Scalable Betweenness Centrality Maximization via Sampling

Ahmad Mahmoody
Brown University
Providence, RI 02912
ahmad@cs.brown.edu

Charalampos E. Tsourakakis
Harvard University
Cambridge, MA 02138
babis@seas.harvard.edu

Eli Upfal
Brown University
Providence, RI 02912
eli@cs.brown.edu

ABSTRACT

Betweenness centrality (BWC) is a fundamental centrality measure in social network analysis. Given a large-scale network, how can we find the most central nodes? This question is of great importance to many key applications that rely on BWC, including community detection and understanding graph vulnerability. Despite the large amount of work on scalable approximation algorithm design for BWC, estimating BWC on large-scale networks remains a computational challenge.

In this paper, we study the *Centrality Maximization* problem (CMP): given a graph $G = (V, E)$ and a positive integer k, find a set $S^* \subseteq V$ that maximizes BWC subject to the cardinality constraint $|S^*| \leq k$. We present an efficient randomized algorithm that provides a $(1 - 1/e - \epsilon)$-approximation with high probability, where $\epsilon > 0$. Our results improve the current state-of-the-art result [40]. Furthermore, we provide the first theoretical evidence for the validity of a *crucial* assumption in betweenness centrality estimation, namely that in real-world networks $O(|V|^2)$ shortest paths pass through the top-k central nodes, where k is a constant. This also explains why our algorithm runs in near linear time on real-world networks. We also show that our algorithm and analysis can be applied to a wider range of centrality measures, by providing a general analytical framework.

On the experimental side, we perform an extensive experimental analysis of our method on real-world networks, demonstrate its accuracy and scalability, and study different properties of central nodes. Then, we compare the sampling method used by the state-of-the-art algorithm with our method. Furthermore, we perform a study of BWC in time evolving networks, and see how the centrality of the central nodes in the graphs changes over time. Finally, we compare the performance of the stochastic Kronecker model [28] to real data, and observe that it generates a similar growth pattern.

KDD '16, August 13 - 17, 2016, San Francisco, CA, USA
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4232-2/16/08…$15.00
DOI: http://dx.doi.org/10.1145/2939672.2939869

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

Keywords

Social Network, Centrality, Sampling, Optimization

1. INTRODUCTION

Betweenness centrality (BWC) is a fundamental measure in network analysis, measuring the effectiveness of a vertex in connecting pairs of vertices via shortest paths [16]. Numerous graph mining applications rely on betweenness centrality, such as detecting communities in social and biological networks [20] and understanding the capabilities of an adversary with respect to attacking a network's connectivity [23]. The betweenness centrality of a node u is defined as

$$B(u) = \sum_{s,t} \frac{\sigma_{s,t}(u)}{\sigma_{s,t}},$$

where $\sigma_{s,t}$ is the number of s-t shortest paths, and $\sigma_{s,t}(u)$ is the number of s-t shortest paths that have u as their internal node. However, in many applications, e.g. [20, 23], we are interested in centrality of sets of nodes. For this reason, the notion of BWC has been extended to sets of nodes [22, 40]. For a set of nodes $S \subseteq V$, we define the betweenness centrality of S as

$$B(S) = \sum_{s,t \in V} \frac{\sigma_{s,t}(S)}{\sigma_{s,t}},$$

where $\sigma_{s,t}(S)$ is the number of s-t shortest paths that have an internal node in S. Note that we cannot obtain $B(S)$ from the values $\{B(v), v \in S\}$. In this work, we study the *Centrality Maximization problem* (CMP) defined formally as follows:

Definition 1 (CMP). *Given a network $G = (V, E)$ and a positive integer k, find a subset $S^* \subseteq V$ such that*

$$S^* \in \operatorname*{arg\,max}_{S \subseteq V : |S| \leq k} B(S).$$

We also denote the maximum centrality of a set of k nodes by $\texttt{OPT}_k$, i.e., $\texttt{OPT}_k = \max_{S \subseteq V : |S| \leq k} B(S)$.

It is known that CMP is APX-complete [15]. The best deterministic algorithms for CMP rely on the fact that BWC is monotone-submodular and provide a $(1-1/e)$-approximation [15, 12]. However, the running time of these algorithms is at least quadratic in the input size, and do not scale well to large-scale networks.

Finding the most central nodes in a network is a computationally challenging problem that we are able to handle accurately and efficiently. In this paper we focus on scalability of CMP, and graph mining applications. Our main contributions are summarized as follows.

Efficient algorithm. We provide a randomized approximation algorithm, HEDGE, based on sampling shortest paths, for accurately estimating the BWC and solving CMP. Our algorithm is simple, scales gracefully as the size of the graph grows, and improves the previous result [40], by (i) providing a $(1-1/e-\epsilon)$-approximation, and (ii) smaller sized samples. Specifically, in Yoshida's algorithm [40], a sample contains all the nodes on "any" shortest path between a pair, whereas in our algorithm, each sample is just a set of nodes from a single shortest path between the pair.

The $\mathtt{OPT}_k = \Theta(n^2)$ assumption. Prior work on BWC estimation strongly relies on the assumption that $\mathtt{OPT}_k = \Theta(n^2)$ for a constant integer k [40]. As we show in Appendix, this assumption is not true in general. Only empirical evidence so far supports this strong assumption.

We show that two broad families of networks satisfy this assumption: bounded treewidth networks and a popular family of stochastic networks that provably generate scale-free, small-world graphs with high probability. Note that the classical Barabási-Albert scale-free random tree model [5, 30] belongs to the former category. Our results imply that the $\mathtt{OPT}_k = \Theta(n^2)$ assumption holds even for $k=1$, for these families of networks. To our knowledge, this is the first theoretical evidence for the validity of this *crucial* assumption on real-world networks.

General analytical framework. To analyze our algorithm, HEDGE, we provide a general analytical framework based on Chernoff bound and submodular optimization, and show that it can be applied to any other centrality measure if it (i) is monotone-submodular, and (ii) admits a hyper-edge sampler (defined in Sect. 3). Two examples of such centralities are the *coverage* [40] and the *κ-path* centralities [2].

Experimental evaluation. We provide an experimental evaluation of our algorithm that shows that it scales gracefully as the graph size increases and that it provides accurate estimates. We also provide a comparison between the method in [40] and our sampling method.

Applications. Our scalable algorithm enables us to study some interesting characteristics of the central nodes. In particular, if S is a set of nodes with high BWC, we focus to answer the following questions.

(1) How does the centrality of the most central set of nodes change in time-evolving networks? We study the DBLP and the AUTONOMOUS systems graphs. We mine interesting growth patterns, and we compare our results to stochastic Kronecker graphs, a popular random graph model that mimics certain aspects of real-world networks. We observe that the Kronecker graphs behave similarly to real-world networks.

(2) Influence maximization has received a lot of attention since the seminal work of Kempe *et al.* [25]. Using our scalable algorithm we can compute a set of central nodes that can be used as seeds for influence maximization. We find that betweenness centrality is performing relatively well compared to a state-of-the-art influence maximization algorithm.

(3) We study four strategies for attacking a network using four centrality measures: betweenness, coverage, κ-path, and triangle centrality. Interestingly, we find that the κ-path and triangle centralities can be more effective at destroying the connectivity of a graph.

2. RELATED WORK

Centrality measures. There exists a wide variety of centrality measures: degree centrality, Pagerank [33], HITS [26], Salsa [27], closeness centrality [6], harmonic centrality [7], betweenness centrality [16], random walk betweenness centrality [32], coverage centrality [40], κ-path centrality [2], Katz centrality [24], rumor centrality [37] are some of the important centrality measures. Boldi and Vigna proposed an axiomatic study of centrality measures [7]. In general, choosing a good centrality measure is application dependent [19]. In the following we discuss in further detail the centrality measure of our focus, the betweenness centrality.

Betweenness centrality (BWC) is a fundamental measure in network analysis. The betweenness centrality index is attributed to Freeman [16]. BWC has been used in a wide variety of graph mining applications. For instance, Girvan and Newman use BWC to find communities in social and biological networks [20]. In a similar spirit, Iyer *et al.* use BWC to attack the connectivity of networks by iteratively removing the most central vertices [23].

The fastest known exact algorithm for computing BWC exactly requires $O(mn)$ time in unweighted, and $O(nm + n^2 \log m)$ for weighted graphs [10, 14, 36]. There exist randomized algorithms [4, 9, 34] which provide either additive error or multiplicative error guarantees with high probability.

For CMP, the state-of-the-art algorithm [40] (and the only scalable proposed algorithm based on sampling) provides a mixed error guarantee, combining additive and multiplicative error. Specifically, this algorithm provides a solution whose centrality is at least $(1-\frac{1}{e})\mathtt{OPT}_k - \epsilon n^2$, by sampling $O(\log n/\epsilon^2)$ *hyper-edges*, where each hyper-edge is a set of all nodes on any shortest path between two random nodes with some assigned weights.

As we mentioned before, CMP is APX-complete, and the best algorithm (i.e. classic greedy algorithm for maximizing the monotone-submodular functions) using *exact* computations of BWC provides $(1-1/e)$-approximation [15]. We call this greedy algorithm by EXHAUST algorithm, and works as follows: It starts with an empty set S. Then, at any round, it selects a node u that maximizes the *adaptive betweenness centrality* (A-BWC) of u according to S defined as

$$B(u|S) = \sum_{(s,t),s,t\neq u} \frac{\sigma_{s,t}(u|S)}{\sigma_{s,t}},$$

where $\sigma_{st}(u|S)$ is the number of s-t shortest paths that do not pass through any node in S and have u as an internal node. The algorithm adds u to S and stops when $|S| = k$.

Note that the A-BWC is intimately connected to the BWC through the following well-known formula [40]:

$$B(S \cup \{u\}) = B(S) + B(u|S).$$

3. ALGORITHM

In this section we provide our algorithm, HEDGE (Hyper-EDge GrEedy), and a general framework for its analysis. We start by defining a *hyper-edge sampler*, that will be used in HEDGE.

Definition 2 (Hyper-edge sampler). *We say that an algorithm $\mathcal{A}$ is a **hyper-edge sampler** for a function $C : 2^V \to \mathbb{R}$ if it outputs a randomly generated subset of nodes $h \subseteq V$ such that*

$$\forall S \subseteq V : \quad Pr_{h\sim\mathcal{A}}(h \cap S \neq \emptyset) = \frac{1}{\alpha}C(S),$$

*where α is a normalizing factor, and independent of the set S. We call each h (sampled by $\mathcal{A}$) a random hyper-edge, or in short, a **hyper-edge**. In this case, we say C **admits** a hyper-edge sampler.*

Our proposed algorithm HEDGE assumes the existence of a hyper-edge sampler and uses it in a black-box manner. Namely, HEDGE is oblivious to the specific mechanics of the hyper-edge sampler. The following lemma provides a simple hyper-edge sampler for BWC.

Lemma 1. *The BWC admits a hyper-edge sampler.*

Proof. Let $\mathcal{A}$ be an algorithm that selects two nodes $s, t \in V$ uniformly at random, selects a s-t shortest path P uniformly at random (this can be done in linear time $O(m + n)$ using bread-first-search from s, counting the number of shortest paths from s and backward pointers; e.g. see [34]), and finally outputs the internal nodes of P (i.e., the nodes of P except s and t).

Now, suppose h is an output of $\mathcal{A}$. Since the probability of choosing each pair is $\frac{1}{n(n-1)}$, and for a given pair s, t the probability of $S \cap h \neq \emptyset$ is $\frac{\sigma_{s,t}(S)}{\sigma_{s,t}}$, for every $S \subseteq V$ we have

$$\Pr_{h\sim\mathcal{A}}(h \cap S \neq \emptyset) = \sum_{s,t\in V} \frac{1}{n(n-1)}\frac{\sigma_{s,t}(S)}{\sigma_{s,t}} = \frac{1}{n(n-1)}B(S).$$

Also note that in this case, the normalizing factor is $\alpha = n(n-1) = \Theta(n^2)$. □

For a subset of nodes $S \subseteq V$, and a set $\mathcal{H}$ of independently generated hyper-edges, denote

$$\deg_{\mathcal{H}}(S) = |\{h \in \mathcal{H} \mid h \cap S \neq \emptyset\}|.$$

The pseudocode of our proposed algorithm HEDGE is given in Algorithm 1. First, it samples q hyper-edges using the hyper-edge sampler $\mathcal{A}$ and then it runs a natural greedy procedure on $\mathcal{H}$.

Algorithm 1: HEDGE

```
Input: A hyper-edge sampler $\mathcal{A}$ for BWC, number of hyper-edges $q$, and the size of the output set $k$.
Output: A subset of nodes, $S$ of size $k$.
begin
    $\mathcal{H} \leftarrow \emptyset$;
    for $i \in [q]$ do
        $h \sim \mathcal{A}$ (sample a random hyper-edge);
        $\mathcal{H} \leftarrow \mathcal{H} \cup \{h\}$;
    $S \leftarrow \emptyset$ ;
    while $|S| < k$ do
        $u \leftarrow \arg\max_{v\in V} \deg_{\mathcal{H}}(\{v\})$;
        $S \leftarrow S \cup \{u\}$;
        for $h \in \mathcal{H}$ such that $u \in h$ do
            $\mathcal{H} \leftarrow \mathcal{H} \setminus \{h\}$;
    return $S$;
```

3.1 Analysis

In this section we provide our general analytical framework for HEDGE, which works with any hyper-edge sampler. To start, define $B_{\mathcal{H}}(S) = \frac{\alpha}{|\mathcal{H}|}\deg_{\mathcal{H}}(S)$ as the centrality (BWC) of a set S according to the sample $\mathcal{H}$ of hyper-edges, and for a graph G let

$$q(G, \epsilon) = \frac{3\alpha(\ell + k)\log(n)}{\epsilon^2 \texttt{OPT}_k},$$

where n is the number of nodes in G, and ℓ is a positive integer. We have the following lemma:

Lemma 2. *Let $\mathcal{H}$ be a sample of independent hyper-edges such that $|\mathcal{H}| \geq q(G, \epsilon)$. Then, for all $S \subseteq V$ where $|S| \leq k$ we have $Pr(|B_{\mathcal{H}}(S) - B(S)| \geq \epsilon \cdot \texttt{OPT}_k) < n^{-\ell}$.*

Proof. Suppose $S \subseteq V$ and $|S| \leq k$, and let X_i be a binary random variable that indicates whether the i-th hyper-edge in $\mathcal{H}$ intersects with S. Notice that $deg_{\mathcal{H}}(S) = \sum_{i=1}^{|\mathcal{H}|} X_i$ and by the linearity of expectation $\mathbb{E}(deg_{\mathcal{H}}(S)) = |\mathcal{H}| \cdot \mathbb{E}(X_1) = \frac{q}{\alpha}B(S)$. Using the independence assumption and the Chernoff bound, we obtain:

$$\begin{aligned}
&\Pr(|B_{\mathcal{H}}(S) - B(S)| \geq \delta \cdot B(S)) &&= \\
&\Pr\left(\left|\frac{q}{\alpha}B_{\mathcal{H}}(S) - \frac{q}{\alpha}B(S)\right| \geq \frac{\delta q}{\alpha} \cdot B(S)\right) &&= \\
&\Pr(|deg_{\mathcal{H}}(S) - \mathbb{E}(deg_{\mathcal{H}}(S))| \geq \delta \cdot \mathbb{E}(deg_{\mathcal{H}'}(S))) &&\leq \\
&2\exp\left(-\frac{\delta^2}{3}\frac{q}{\alpha}B(S)\right).
\end{aligned}$$

Now, by letting $\delta = \frac{\epsilon \texttt{OPT}_k}{B(S)}$ and substituting the lower bound for $q(G, \epsilon)$ we obtain

$$\Pr(|B_{\mathcal{H}}(S) - B(S)| \geq \epsilon\texttt{OPT}_k) \leq n^{-(\ell+k)},$$

and by taking a union bound over all possible subsets $S \subseteq V$ of size k we obtain $|B_{\mathcal{H}}(S) - B(S)| < \epsilon \cdot \texttt{OPT}_k$ with probability at least $1 - 1/n^{\ell}$, for all such subsets S. □

Now, the following theorem shows that if the number of samples, i.e. $|\mathcal{H}|$, is at least $q(G, \epsilon/2)$, then HEDGE provides a $(1 - 1/e - \epsilon)$-approximate solution.

Theorem 1. *If $\mathcal{H}$ is a sample of at least $q(G, \epsilon/2)$ hyper-edges for some $\epsilon > 0$, and S is the output of HEDGE, we have $B(S) \geq (1 - 1/e - \epsilon)\texttt{OPT}_k$, with high probability.*

Proof. Note that B is (i) monotone since if $S_1 \subseteq S_2$ then $B(S_1) \leq B(S_2)$, and (ii) submodular since if $S_1 \subseteq S_2$ and $u \in V \setminus S_2$ then $B(S_2 \cup \{u\}) - B(S_2) \leq B(S_1 \cup \{u\}) - B(S_1)$.

Similarly, $B_{\mathcal{H}}$ is monotone and submodular. Therefore, using the greedy algorithm (second part of HEDGE) we have (see [31])

$$B_{\mathcal{H}}(S) \geq (1 - 1/e)B_{\mathcal{H}}(S') \geq (1 - 1/e)B_{\mathcal{H}}(S^*),$$

where

$$S' = \arg\max_{T:|T|\leq k} B_{\mathcal{H}}(T), \quad \text{and} \quad S^* = \arg\max_{T:|T|\leq k} B(T).$$

Notice that $\texttt{OPT}_k = B(S^*)$. Since $|\mathcal{H}| \geq q(G, \epsilon/2)$, by Lemma 2 with probability $1 - \frac{1}{n^{\ell}}$ we have

$$B(S) \geq B_{\mathcal{H}}(S) - \frac{\epsilon}{2}\texttt{OPT}_k \geq \left(1-\frac{1}{e}\right) B_{\mathcal{H}}(S^*) - \frac{\epsilon}{2}\texttt{OPT}_k$$
$$\geq \left(1-\frac{1}{e}\right)\left(B(S^*) - \frac{\epsilon}{2}\texttt{OPT}_k\right) - \frac{\epsilon}{2}\texttt{OPT}_k \geq \left(1-\frac{1}{e}-\epsilon\right)\texttt{OPT}_k,$$

where we used the fact $B(S^*) = \texttt{OPT}_k$, and the proof is complete. □

The total running time of HEDGE depends on the running time of the hyper-edge sampler and the greedy procedure. Specifically, the total running time is $O(t_{he} \cdot |\mathcal{H}| + (n\log(n) + |\mathcal{H}|))$, where t_{he} is the *expected* required amount of time for the sampler to output a single hyper-edge. The first term corresponds to the total required time for sampling, and the second term corresponds to an almost linear time implementation of greedy procedure as in [8].

Remark 1. *Note that if $\texttt{OPT}_k = \Theta(n^2)$, the sample complexity in Theorem 1 becomes $O\left(\frac{k\log(n)}{\epsilon^2}\right)$. We provide the first theoretical study on this assumption in Sect. 4.*

Finally, we provide a lower bound on the sample complexity of HEDGE, in order to output a $(1-1/e-\epsilon)$-approximate solution. This lower bound is still valid even if $\texttt{OPT}_k = \Theta(n^2)$.

Theorem 2. *In order to output a set S of size k such that $B(S) \geq (1-1/e-\epsilon)\texttt{OPT}_k$ w.h.p., the sample size in both HEDGE and [40]'s algorithm needs to be at least $O\left(\frac{n}{\epsilon^2}\right)$.*

Proof. Define a graph $A = (V_A, E_A)$ where

$$V_A = \left\{(i,j) \mid 1 \leq i \leq \epsilon\sqrt{n} \text{ and } 1 \leq j \leq \sqrt{n}\right\},$$

and two nodes (i,j) and (i',j') are connected if $i = i'$ or $j = j'$[1]. Note that the distance between every pair of nodes is at most 2, and there are at most 2 shortest paths between a pair of nodes in A. Let G be a graph of size n which has $(1-\epsilon)n$ isolated nodes and A as its largest connected components. We have the following lemma:

Lemma 3. *If $k = \epsilon n$, then $\texttt{OPT}_k = \sqrt{n}(\sqrt{n}-1)\cdot\epsilon\sqrt{n}(\epsilon\sqrt{n}-1) = \Theta(\epsilon^2 n^2) = \Theta(n^2)$, since ϵ is a constant.*

Proof of the lemma. Obviously, all the isolated nodes in G have zero BWC. So, the optimal set, S^*, is A (which is already of size $k = \epsilon n$). Now, if for two nodes s, t in G, there is a shortest path with an internal node in A, we have $s, t \in V_A$ such that $s = (i,j)$ and $t = (i',j')$ where $i \neq i'$ and $j \neq j'$. In this case, there are exactly two s-t shortest paths with exactly 1 internal nodes. Finally, the number of such pairs is exactly $\sqrt{n}(\sqrt{n}-1)\cdot\epsilon\sqrt{n}(\epsilon\sqrt{n}-1)$. □

Now note that in both HEDGE and the algorithm of [40], we first choose a pair of nodes in s, t in G, and if s and t are not in the same connected component, the returned hyper-edge is an empty set. Therefore, in order to have a non-empty hyper-edge both nodes should be chosen from V_A, which occurs with probability ϵ^2. Thus, sampling $o(n/\epsilon^2)$ hyper-edge results in reaching to at most $o(n) = o(|A|) = o(k)$ nodes, and so, the algorithm will not be able to tell the difference between the isolated nodes and many (arbitrarily large) number of nodes in A as they were not detected by any hyper-edge. □

[1]Without loss of generality we can assume $\epsilon\sqrt{n}$ and $\sqrt{n}$ are integers, as the arguments still hold after rounding them to the closest integers.

Remark 2. *Theorem 2 implies that the number of samples $M = O(\log(n)/\epsilon^2)$ as claimed in [40] is not sufficient.*

3.2 Beyond betweenness centrality

Suppose $C : 2^V \rightarrow \mathbb{R}^{\geq 0}$ is a centrality measure that is also defined on subset of nodes. Clearly, if C is monotone-submodular and admits a hyper-edge sampler, the algorithm HEDGE can be applied to and all the results in this section hold for C. Here, we give a couple of examples of such centrality measures, and it is easy to verify their monotonicity and submodularity.

Coverage centrality. The coverage centrality [40] for a set $S \subseteq V$ is defined as $C(S) = \sum_{(s,t)\in V^2} P_{s,t}(S)$, where $P_{s,t}(S)$ is 1 if S has an internal node on any s-t shortest path, and 0 otherwise. The coverage centrality admits a hyper-edge sampler $\mathcal{A}$ as follows: uniformly at random pick two nodes s and t. By running a breadth-first-search from s, we output every node that is on at least one shortest path from s to t. Note that for every subset of nodes $\Pr_{h\sim\mathcal{A}}(h \cap S \neq \emptyset) = \frac{1}{n(n-1)}C(S)$.

κ-Path centrality Alahakoon et al. introduced the κ-path centrality of a node [2].Their notion naturally generalizes to any subset of nodes as $C(S) = \sum_{s\in V} P^s_\kappa(S)$, where $P^s_\kappa(S)$ is the probability that a *random simple path* of length κ starting at s will pass a node in S: a random simple path starting at node s is generated by running a random walk that always chooses an unvisited neighbor uniformly at random, and stops after κ of edges being traversed or if there is no unvisited neighbor. Note that κ-path centrality is a generalization of *degree centrality* by letting $\kappa = 1$ and considering sets of single nodes.

Obviously, κ-path centrality admits a hyper-edge sampler based on its definition: Let $\mathcal{A}$ be an algorithm that picks a node uniformly at random, and generates a random simple path of length at most κ, and outputs the generated simple path as a hyper-edge. Therefore, for any subset S we have $\Pr_{h\sim\mathcal{A}}(h \cap S \neq \emptyset) = \frac{1}{n}\sum_{s\in V} P^s_\kappa(S) = \frac{1}{n}C(S)$.

4. ON $\texttt{OPT}_K = \Theta(N^2)$ EQUALITY

Recall that all additive approximation guarantees for BWC as well as all existing approximation guarantees for A-BWC involve an error term which grows as $\Theta(n^2)$. In this Section we provide strong theoretical evidence in favor of the following question: "Why does prior work which relies heavily on the strong assumption that $\texttt{OPT}_k = \Theta(n^2)$ perform well on real-world networks?" We quote Yoshida [40]: *This additive error should not be critical in most applications, as numerous real-world graphs have vertices of centrality $\Theta(n^2)$.*

We show that the $\texttt{OPT}_k = \Theta(n^2)$ assumption holds for two important classes of graphs: graphs of bounded treewidth networks, and for certain stochastic graph models that generate scale-free and small-world networks, known as *random Apollonian networks* [17].

4.1 Bounded treewidth graphs

We start by defining the treewidth of a graph:

Definition 3 (Treewidth). *For an undirected graph $G = (V, E)$, a* ***tree decomposition*** *is a tree T with nodes $V_1, \dots, V_r$ where each V_i is (assigned to) a subset of V such that (i) for every $u \in V$ there exists at least an i where $u \in V_i$, (ii) if V_i and V_j both contains a node u, then u belongs to every V_k on the unique shortest path from V_i to V_j in T, and (iii) for every edge $(u, v) \in E$ there exists a V_i such that $u, v \in V_i$. The* ***width*** *of the tree decomposition T is defined as $\max_{1 \le i \le r} |V_i| - 1$, and the* ***treewidth*** *of the graph G is the minimum possible width of any tree decomposition of G.*

Now, we have the following theorem.

Theorem 3. *Let $G = (V, E)$ be an undirected, connected graph of bounded treewidth. Then $\texttt{OPT}_k = \Theta(n^2)$.*

Proof. Suppose w is the treewidth of G, which is a constant (bounded). It is known that any graph of treewidth w has a balanced vertex separator[2] $S \subseteq V$ of size at most $w + 1$ [35]. This implies that $O(n^2)$ shortest paths pass through S. Since $|S| = w + 1 = \Theta(1)$, there exists at least one vertex $u \in S$ such that $B(u) = \Theta(n^2)$. Hence, $\texttt{OPT}_1 = \Theta(n^2)$, and since $\texttt{OPT}_1 \le \texttt{OPT}_k$ we have $\texttt{OPT}_k = \Theta(n^2)$. $\square$

It is worth emphasizing that the classical Barabási-Albert random tree model [5, 30] belongs to this category. For a recent study of the treewidth parameter on real-world networks, see [1].

4.2 Scale-free, small-world networks

We show that $\texttt{OPT}_k = \Theta(n^2)$ for random Apollonian networks. Our proof for the latter model relies on a technique developed by Frieze and Tsourakakis [17] and carries over for random unordered increasing k-trees [18]. A *random Apollonian network* (RAN) is a network that is generated iteratively. The RAN generator takes as input the desired number of nodes $n \ge 3$ and runs as follows:

- Let G_3 be the triangle graph, whose nodes are $\{1, 2, 3\}$, and drawn in the plane.
- **for** $t \leftarrow 4$ to n:
  - Sample a face $F_t = (i, j, k)$ of the planar graph G_{t-1} uniformly at random, except for the outer face.
  - Insert the new node t inside this face connecting it to i, j, k.

Figure 1(a) shows an instance of a RAN for $n = 100$. The triangle is originally embedded on the plane as an equilateral triangle. Also, when a new node t chooses its face (i, j, k) it is embedded in the barycenter of the corresponding triangle and connects to i, j, k via the straight lines: (i, t), (j, t), and (k, t). It has been shown that the diameter of a RAN is $O(\log(n))$ with high probability [17, 13].

At any step, we refer to set of candidate faces as the set of active faces. Note that there is a bijection between the active faces of a RAN and the leaves of random ternary trees as illustrated in Figure 1(b), and noticed first by [17].

We shall make use of the following formulae for the number of nodes (v_t), edges (e_t) and faces (f_t; excluding the outer face) after t steps in a RAN G_t:

$$v_t = t, \quad e_t = 3t - 6, \quad f_t = 2t - 5.$$

[2]Means a set of nodes Γ such that $V \setminus \Gamma = A \cup B$, where A and B are disjoint and both have size $\Theta(n)$.

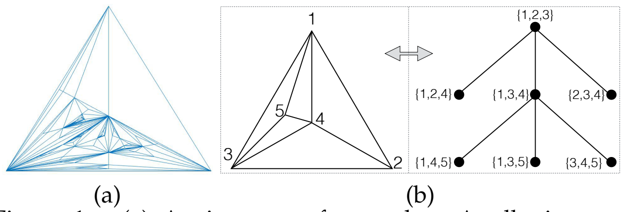


Figure 1: (a) An instance of a random Apollonian network for $n = 100$. (b) Bijection between RANs and random ternary trees.

Note that $f_t = \Theta(v_T) = \Theta(t)$.

Theorem 4. *Let G be a RAN of size n. Then $\texttt{OPT}_k = \Theta(n^2)$.*

Proof. Note that removing a node from the random ternary tree $T = (V_T, E_T)$ (as in Fig. 1(b)) corresponds to removing three nodes from G, corresponding to a face F that existed during the RAN generation process. Clearly, the set of these three nodes is a vertex separator that separates the nodes inside and outside of F. Therefore, all nodes in the tree except for the root r correspond to a vertex separator in G. Observe that the leaves in T correspond to the set of active faces, and thus, T has $f_n = 2n - 5$ leaves after n steps.

We claim that there exists an edge $(F, F') \in E_T$ (recall that the nodes of T are active faces during the generating process of G) such that the removal of e from E_T results in two subtrees with $\Theta(n)$ leaves. We define $g : V_T \to \mathbb{Z}$ to be the function that returns for a node $v \in V_T$ the number of leaves of the subtree rooted at v. Hence, $g(r) = 2n - 5, g(u) = 1$ for any leaf u. To find such an edge consider the following algorithm. We start from the root r of T descending to the bottom according to the following rule which creates a sequence of nodes $u_0 = r, u_1, u_2, \dots$: we set u_{i+1} to be the child with most leaves among the tree subtrees rooted at the three children of u_i. We stop when we first find a node u_i such that $g(u_i) \ge cn$ and $g(u_{i+1}) < cn$ for some constant c. Clearly, $g(u_{i+1}) \ge cn/3 = \Theta(n)$, by pigeonhole principle. So, let $F = u_i$ and $F' = u_{i+1}$.

Now suppose $F' = \{x, y, z\}$, and consider removing x, y, and z from G. Clearly, $F' \neq r$ as F' is a child of F. Also, due to the construction of a RAN, after removing x, y, z, there are exactly two connected components, G_1 and G_2. Also, since $cn/3 \le g(F') < cn$, the number of nodes in each of G_1 and G_2 is $\Theta(n)$.

Finally, observe that at least one of the three nodes x, y, z must have betweenness centrality score $\Theta(n^2)$, as the size of the separator is 3 and there exist $\Theta(n^2)$ paths that pass through it (connecting the nodes in G_1 and in G_2). Therefore, $\texttt{OPT}_1 \ge \max\{B(x), B(y), B(z)\} = \Theta(n^2)$, and since $\texttt{OPT}_1 \le \texttt{OPT}_k$ we have $\texttt{OPT}_k = \Theta(n^2)$. $\square$

We believe that this type of coupling can be used to prove similar results for other stochastic graph models.

5. EXPERIMENTAL RESULTS

In this section we present our experimental results. We first start by comparing `HEDGE` (our sampling based algorithm) with `EXHAUST` (the exhaustive algorithm defined in Sect. 2) and show that the centrality of `HEDGE`'s output is close to the centrality of `EXHAUST`'s output, with great speedup. This part is done for 3 small graphs as `EXHAUST` cannot scale to larger graphs.

We then compare our sampling method with the method presented in [40]. We show that, although our method stores less per each hyper-edge, it does not loose its accuracy.

Equipped with our scalable algorithm, HEDGE, we will be able to focus on some of the interesting characteristics of the central nodes: (i) How does their centrality change over time in evolving graphs? (ii) How influential are they? and (iii) How does the size of the largest connected component change after removing them?

In our experiments, we assume the graphs are simple (no self-loop or parallel edge) but the edges can be directed. We used publicly available datasets in our experiments[3]. HEDGE is implemented in C++.

5.1 Accuracy and time efficiency

Table 1 shows the results of EXHAUST and HEDGE on three graphs for which we were able to run EXHAUST. The fact that EXHAUST is able to run only on networks of this scale indicates already the value of HEDGE's scalability. As we can see, HEDGE results in significant speedups and negligible loss of accuracy.

In Table 1 the centrality of the output sets and the speed up gained by HEDGE is given, and as shown, HEDGE gives a great speedup with almost the same quality (i.e., the centrality of the output) of EXHAUST. The centrality of the outputs are *scaled* by $\frac{1}{n(n-1)}$, where n is the number of nodes in each graph. Motivated by the result in Sect. 4 we run HEDGE using $k\log(n)/\epsilon^2$ hyper-edges for $\epsilon = 0.1$, and for each case, ten times (averages are reported). For sake of comparison, these experiments were executed using a single machine with Intel Xeon CPU at 2.83GHz and with 36GB RAM.

				Algorithms		
GRAPHS	#nodes	#edges	k	EXHAUST	HEDGE	speedup
ca-GrQd	5242	14496	10	0.242	0.241	2.616
			50	0.713	0.699	2.516
			100	0.974	0.951	2.217
p2p-Gnutella08	6301	20777	10	0.013	0.011	6.773
			50	0.036	0.035	6.478
			100	0.053	0.051	6.117
ca-HepTh	9877	25998	10	0.165	0.164	4.96
			50	0.498	0.497	4.729
			100	0.747	0.745	4.473

Table 1: HEDGE vs. EXHAUST: centralities and speedups.

5.2 Comparison against [40]

We compare our method against Yoshida's algorithm (Y-ALG) [40] on four undirected graphs (as Y-ALG runs on undirected graphs). We use Yoshida's implementation which he kindly provided to us. Note that Yoshida's algorithm applies a different sampling method than ours: it is based on sampling random *s*-*t* pairs of nodes and assigning weights to *every* node that is on any *s*-*t* shortest path, whereas in our method we only pick one randomly chosen *s*-*t* shortest path with no weight on the nodes.

Y-ALG and HEDGE use $\frac{2\log(2n^3)}{\epsilon^2}$ and $\frac{k\log(n)}{\epsilon^2}$ samples, respectively, where n is the number of nodes in the graph, and we set $\epsilon = 0.1$. We also run a variation of our algorithm, HEDGE$_=$, which is essentially HEDGE but with $\frac{2\log(2n^3)}{\epsilon^2}$ samples. This allows a more fair comparison between the methods.

[3]http://snap.stanford.edu and http://konect.uni-koblenz.de/networks/dblp_coauthor

Table 2 shows the estimated centrality of the output sets, and the number of samples each algorithm uses. Surprisingly, Y-ALG does not outperform HEDGE$_=$, despite the fact that it maintains extra information. Finally, our proposed algorithm HEDGE is consistently better than the other two algorithms.

		Betw. Centrality			# of Samples		
GRAPHS	k	*Y-ALG*	*HEDGE$_=$*	*HEDGE*	*Y-ALG*	*HEDGE$_=$*	*HEDGE*
CA-GrQc	*10*	0.208	0.214	0.215			8565
	50	0.484	0.483	0.49	5278		42822
	100	0.569	0.568	0.577			85643
CA-HepTh	*10*	0.151	0.151	0.154			9198
	50	0.403	0.4	0.409	5658		45989
	100	0.534	0.533	0.547			91978
ego-Facebook	*10*	0.924	0.932	0.933			8304
	50	0.959	0.957	0.959	5121		41519
	100	0.962	0.96	0.964			83038
email-Enron	*10*	0.329	0.335	0.335			10511
	50	0.644	0.646	0.65	6445		52552
	100	0.754	0.756	0.762			105104

Table 2: Comparison against Y-ALG

5.3 Applications

For the next three experiments, we consider 3 more larger graphs that HEDGE can handle due to its scalability: email-Enron, loc-Brightkite, and soc-Epinion1, with 36692-183831, 58228-214078, and 75879-508837 number of *nodes-edges*, respectively. These experiments are based on orders defined over the set of nodes as follows: we generate $100\log(n)/\epsilon^2$ hyper-edges, where n is the number of nodes, and $\epsilon = 0.25$. Then we order the nodes based on the order HEDGE picks the nodes.

For sake of comparison, we ran HEDGE using the coverage and κ-path (for $\kappa = 2$) centralities, since both of them admit hyper-edge sampler as we showed in Sect. 3.2. Also, we considered a fourth centrality that we call triangle centrality, where the centrality of a set of nodes S equals to the number of triangles that intersect with S. For the triangle centrality, we run EXHAUST as computing this centrality is easy and scalable to large graphs[4]. All these experiments are run ten times, and we report the average values.

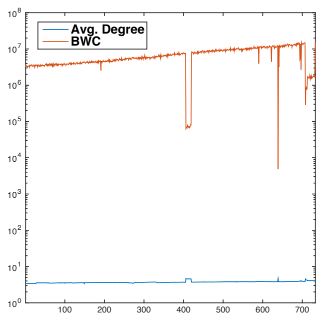
Time evolving networks. Leskovec *et al.* studied empirically properties of time-evolving real-world networks [29]. In this section we investigate how BWC of the most central nodes changes as a function of time.

We study two temporal datasets, the DBLP [5] and Autonomous Systems (AS) datasets. We also generate stochastic Kronecker graphs on 2^i vertices for $i \in \{8, \ldots, 20\}$, using $\left(\begin{smallmatrix} 0.9 & 0.5 \\ 0.5 & 0.2 \end{smallmatrix}\right)$ as the core-periphery seed matrix. We assume that the i-th time snapshot for Kronecker graphs corresponds to 2^i vertices, for $i = 8, \ldots, 20$. Note that in these evolving sets, the number of nodes also increases along with new edges. Also, note that the main difference between DBLP and Autonomous Systems is that for DBLP edges and nodes only can be added, where in Autonomous Systems nodes and edges can be increased and decreased.

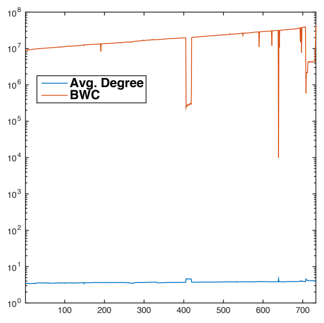
[4]EXHAUST for the triangle centrality, at every iteration simply chooses a node that is incident with more number of new triangles.

[5]Timestamps are in Unix time and can be negative.

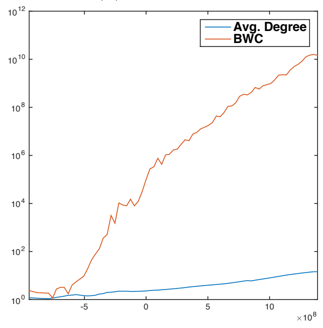
The results are plotted in logarithmic scale (Fig. 2), and as shown, we observe that the centrality of the highly central set of nodes increases. Also, we observe that the model of stochastic Kronecker graphs behaves similar to the real-world evolving networks with respect to these parameters.



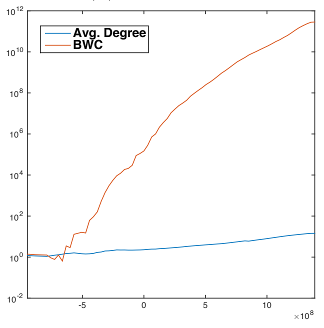
(a) AS: $k = 1$



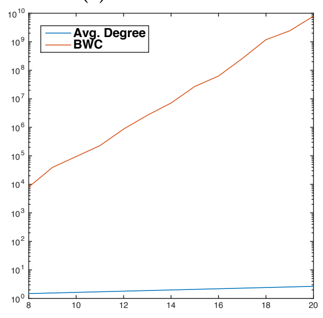
(b) AS: $k = 50$



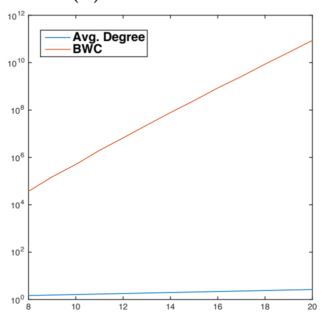
(c) DBLP: $k = 1$



(d) DBLP: $k = 50$



(e) KG: $k = 1$



(f) KG: $k = 50$

Figure 2: Largest betweenness centrality score and number of nodes, edges and average degree versus time on the (i) Autonomous systems (a),(b) (ii) DBLP dataset (c),(d) and (iii) stochastic Kronecker graphs (e),(f).

Influence maximization. We consider the Independent Cascade model [25], where each edge has the probability 0.01 of being active. For computing and maximizing the influence , we consider the algorithm of [8] using 10^6 number of samples (called hyper-edge but defined differently). We compute the influence of output of HEDGE with output of [8]. As shown in Table 3, and as we observe, the central nodes also have high influence, which shows a great correlation between being highly central and highly influential. It is worth outlining that our main point is to show that our proposed algorithm can be used to scale heuristic uses of BWC.

		METHODS				
GRAPHS	*k*	*IM*	*betw.*	*cov.*	*κ-path*	*tri.*
CA-GrQc	*10*	19.12	13.67	14.93	14.10	18.48
	50	76.65	67.28	67.44	65.06	69.30
	100	141.33	126.76	126.66	124.51	124.06
CA-HepTh	*10*	17.33	15.61	15.58	14.63	12.98
	50	77.88	70.53	69.95	67.80	63.95
	100	147.75	133.45	133.24	130.41	127.52
p2p-Gnutella08	*10*	19.61	13.05	13.71	10.39	18.06
	50	83.64	60.58	61.73	51.57	74.19
	100	148.86	118.27	118.76	103.58	132.04
email-Enron	*10*	461.84	458.70	450.34	455.25	451.53
	50	719.86	703.08	695.81	699.74	681.05
	100	887.63	863.66	858.39	865.76	830.15
loc-Brightkite	*10*	184.40	162.64	160.35	163.16	145.19
	50	402.85	372.64	360.64	366.28	330.45
	100	563.13	521.18	508.59	512.77	445.11
soc-Epinion1	*10*	343.89	81.57	111.47	14.43	311.74
	50	846.18	300.88	282.88	72.90	778.56
	100	1161.45	463.04	457.29	133.20	1062.99

Table 3: Comparing the *influence* of influential nodes (IM) and central nodes obtained by different centrality methods.

Graph attacks. It is a well-known fact that scale-free networks are robust to random failures but vulnerable to targeted attacks [3]. Some of the most efficient strategies for attacking graph connectivity is based on removing iteratively the most central vertex in the graph [21]. Such sequential attacks are central in studying the robustness of the Internet and biological networks such as protein-protein interaction networks [23].

We remove the nodes one-by-one (according to the order induced by these centralities and picked by HEDGE) and measure the size of the largest connected component. The results are plotted in Fig. 3. Our observation is that all the sizes of largest connected components decline significantly (almost linearly in size of S), which is compatible with our intuition of centralities. We also find that the κ-path and triangle centralities can be more effective at destroying the connectivity.

6. CONCLUSION

In this work, we provide HEDGE, a scalable algorithm for the (betweenness) Centrality Maximization problem, with theoretical guarantees. We also provide a general analytical framework for our analysis which can be applied to any monotone-submodular centrality measure that admits a hyper-edge sampler. We perform an experimental analysis of our method on real-world networks which shows that our algorithm scales gracefully as the size of the graph grows while providing accurate estimations. Finally, we study some interesting properties of the most central nodes.

A question worth investigating is whether removing nodes in the reserve order, namely by starting from the least central ones, produces a structure-revealing permutation, as it happens with peeling in the context of dense subgraph discovery [11, 38].

7. ACKNOWLEDGEMENTS

This work was supported in part by NSF grant IIS-1247581 and NIH grant R01-CA180776.

8. REFERENCES

[1] A. B. Adcock, B. D. Sullivan, and M. W. Mahoney. Tree decompositions and social graphs. *arXiv preprint arXiv:1411.1546*, 2014.

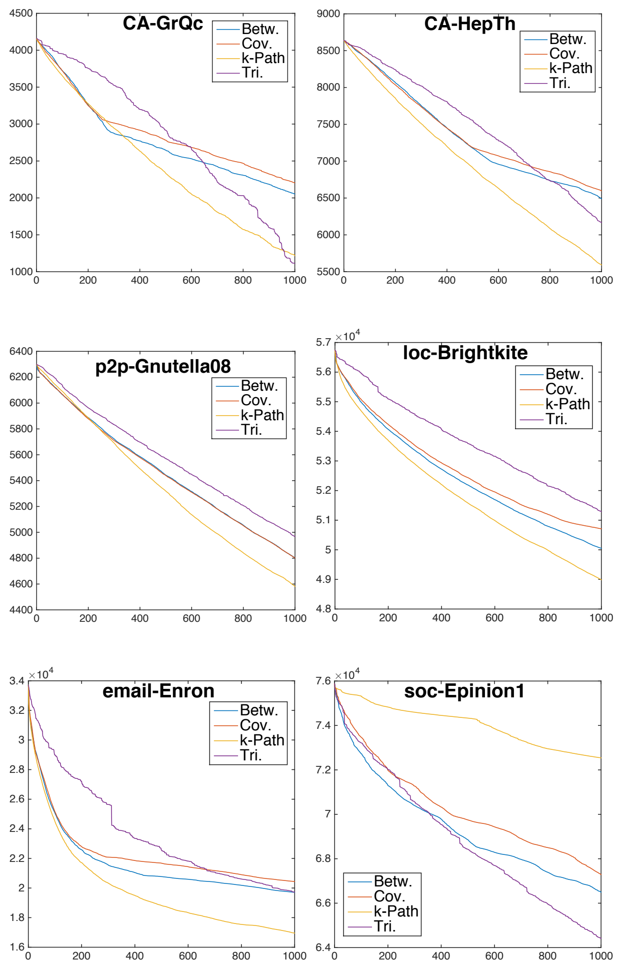


Figure 3: The size of the largest connected component, as we remove the first 1000 nodes in the order induced by centralities.

[2] T. Alahakoon et al. K-path centrality: A new centrality measure in social networks. In *Proceedings of the 4th Workshop on Social Network Systems*, page 1. ACM, 2011.

[3] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[4] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In *Algorithms and Models for the Web-Graph*, pages 124–137. Springer, 2007.

[5] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[6] A. Bavelas. A mathematical model for group structures. *Human Organization*, 7:16–30, 1948.

[7] P. Boldi and S. Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014.

[8] C. Borgs et al. Maximizing social influence in nearly optimal time. SODA, 2014.

[9] C. Brandes and C. Pich. Centrality estimation in large networks. *Int. J. Bifurcation and Chaos*, 17(7):2303–2318, 2007.

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

[11] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, 2000.

[12] S. Dolev, Y. Elovici, R. Puzis, and P. Zilberman. Incremental deployment of network monitors based on group betweenness centrality. *Information Processing Letters*, 109(20):1172–1176, 2009.

[13] E. Ebrahimzadeh, L. Farczadi, P. Gao, A. Mehrabian, C. M. Sato, N. Wormald, and J. Zung. On the longest paths and the diameter in random apollonian networks. *Electronic Notes in Discrete Mathematics*, 43:355–365, 2013.

[14] D. Erdos, V. Ishakian, A. Bestavros, and E. Terzi. A divide-and-conquer algorithm for betweenness centrality. SIAM, 2015.

[15] M. Fink and J. Spoerhase. Maximum betweenness centrality: approximability and tractable cases. In *WALCOM: Algorithms and Computation*, pages 9–20. Springer, 2011.

[16] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[17] A. Frieze and C. E. Tsourakakis. Some properties of random apollonian networks. *Internet Mathematics*, 10(1-2):162–187, 2014.

[18] Y. Gao. The degree distribution of random k-trees. *Theoretical Computer Science*, 410(8):688–695, 2009.

[19] R. Ghosh, S. Teng, K. Lerman, and X. Yan. The interplay between dynamics and networks: centrality, communities, and cheeger inequality. KDD, 2014.

[20] M. Girvan and M. E. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.

[21] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han. Attack vulnerability of complex networks. *Physical Review E*, 65(5):056109, 2002.

[22] V. Ishakian, D. Erdös, E. Terzi, and A. Bestavros. A framework for the evaluation and management of network centrality. In *SDM*, pages 427–438. SIAM, 2012.

[23] S. Iyer, T. Killingback, B. Sundaram, and Z. Wang. Attack robustness and centrality of complex networks. *PloS one*, 8(4):e59613, 2013.

[24] L. Katz. A new status index derived from sociometric index. *Psychometrika*, pages 39–43, 1953.

[25] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. KDD, 2003.

[26] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[27] R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *TOIS*, 19(2):131–160, 2001.

[28] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *Knowledge Discovery in Databases: PKDD 2005*, pages 133–145. Springer, 2005.

[29] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph

evolution: Densification and shrinking diameters. *TKDD*, 1(1):2, 2007.

[30] T. F. Móri. The maximum degree of the barabási–albert random tree. *Combinatorics, Probability and Computing*, 14(03):339–348, 2005.

[31] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.

[32] M. E. Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.

[33] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[34] M. Riondato and E. M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. WSDM, 2014.

[35] N. Robertson and P. D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986.

[36] A. E. Sarıyüce, E. Saule, K. Kaya, and Ü. V. Çatalyürek. Shattering and compressing networks for centrality analysis. *arXiv preprint arXiv:1209.6007*, 2012.

[37] D. Shah and T. Zaman. Rumor centrality: a universal source detector. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 199–210. ACM, 2012.

[38] C. Tsourakakis. The k-clique densest subgraph problem. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1122–1132. International World Wide Web Conferences Steering Committee, 2015.

[39] D. B. West. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[40] Y. Yoshida. Almost linear-time algorithms for adaptive betweenness centrality using hypergraph sketches. KDD, 2014.

APPENDIX

A. ON MAXIMUM CENTRALITY

Here, we present two examples in which the assumption of $\mathtt{OPT}_k = \Theta(\alpha)$ does not hold. However, it still remains as an interesting open problem to see what properties of the graphs result in this assumption.

Complete Graph Obviously the centrality of each set of nodes, both for Betweenness and Coverage centralities, is zero. Thus $\mathtt{OPT}_k = o(n^2) \neq \Theta(n^2)$. Note that in Betweenness and Coverage centrality we had $\alpha = n(n-1) = \Theta(n^2)$.

Hypercube The hypercube Q_r (with $n = 2^r$ nodes) is a *vertex-transitive graph*, i.e., for each pair of nodes there is an automorphism that maps one onto the other [39]. So, the centrality of the nodes are the same.

First consider the **coverage centrality**, and lets count how many pairs of nodes have a shortest path that pass the node $(0, \ldots, 0)$. Note that $a, b \in \{0,1\}^r$ have a shortest path that passes the origin if and only if $\forall i \in 1, \ldots, r : a_i b_i = 0$. To count the number of such pairs, we have to first choose a subset $I \subseteq \{1, \ldots, r\}$ of the bits that are non-zero either in a or b, in $\binom{r}{|I|}$ ways, and partition the bits of I between a and b (in $2^{|I|}$ ways). Therefore, the number of $(a, b) \in V^2$ pairs that their shortest path passes the node $(0, \ldots, 0)$ is

$$\sum_{i=0}^{r} \binom{r}{i} 2^i = (1+2)^r = 3^{\log(n)} = n^{\log(3)} = o(n^2).$$

So, the maximum coverage centrality of a node is at most $n^{\log(3)}$ (since we counted the endpoints as well, but should not have). Now by submodularity of the coverage centrality we have $\mathtt{OPT}_k \leq k n^{\log(3)} = O(n^{1+\log(3)}) = o(n^2)$.

Finally, since the betweenness centrality is no more than the coverage centrality, we have the similar result for betweenness centrality as well.