Ranking of Closeness Centrality for Large-Scale Social Networks

Kazuya Okamoto[1], Wei Chen[2], and Xiang-Yang Li[3]

[1] Kyoto University
okia@kuis.kyoto-u.ac.jp
[2] Microsoft Research Asia
weic@microsoft.com
[3] Illinois Institute of Technology and Microsoft Research Asia
xli@cs.iit.edu

Abstract. Closeness centrality is an important concept in social network analysis. In a graph representing a social network, closeness centrality measures how close a vertex is to all other vertices in the graph. In this paper, we combine existing methods on calculating exact values and approximate values of closeness centrality and present new algorithms to rank the top-k vertices with the highest closeness centrality. We show that under certain conditions, our algorithm is more efficient than the algorithm that calculates the closeness-centralities of all vertices.

1 Introduction

Social networks have been the subject of study for many decades in social science research. In recent years, with the rapid growth of Internet and World Wide Web, many large-scale online-based social networks such as Facebook, Friendster appear, and many large-scale social network data, such as coauthorship networks, become easily available online for analysis [Ne04a, Ne04b, EL05, PP02]. A social network is typically represented as a graph, with individual persons represented as vertices, the relationships between pairs of individuals as edges, and the strengths of the relationships represented as the weights on edges (for the purpose of finding the shortest weighted distance, we can treat lower-weight edges as stronger relationships). Centrality is an important concept in studying social networks [Fr79, NP03]. Conceptually, centality measures how central an individual is positioned in a social network. Within graph theory and network analysis, various measures (see [KL05] for details) of the centrality of a vertex within a graph have been proposed to determine the relative importance of a vertex within the graph. Four measures of centrality that are widely used in network analysis are *degree centrality*, *betweenness centrality*[1], *closeness centrality*,

[1] For a graph $G = (V, E)$, the betweenness centrality $C_B(v)$ for a vertex v is $C_B(v) = \sum_{s,t:s\neq t\neq v} \frac{\sigma_v(s,t)}{\sigma(s,t)}$ where $\sigma(s,t)$ is the number of shortest paths from s to t, and $\sigma_v(s,t)$ is the number of shortest paths from s to t that pass through v.

F.P. Preparata, X. Wu, and J. Yin (Eds.): FAW 2008, LNCS 5059, pp. 186–195, 2008.

and *eigenvector centrality*[2]. In this paper, we focus on *shortest-path closeness centrality* (or closeness centrality for short) [Ba50, Be65]. The closeness centrality of a vertex in a graph is the inverse of the average shortest-path distance from the vertex to any other vertex in the graph. It can be viewed as the efficiency of each vertex (individual) in spreading information to all other vertices. The larger the closeness centrality of a vertex, the shorter the average distance from the vertex to any other vertex, and thus the better positioned the vertex is in spreading information to other vertices.

The closeness centrality of all vertices can be calculated by solving all-pairs shortest-paths problem, which can be solved by various algorithms taking $O(nm + n^2 \log n)$ time [Jo77, FT87], where n is the number of vertices and m is the number of edges of the graph. However, these algorithms are not efficient enough for large-scale social networks with millions or more vertices. In [EW04], Eppstein and Wang developed an approximation algorithm to calculate the closeness centrality in time $O(\frac{\log n}{\epsilon^2}(n \log n + m))$ within an additive error of $\epsilon\Delta$ for the inverse of the closeness centrality (with probability at least $1 - \frac{1}{n}$), where $\epsilon > 0$ and Δ is the diameter of the graph.

However, applications may be more interested in ranking vertices with high closeness centralities than the actual values of closeness centralities of all vertices. Suppose we want to use the approximation algorithm of [EW04] to rank the closeness centralities of all vertices. Since the average shortest-path distances are bounded above by Δ, the average difference in average distance (the inverse of closeness centrality) between the ith-ranked vertex and the $(i+1)$th-ranked vertex (for any $i = 1, \ldots, n-1$) is $O(\frac{\Delta}{n})$. To obtain a reasonable ranking result, we would like to control the additive error of each estimate of closeness centrality to within $O(\frac{\Delta}{n})$, which means we set ϵ to $\Theta(\frac{1}{n})$. Then the algorithm takes $O(n^2 \log n(n \log n + m))$ time, which is worse than the exact algorithm.

Therefore, we cannot use either purely the exact algorithm or purely the approximation algorithm to rank closeness centralities of vertices. In this paper, we show a method of ranking top k highest closeness centrality vertices, combining the approximation algorithm and the exact algorithm. We first provide a basic ranking algorithm TOPRANK(k), and show that under certain condi-

[2] Given a graph $G = (V, E)$ with adjacency matrix A, let x_i be the eigenvector centrality of the ith node v_i. Then vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$ is the solution of equation $A\mathbf{x} = \lambda\mathbf{x}$, where λ is the greatest eigenvalue of A to ensure that all values x_i are positive by the Perron-Frobenius theorem. Google's PageRank [BP98] is a variant of the eigenvector centrality measure. The PageRank vector $\mathbf{R} = (r_1, r_2, \cdots, r_n)^T$, where r_i is the PageRank of webpage i and n is the total number of webpages, is the solution of the equation

$$\mathbf{R} = \frac{1-d}{n} \cdot \mathbf{1} + d\mathbf{L}\mathbf{R}.$$

Here d is a damping factor set around 0.85, $\mathbf{L}$ is a modified webpage-adjacency matrix: $l_{i,j} = 0$ if page j does not link to i, and normalised such that, for each j, $\sum_{i=1}^{n} l_{i,j} = 1$, i.e., $l_{i,j} = \frac{a_{i,j}}{d_j}$ where $a_{i,j} = 1$ only if page j has link to page i, and $d_j = \sum_{i=1}^{n} a_{i,j}$ is the out-degree of page j.

tions, the algorithm ranks all top k highest closeness centrality vertices (with high probability) in $O((k+n^{\frac{2}{3}}\cdot\log^{\frac{1}{3}} n)(n\log n+m))$ time, which is better than $O(n(n\log n+m))$ (when $k=o(n)$), the time needed by a brute-force algorithm that simply computes all average shortest distances and then ranks them. We then use a heuristic to further improve the algorithm. Our work can be viewed as the first step toward designing and evaluating efficient algorithms in finding top ranking vertices with highest closeness centralities. We discuss in the end several open problems and future directions of this work.

2 Preliminary

We consider a connected weighted undirected graph $G=(V,E)$ with n vertices and m edges ($|V|=n, |E|=m$). We use $d(v,u)$ to denote the length of a shortest-path between v and u, and Δ to denote the diameter of graph G, i.e., $\Delta=\max_{v,u\in V} d(v,u)$. The *closeness centrality* c_v of vertex v [Be65] is defined as

$$c_v=\frac{n-1}{\Sigma_{u\in V} d(v,u)}. \tag{2.1}$$

In other words, the closeness centrality of v is the inverse of the average (shortest-path) distance from v to any other vertex in the graph. The higher the c_v, the shorter the average distance from v to other vertices, and v is more important by this measure. Other definitions of closeness centralities exist. For example, some define the closeness centrality of a vertex v as $\frac{1}{\Sigma_{u\in V} d(v,u)}$ [Sa66], and some define the closeness centrality as the mean geodesic distance (i.e the shortest path) between a vertex v and all other vertices reachable from it, i.e., $\frac{\Sigma_{u\in V} d(v,u)}{n-1}$, where $n\geq 2$ is the size of the network's connected component V reachable from v. In this paper, we will focus on closeness centrality defined in equation (2.1).

The problem to solve in this paper is to find the top k vertices with the highest closeness centralities and rank them, where k is a parameter of the algorithm. To solve this problem, we combine the exact algorithm [Jo77, FT87] and the approximation algorithm [EW04] for computing average shortest-path distances to rank vertices on the closeness centrality. The exact algorithm iterates Dijkstra's single-source shortest-paths (SSSP for short) algorithm n times for all n vertices to compute the average shortest-path distances. The original Dijkstra's SSSP algorithm [Di59] computes all shortest-path distances from one vertex, and it can be efficiently implemented in $O(n\log n+m)$ time [Jo77, FT87].

The approximation algorithm RAND given in [EW04] also uses Dijkstra's SSSP algorithm. RAND samples ℓ vertices uniformly at random and computes SSSP from each sample vertex. RAND estimates the closeness centrality of a vertex using the average of ℓ shortest-path distances from the vertex to the ℓ sample vertices instead of to all n vertices. The following bound on the accuracy of the approximation is given in [EW04], which utilizes the Hoeffding's theorem [Ho63]:

$$\Pr\{|\frac{1}{\hat{c}_v}-\frac{1}{c_v}|\geq\epsilon\Delta\}\leq\frac{2}{n^{2\ell\frac{\epsilon^2}{\log n}(\frac{n-1}{n})^2}}, \tag{2.2}$$

for any small positive value ϵ, where $\hat{c}_v$ is the estimated closeness centrality of vertex v. Let a_v be the average shortest-path distance of vetex v, i.e.,

$$a_v = \frac{\Sigma_{u \in V} d(v,u)}{n-1} = \frac{1}{c_v}.$$

Using the average distance, inequality (2.2) can be rewritten as

$$\Pr\{|\hat{a}_v - a_v| \geq \epsilon \Delta\} \leq \frac{2}{n^{2\ell \frac{\epsilon^2}{\log n}(\frac{n-1}{n})^2}}, \tag{2.3}$$

where $\hat{a}_v$ is the estimated average distance of vertex v to all other vertices. If the algorithm uses $\ell = \alpha \frac{\log n}{\epsilon^2}$ samples ($\alpha > 1$ is a constant number) which will cause the probability of $\epsilon\Delta$ error at each vertex to be bounded above by $\frac{1}{n^2}$, the probability of $\epsilon\Delta$ error anywhere in the graph is then bounded from above by $\frac{1}{n}$ ($\geq 1 - (1 - \frac{1}{n^2})^n$). It means that the approximation algorithm calculates the average lengths of shortest-paths of all vertices in $O(\frac{\log n}{\epsilon^2}(n \log n + m))$ time within an additive error of $\epsilon\Delta$ with probability at least $1 - \frac{1}{n}$, i.e., with high probability (w.h.p.).

3 Ranking Algorithms

Our top-k ranking algorithm is based on the approximation algorithm as well as the exact algorithm. The idea is to first use the approximation algorithm with ℓ samples to obtain estimated average distances of all vertices and find a candidate set E of top-k' vertices with estimated shortest distances. We need to guarantee that all final top-k vertices with the exact average shortest distances are included in set E with high probability. Thus, we need to carefully choose number $k' > k$ using the bound given in formula (2.3). Once we find set E, we can use the exact algorithm to compute the exact average distances for all vertices in E and rank them accordingly to find the final top-k vertices with the highest closeness centralities. The key of the algorithm is to find the right balance between sample size ℓ and the candidate set size k': If we use a too small sample size ℓ, the candidate set size k' could be too large, but if we try to make k' small, the sample size ℓ may be too large. Ideally, we want an optimal ℓ that minimizes $\ell + k'$, so that the total time of both the approximation algorithm and the computation of exact closeness centralities of vertices in the candidate set is minimized. In this section we will show the basic algorithm first, and then provide a further improvement of the algorithm with a heuristic.

3.1 Basic Ranking Algorithm

We name the vertices in V as $v_1, v_2, \ldots, v_n$ such that $a_{v_1} \leq a_{v_2} \leq \cdots \leq a_{v_n}$. Let $\hat{a}_v$ be the estimated average distance of vertex v using approximation algorithm based on sampling. Figure 1 shows our basic ranking algorithm TOPRANK(k), where k is the input parameter specifying the number of top ranking vertices

the algorithm should extract. The algorithm also has a configuration parameter ℓ, which is the number of samples used by the RAND algorithm in the first step. We will specify the value of ℓ in Lemma 2. Function $f(\ell)$ in step 4 is defined as follows: $f(\ell) = \alpha' \sqrt{\frac{\log n}{\ell}}$ (where $\alpha' > 1$ is a constant number), such that the probability of the estimation error for any vertex being at least $f(\ell) \cdot \Delta$ is bounded above by $\frac{1}{2n^2}$, based on inequality (2.3) (when setting $\epsilon = f(\ell)$).

Algorithm *TOPRANK(k)*

1 Use the approximation algorithm RAND with a set S of ℓ sampled vertices to obtain the estimated average distance $\hat{a}_v$ for each vertex v.
// Rename all vertices to $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_n$ such that $\hat{a}_{\hat{v}_1} \leq \hat{a}_{\hat{v}_2} \leq \cdots \leq \hat{a}_{\hat{v}_n}$.
2 Find $\hat{v}_k$.
3 Let $\hat{\Delta} = 2 \min_{u \in S} \max_{v \in V} d(u, v)$.
// $d(u, v)$ for all $u \in S, v \in V$ have been calculated at step 1 and $\hat{\Delta}$ is determined in $O(\ell n)$ time.
4 Compute candidate set E as the set of vertices whose estimated average distances are less than or equal to $\hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \hat{\Delta}$.
5 Calculate exact average shortest-path distances of all vertices in E.
6 Sort the exact average distances and find the top-k vertices as the output.

Fig. 1. Algorithm for ranking top-k vertices with the highest closeness centralities

Lemma 1. *Algorithm TOPRANK(k) given in Figure 1 ranks the top-k vertices with the highest closeness centralities correctly w.h.p., with any configuration parameter ℓ.*

Proof. We show that the set E computed at step 4 in algorithm TOPRANK(k) contains all top-k vertices with the exact shortest distances w.h.p.

Let $T = \{v_1, \ldots, v_k\}$ and $\hat{T} = \{\hat{v}_1, \ldots, \hat{v}_k\}$. Since for any vertex v, the probability of the estimate $\hat{a}_v$ exceeding the error range of $f(\ell) \cdot \Delta$ is bounded above by $\frac{1}{2n^2}$, i.e., $\Pr\left(\neg\{a_v - f(\ell) \cdot \Delta \leq \hat{a}_v \leq a_v + f(\ell) \cdot \Delta\}\right) \leq \frac{1}{2n^2}$, we have

$$\Pr\left(\neg\{\bigwedge_{v \in T} \hat{a}_v \leq a_v + f(\ell) \cdot \Delta \leq a_{v_k} + f(\ell) \cdot \Delta\}\right) \leq \frac{k}{2n^2}; \text{ and}$$

$$\Pr\left(\neg\{\bigwedge_{\hat{v} \in \hat{T}} a_{\hat{v}} \leq \hat{a}_{\hat{v}} + f(\ell) \cdot \Delta \leq \hat{a}_{\hat{v}_k} + f(\ell) \cdot \Delta\}\right) \leq \frac{k}{2n^2}.$$

The latter inequality means that, with error probability of at most $\frac{k}{2n^2}$, there are at least k vertices whose real average distances are less than or equal to $\hat{a}_{\hat{v}_k} + f(\ell) \cdot \Delta$, which means $a_{v_k} \leq \hat{a}_{\hat{v}_k} + f(\ell) \cdot \Delta$ with error probability bounded above by $\frac{k}{2n^2}$. Then $\hat{a}_v \leq a_{v_k} + f(\ell) \cdot \Delta \leq \hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \Delta$ for all $v \in T$ with error probability bounded above by $\frac{k}{n^2}$. Moreover, we have $\Delta \leq \hat{\Delta}$, because for any $u \in S$, we have

$$\begin{aligned} \Delta &= \max_{v, v' \in V} d(v, v') \leq \max_{v, v' \in V} (d(u, v) + d(u, v')) \\ &= \max_{v, v' \in V} d(u, v) + \max_{v, v' \in V} d(u, v') = 2 \max_{v \in V} d(u, v). \end{aligned}$$

and thus

$$\Delta \leq 2 \min_{u \in S} \max_{v \in V} d(u,v) = \hat{\Delta}.$$

Therefore, for all $v \in T$, $\hat{a}_v \leq \hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \hat{\Delta}$ with probability at least $(1 - \frac{1}{n})$ (because $k \leq n$). Hence, TOPRANK(k) includes all top-k vertices with exact average distances in E in step 4, and TOPRANK(k) finds these exact k vertices in steps 5 and 6, with high probability. This finishes the proof of the lemma. □

We now evaluate the complexity of algorithm TOPRANK(k). The major computation tasks are ℓ computations of SSSP in step 1 and $|E|$ computations of SSSP in step 5. We need to choose an appropriate ℓ to minimize the sum of these computations. The number of computations of SSSP in step 5 depends on the distribution of estimated average distances of all vertices. The following lemma provides an answer when this distribution is uniform.

Lemma 2. *If the distribution of estimated average distances is uniform with range* $\mathrm{c}\Delta$ *(*c *is a constant number), then TOPRANK(k) takes* $O((k + n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n)(n \log n + m))$ *time, when we choose* $\ell = \Theta(n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n)$.

Proof. TOPRANK(k) takes $O(\ell(n \log n + m))$ time at step 1 because SSSP algorithm takes $O(n \log n + m)$ time and TOPRANK(k) iterates SSSP algorithm ℓ times.

Since the distribution of estimated average distances is uniform with range $\mathrm{c}\Delta$, there are $n \cdot \frac{2f(\ell) \cdot \hat{\Delta}}{\mathrm{c}\Delta}$ vertices between $\hat{a}_{\hat{v}_k}$ and $\hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \hat{\Delta}$, and $n \cdot \frac{2f(\ell) \cdot \hat{\Delta}}{\mathrm{c}\Delta}$ is $O(nf(\ell))$ because $\hat{\Delta} = 2 \min_{u \in S} \max_{v \in V} d(u,v) \leq 2 \max_{u,v \in V} d(u,v) = 2\Delta$. So, the number of vertices in E is $k + O(nf(\ell))$ and TOPRANK(k) takes $O((k + O(nf(\ell)))(n \log n + m))$ time at step 5.

Therefore, we select an ℓ that could minimize the total running time at step 1 and 5. In other words, we choose an ℓ to minimize $\ell + nf(\ell)$, which implies $\ell = \Theta(n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n)$. Then TOPRANK($k$) takes $O(n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n(n \log n + m))$ time at step 1, and takes $O((k + n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n)(n \log n + m))$ at step 5. Obviously TOPRANK(k) takes $O(n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n(n \log n + m))$ time at the other steps. So, TOPRANK(k) takes $O((k + n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n)(n \log n + m))$ total running time. □

Combining Lemmata 1 and 2, we arrive at the following theorem.

Theorem 1. *If the distribution of estimated average distances is uniform with range* $\mathrm{c}\Delta$ *(*c *is a constant number), then algorithm TOPRANK(k) given in Figure 1 ranks the top-k vertices with the highest closeness centralities in* $O((k + n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n)(n \log n + m))$ *time w.h.p., when we choose* $\ell = \Theta(n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n)$.

Theorem 1 only addresses the case when the distribution of estimated average distances is uniform. In this case, the complexity of TOPRANK(k) is better than a brute-force algorithm that simply computes all average shortest distances and ranks them, which takes $O(n(n \log n + m))$ time (assuming $k = o(n)$). Even

though the theorem is only for the case of uniform distribution, it could be applied to more general situations, as explained now. Given an estimated average distance x, its density $d(x)$ is the number of vertices whose estimate average distance is around x. The uniform distribution means that the density $d(x)$ is the same anywhere in the range of x. For any other distribution, it has an average density of d, which is the average of $d(x)$ over all x's. Suppose that the distribution is such that when x is sufficiently small, $d(x) \leq d$ (this property requires further investigation but we believe it is reasonable for social networks). Let x_0 be the largest value such that for all $x \leq x_0$, $d(x) \leq d$. Then, in our algorithm, as long as $\hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \hat{\Delta} \leq x_0$, the number of vertices between $\hat{a}_{\hat{v}_k}$ and $\hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \hat{\Delta}$ is at most $n \cdot \frac{2f(\ell) \cdot \hat{\Delta}}{c\Delta}$, as given in the proof of Lemma 2. Thus, Lemma 2 uses a conservative upper bound for this number, and it will still hold for the distributions with the above property.

Even with the above generalization, however, the savings from $O(n(n \log n + m))$ to $O((k + n^{\frac{2}{3}} \cdot \log^{\frac{1}{3}} n)(n \log n + m))$ is still not very significant. Ideally, we would like a ranking algorithm that is $O(poly(k)(n \log n + m))$, where $poly(k)$ is a polynomial of k, which means the number of SSSP calculations is only related to k, not n. This is possible for small k when the distribution of estimated average distances is not uniform but other distributions like the normal distribution. In this case, the number of additional SSSP computations for vertices in the range from $\hat{a}_{\hat{v}_k}$ to $\hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \hat{\Delta}$ could be small and not related to n. We leave this as a future research work (see more discussion in Section 4).

3.2 Improving the Algorithm with a Heuristic

The algorithm in Figure 1 spends its majority of computation on the following two steps: (1) step 1 computing SSSP for ℓ samples, and (2) step 5 computing

---

Algorithm ***TOPRANK2(k)***

1 Use the approximation algorithm RAND with a set S of ℓ sampled vertices to obtain the estimated average distance $\hat{a}_v$ for each vertex v.
// Rename all vertices to $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_n$ such that $\hat{a}_{\hat{v}_1} \leq \hat{a}_{\hat{v}_2} \leq \cdots \leq \hat{a}_{\hat{v}_n}$.
2 Find $\hat{v}_k$.
3 Let $\hat{\Delta} = 2 \min_{u \in S} \max_{v \in V} d(u, v)$.
4 Compute candidate set E as the set of vertices whose estimated average distances are less than or equal to $\hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \hat{\Delta}$.
5 **repeat**
6 $p \leftarrow |E|$
7 Select additional q vertices S^+ as new samples uniformly at random.
8 Update estimated average distances of all vertices using new samples in S^+ (need to compute SSSP for all new sample vertices).
9 $S \leftarrow S \cup S^+; \ell \leftarrow \ell + q; \hat{\Delta} \leftarrow \min(\hat{\Delta}, 2 \min_{u \in S^+} \max_{v \in V} d(u, v))$
// Rename all vertices to $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_n$ such that $\hat{a}_{\hat{v}_1} \leq \hat{a}_{\hat{v}_2} \leq \cdots \leq \hat{a}_{\hat{v}_n}$.
10 Find $\hat{v}_k$.
11 Compute candidate set E as the set of vertices whose estimated average distances are less than or equal to $\hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \hat{\Delta}$.
12 $p' \leftarrow |E|$
13 **until** $p - p' \leq q$
14 Calculate exact average shortest-path distances of all vertices in E.
15 Sort the exact average distances and find the top-k vertices as the output.

Fig. 2. Improved algorithm for ranking vertices with top k closeness centralities

SSSP for all candidates in set E. The key in reducing the running time of the algorithm is to find the right sample size ℓ to minimize $\ell + |E|$, the total number of SSSP calculations. However, this number is difficult to obtain before running the algorithm, especially when the distribution of average distances is unknown. In this section, we improve the algorithm by a heuristic to achieve the above goal.

The idea of the heuristic is to incrementally add new samples to compute more accurate average distances of all vertices. In each iteration, q new sample vertices are added. After computing the new average distances with these q new vertices, we obtain a new candidate set E. If the size of the candidate set E decreases more than q, then we know that the savings by reducing the number of candidates outweigh the cost of adding more samples. In this case, we continue the next iteration of adding more samples. This procedure ends when the cost of adding more samples outweighs the savings obtained by the reduced number of candidates. Figure 2 provides this heuristic algorithm. Essentially, this is the dynamic way of finding the optimal ℓ to minimize $\ell + |E|$ (or to make $\Delta\ell = -\Delta|E|$, where $\Delta\ell$ is the small change in ℓ and $\Delta|E|$ is the corresponding change in $|E|$).

The initial value of the ℓ in step 1 can be obtained based on Theorem 1 if we know that the distribution of the estimated average distances is uniform. Otherwise, we can choose a basic value, for example k, since we need to compute at least k SSSP in step 14 in any case. The incremental unit q could be a small value, for example, $\log n$. However, we do not know yet if $|E|$ strictly decreases when the sample size ℓ for estimating average distances increases, and if the rate of decrease of $|E|$ slows down when adding more and more sample vertices. Therefore, it is not guaranteed that the heuristic algorithm will always stop at the optimal sample size ℓ. An open problem is to study the conditions under which the change of $|E|$ with respect to the change of sample size ℓ indeed has the above properties, and thus the heuristic algorithm indeed provides the most efficient solution.

4 Conclusion and Discussions

This paper can be viewed as the first step towards the design of more efficient algorithms in obtaining highest ranked closeness centrality vertices. By combining the approximation algorithm with the exact algorithm, we obtain an algorithm that has better complexity than the brute-force exact algorithm.

There are many directions to extend this study.

- First, as mentioned in the previous section, we are interested in more efficient algorithms such that the number of SSSP computations is only related to k, not to n. This may be possible for some classes of social networks with certain properties on their average distance distributions.
- Second, the condition under which the heuristic algorithm results in the least number of SSSP computation is an open problem and would be quite interesting to study.

- Third, we may be able to obtain faster algorithm if we can relax the problem requirement. Instead of finding all top-k vertices with high probability, we may allow the output to have an error bound t, which is the number of vertices that should be ranked within top-k vertices but are missed in the output. Generally, given an algorithm for top-k query of various centralities, we define the *hit-ratio*, denoted as $\eta(\mathcal{A})$, of an output $\mathcal{A} = \{v_1, v_2, \cdots, v_\kappa\}$ (not necessarily of size k) as $\frac{|\mathcal{A} \cap \mathcal{Q}_k|}{k}$, where $\mathcal{Q}_k$ is the actual set of top-k vertices. Let $r(v_i)$ denote the the actual rank of v_i. Here, we require that $r(v_i) < r(v_{i+1})$, for $i \in [1, \kappa - 1]$. Thus we have $r(v_i) \geq i$. We define the *accuracy* of $\mathcal{A}$, denoted as $\alpha(\mathcal{A})$, as $\frac{\kappa(\kappa+1)}{2\sum_{i=1}^{\kappa} r(v_i)}$ (other definitions of accuracy are possible). Clearly, $\eta(\mathcal{A}) \in [0, 1]$ and $\alpha(\mathcal{A}) \in [0, 1]$. The hit-ratio and accuracy of an algorithm are then its worst performance over all possible inputs. We then may only require that $\Pr(\eta(\mathcal{A}) \geq 1 - \varrho, \alpha(\mathcal{A}) \geq 1 - \epsilon) \geq 1 - \delta$, for sufficiently small ϱ, ϵ and δ. Such relaxations in the requirement may allow much more efficient algorithms, since we observe that the complexity of our algorithm is mainly because we need to include all vertices in the extra range from $\hat{a}_{\hat{v}_k}$ to $\hat{a}_{\hat{v}_k} + 2f(\ell) \cdot \hat{\Delta}$ in order to include all top-k vertices with high probability.
- Fourth, we would like to study the stability of our algorithms for top-k query using random sampling. Costenbader et al. [CV03] studied the stability of various centrality measures. Their study shows that, with a 50% sample of the original network nodes, average correlations for the closeness measure ranged from 0.54 to 0.71.
- Finally, we can look into other type of centralities and see how to rank them efficiently using the technique in this paper. For example, Brandes [Br00] presents an algorithm for betweenness centrality of weighted graph with time-complexity $n(m + n \log n)$. Brandes et al. [BP00] present the first approximation algorithm for betweenness centrality. Improvements over their method were presented recently in [GS08, BK07].

To conclude, we would like to provide a brief comparison of our algorithms with the well known PageRank algorithm [BP98]. PageRank is an algorithm used to rank the importance of the webpages, which are viewed as vertices connected by directed edges (hyperlinks). As explained in Footnote 2, PageRank is a variant of the eigenvector centrality. Thus, it is a different measure from closeness centrality. More importantly, by definition the PageRank of a vertex (a webpage) depends on the PageRanks of other vertices linking to it, so the PageRank calculation requires computing all PageRank values of all vertices, even if only the top-k PageRank vertices are desired. However, for closeness centrality measure, our algorithms do not need to compute closeness centralities for all vertices. Instead, we may start with rough estimates of closeness centralities of vertices, and through refining the estimates we reduce the candidate set containing the top-k vertices. This results in reduced time complexity in our computation.

References

[BK07] Bader, D.A., Kintali, S., Madduri, K., Mihail, M.: Approximating Betweenness Centrality. In: The 5th Workshop on Algorithms and Models for the Web-Graph, pp. 124–137 (2007)

[Ba50] Bavelas, A.: Communication patterns in task-oriented groups. The Journal of the Acoustical Society of America 22(6), 725–730 (1950)

[Be65] Beauchamp, M.A.: An improved index of centrality. Behavioral Science 10(2), 161–163 (1965)

[Br00] Brandes, U.: Faster Evaluation of Shortest-Path Based Centrality Indices. Konstanzer Schriften in Mathematik und Informatik 120 (2000)

[BP00] Brandes, U., Pich, C.: Centrality Estimation in Large Networks. Intl. Journal of Bifurcation and Chaos in Applied Sciences and Engineering 17(7), 2303–2318

[BP98] Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30, 107–117 (1998)

[CV03] Costenbader, E., Valente, T.W.: The stability of centrality measures when networks are sampled. Social Networks 25, 283–307 (2003)

[Di59] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)

[EL05] Elmacioglu, E., Lee, D.: On six degrees of separation in DBLP-DB and more. ACM SIGMOD Record 34(2), 33–40 (2005)

[EW04] Eppstein, D., Wang, J.: Fast Approximation of Centrality. Journal of Graph Algorithms and Applications 8, 39–45 (2004)

[FT87] Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM 34(3), 596–615 (1987)

[Fr79] Freeman, L.C.: Centrality in social networks conceptual clarification. Social Networks 1(3), 215–239 (1978/79)

[GS08] Geisberger, R., Sanders, P., Schultes, D.: Better Approximation of Betweenness Centrality. In: Proceedings of the 10th Workshop on Algorithm Engineering and Experiments (ALENEX 2008), pp. 90–100. SIAM, Philadelphia (2008)

[Ho63] Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the ACM 58(1), 13–30 (1963)

[Jo77] Johnson, D.B.: Efficient algorithms for shortest paths in sparse networks. Journal of the ACM 24(1), 1–13 (1977)

[KL05] Koschutzki, D., Lehmann, K.A., Peeters, L., Richter, S., Tenfelde-Podehl, D., Zlotowski, O.: Centrality indices. Network Analysis, 16–61 (2005)

[Ne04a] Newman, M.E.J.: Coauthorship networks and patterns of scientific collaboration. Proceedings of the National Academy of Sciences 101, 5200–5205 (2004)

[Ne04b] Newman, M.E.J.: Who is the best connected scientist? A study of scientific coauthorship networks. Complex Networks, 337–370 (2004)

[NP03] Newman, M.E.J., Park, J.: Why social networks are different from other types of networks. Physical Review E 68, 036122 (2003)

[PP02] Potterat, J.J., Phillips-Plummer, L., Muth, S.Q., Rothenberg, R.B., Woodhouse, D.E., Maldonado-Long, T.S., Zimmerman, H.P., Muth, J.B.: Risk network structure in the early epidemic phase of HIV transmission in Colorado Springs. Sexually Transmitted Infections 78, i159–i163 (2002)

[Sa66] Sabidussi, G.: The centrality index of a graph. Psychometrika 31(4), 581–603 (1966)