/networks/dblp-cite

Table 3: Time complexity of vertex co-betweenness centrality computation ($|K|$ is the size of the set (sequence)).

	Unweighted graphs	Weighted graphs
Set of vertices	$O(nm - \|K\|m + n\|K\| \log\|K\| - \|K\|^2 \log\|K\|)$	$O(nm - \|K\|m + n^2 \log n - \|K\|n \log n)$
Sequence of vertices	$O(nm - \|K\|m)$	$O(nm - \|K\|m + n^2 \log n - \|K\|n \log n)$

[2] C. Ballester, A. C. Armengol and Y. Zenou, Who's who in networks. wanted: The key player, Econometrica, 74(5), 1403-1417, Sep. 2006.

[3] M. Barthelemy, Betweenness centrality in large complex networks. The Europ. Phys. J. B - Condensed Matter 38, 2(Mar.), 163-168, 2004.

[4] S. P. Borgatti, Types of network flows and how to destabilize terrorist networks, Sunbelt International Social Networks Conference, New Orleans, 2002.

[5] S. P. Borgatti, Centrality and network flow, Social Networks, 27(1), 55-71, 2005.

[6] S. P. Borgatti, Identifying sets of key players in a social network, Comput. Math. Organ. Theory, 12(1), 21-34, 2006.

[7] U. Brandes, A Faster Algorithm for Betweenness Centrality. Journal of Mathematical Sociology, 25(2), 163-177, 2001.

[8] U. Brandes, On variants of shortest-path betweenness centrality and their generic computation, Social Networks, 30(2), 136-145, May 2008.

[9] S. Brin, R. Motwani, L. Page and T. Winograd, What can you do with a web in your pocket? IEEE Bulletin of the Technical Committee on Data Engineering, 21(2), 37-47, 1998.

[10] A. Cuzzocrea, A. Papadimitriou, D. Katsaros and Y. Manolopoulos, Edgebetweenness centrality: A novel algorithm for QoS-based topology control over wireless sensor networks, Journal of Network and Computer Applications, 35(4), 1210-1217, July 2012.

[11] S. Dolev, Y. Elovici and R. Puzis, Routing betweenness centrality, J. ACM, 57(4), 2010.

[12] M. Everett and S. Borgatti. The centrality of groups and classes, Journal of Mathematical Sociology, 23(3), 181-201, 1999.

[13] L. C. Freeman, A set of measures of centrality based upon betweenness, Sociometry, 40, 35-41, 1977.

[14] L. C. Freeman, Centrality in social networks: Conceptual clarification. Social Networks, 1(3), 215-239, 1979.

[15] L. C. Freeman, S. P. Borgatti, and D. R. White, Centrality in valued graphs: A measure of betweenness based on network ïňĆow, Social Networks, 13(2), 141-154, 1991.

[16] M. Girvan and M. E. J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99, 7821-7826, 2002.

[17] P. Holme, Congestion and centrality in traffic flow on complex networks, Adv. Complex Syst, 6(2), 163-176, 2003.

[18] J. M. Kleinberg, Authoritative sources in a hyperlinked environment, Journal of the Association for Computing Machinery, 46(5), 604-632, 1999.

[19] D. Knuth, Section 5.2.4: Sorting by Merging, Sorting and Searching. The Art of Computer Programming 3 (2nd ed.), Addison-Wesley, 158-168, 998.

[20] E. D. Kolaczyk, D. B. Chua and M. Barthelemy. Group-betweenness and co-betweenness: Inter-related notions of coalition centrality, Social Networks, 31(3), 190-203, 2009.

[21] S. Lammer, B. Gehlsen, D. Helbing. Scaling laws in the spatial structure of urban road networks. Physica A, 363(1), 89-95, 2006.

[22] J. Leskovec, J. Kleinberg and C. Faloutsos, Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations, KDD, 2005.

[23] J. Leskovec, J. Kleinberg and C. Faloutsos, Graph Evolution: Densification and Shrinking Diameters, ACM Transactions on Knowledge Discovery from Data (ACM TKDD), 1(1), 2007.

[24] M. Ley, The DBLP computer science bibliography: Evolution, research issues, perspectives, In Proc. Int. Symposium on String Processing and Information Retrieval, 1-10, 2002.

[25] X. Li, J. Han, J.-G. Lee and H. Gonzalez, Traffic density-based discovery of hot routes in road networks, 10th Symposium on Spatial and Temporal Databases (SSTD), 441-459, 2007.

[26] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations, Behavioral Ecology and Sociobiology 54, 396-405, 2003.

[27] P.Massa, M. Salvetti and D. Tomasoni, Bowling alone and trust decline in social network sites, In Proc. Int. Conf. Dependable, Autonomic and Secure Computing, 658-663, 2009.

[28] J. McAuley and J. Leskovec, Learning to discover social circles in ego networks, NIPS, pages 548-556. 2012.

[29] M. E. J. Newman, A measure of betweenness centrality based on random walks, Social Networks, 27(1), 39-54, 2005.

[30] K. Norlen, G. Lucas, M. Gebbie and J. Chuang, EVA: Extraction, Visualization and Analysis of the Telecommunications and Media Ownership Network. Proceedings of International Telecommunications Society 14th Biennial Conference (ITS2002), Seoul Korea, August 2002.

[31] A. Perer, B. Shneiderman, Balancing systematic and flexible exploration of social networks, IEEE Transactions on Visualization and Computer Graphics, 12(5), 693-700, 2006.

[32] R. Puzis, Y. Elovici and S. Dolev. Fast algorithm for successive computation of group betweenness centrality, Phys. Rev. E, 76(5), 2007.

[33] R. Puzis, Y. Elovici and S. Dolev, Finding the most prominent group in complex networks, AI Commun. 20(4), 287-296, 2007.

[34] J. M. Reitz, ODLIS: Online Dictionary of Library and Information Science, 2002.

[35] G. Yan, T. Zhou, B. Hu, Z. Q. Fu, and B.-H. Wang, Efficient routing on complex networks. Phys. Rev. E, 73, 046108, 2006.