Random-Radius Ball Method for Estimating Closeness Centrality

Wataru Inariba
The University of Tokyo
and JST, ERATO,
Kawarabayashi Large Graph Project
oinari17@gmail.com

Takuya Akiba
Preferred Networks, Inc.
akiba@preferred.jp

Yuichi Yoshida
National Institute of Informatics
and Preferred Infrastructure, Inc.
yyoshida@nii.ac.jp

Abstract

In the analysis of real-world complex networks, identifying important vertices is one of the most fundamental operations. A variety of centrality measures have been proposed and extensively studied in various research areas. Many of distance-based centrality measures embrace some issues in treating disconnected networks, which are resolved by the recently emerged harmonic centrality. This paper focuses on a family of centrality measures including the harmonic centrality and its variants, and addresses their computational difficulty on very large graphs by presenting a new estimation algorithm named the random-radius ball (RRB) method. The RRB method is easy to implement, and a theoretical analysis, which includes the time complexity and error bounds, is also provided. The effectiveness of the RRB method over existing algorithms is demonstrated through experiments on real-world networks.

Introduction

A measure that indicates the relative importance of a vertex is called the *centrality*. Centrality has found many applications in broad research areas such as sociology, psychology, biology, and computer science (Borgatti and Everett 2006; Boldi and Vigna 2014).

Bavelas' closeness centrality (Bavelas 1950) is one the most classic centralities and is still actively studied (Chechik, Cohen, and Kaplan 2015; Bergamini et al. 2016). It formulates the centrality of a vertex as the inverse of the sum of the distances to the other vertices. More specifically, Bavelas' closeness centrality of a vertex $v \in V$ in a graph $G = (V, E)$ is defined by

$$C(v) = \frac{1}{\sum_{u \in V} d(u, v)},$$

where $d(u, v)$ denotes the distance from u to v. However, Bavelas' closeness centrality has several drawbacks. First, it makes sense only on strongly connected graphs; any vertex that is not reachable from some other vertex has the null centrality value. Second, even if the graph is strongly connected, it can easily be affected by distant vertices.

The recently emerged *harmonic centrality* (Opsahl, Agneessens, and Skvoretz 2010; Boldi and Vigna 2014) is a possible solution to these problems. The harmonic centrality has a similar definition, specifically,

$$C_{\mathrm{H}}(v) = \sum_{u \in V \setminus \{v\}} \frac{1}{d(u, v)}.$$

This slight modification produces meaningful values even on disconnected graphs. Moreover, a recent survey on centralities (Boldi and Vigna 2014) concluded that, although the appropriate centrality may differ by application, the harmonic centrality has the most desirable features among popular centrality notions. In this paper, we focus on a generalized notion of the harmonic centrality that we call the *(generalized) closeness centrality*. The closeness centrality with respect to a *distance-decay* (monotone decreasing) function $\alpha : \mathbb{N} \to \mathbb{R}_+$ is defined by

$$C_\alpha(v) = \sum_{u \in V \setminus \{v\}} \alpha(d(u, v)).$$

For example, $C_\alpha(v)$ with $\alpha(x) = 1/x$ and $\alpha(x) = 2^{-x}$ coincides with the harmonic centrality and exponentially attenuated centrality (Dangalchev 2006), respectively.

However, the exact computation of the closeness centrality is impractical on today's massive graphs. Computing the closeness centralities of all the vertices requires the all-pairs shortest paths (APSP) computation, which costs $O(nm)$ time on unweighted graphs, where n and m are the number of vertices and edges, respectively. Moreover, computing the closeness centrality of a single vertex takes $O(m)$ time, which is not acceptable on large graphs.

Therefore, faster approximation algorithms are needed. Regarding the closeness centrality, two ingenious algorithms have been proposed independently: HyperBall (Boldi, Rosa, and Vigna 2011; Boldi and Vigna 2013) and All-Distances Sketches (ADS) (Cohen 1997; 2015).

HyperBall is practically the fastest method for approximating the closeness centrality. However, this method has two drawbacks. First, HyperBall relies heavily on a complicated data structure called a HyperLogLog counter (Flajolet et al. 2007; Heule, Nunkesser, and Hall 2013) and is hard to implement. Second, although the experimental results on real-world networks in (Boldi and Vigna 2013) are promising, no theoretical error bounds are available.

ADS is a framework that can be used to estimate various statistics of graphs, including the closeness centrality.

Although ADS boasts its theoretical guarantee on the time complexity and error bounds, it has difficulties on massive graphs due to its space usage (Akiba and Yano 2016).

Our Contributions. In this paper, we present a new randomized approximation method for the closeness centrality called the *random-radius ball* (RRB) method. The RRB method builds upon very simple procedures but is efficient in practice and has theoretical guarantees. The features of the RRB method can be summarized as follows.

- **Easy to implement**: The RRB method is easy to implement. With a basic breadth-first search (BFS) library, it can be implemented within only 30 lines of code.
- **Theoretically guaranteed**: We provide a solid theoretical analysis. The estimation is unbiased (as opposed to HyperBall). Moreover, we can control the trade-off between the running time and the root-mean-square error (RMSE) of the estimation to the true centrality value.
- **Efficient in practice**: Our experimental results on real-world networks demonstrate that the RRB method outperforms HyperBall and ADS. Space usage (in addition to the graph itself) is only $O(n \log n)$. This feature is crucial when the input graph is massive.

Related Work

Since most centrality notions require a high polynomial time to compute (such as $O(nm)$), a lot of estimation methods have been developed, including methods for the betweenness centrality (Brandes and Pich 2007; Geisberger and Schultes 2008; Riondato and Kornaropoulos 2014; Riondato and Upfal 2016), Bavelas' closeness centrality (Eppstein and Wang 2004; Cohen et al. 2014; Chechik, Cohen, and Kaplan 2015), harmonic centrality (Boldi and Vigna 2013; Cohen 2015), and spanning tree centrality (Mavroforakis et al. 2015; Hayashi, Akiba, and Yoshida 2016).

It is shown that computing the APSP is equivalent to computing the vertex with the highest Bavelas' closeness centrality in terms of (polynomial) time complexity (Abboud, Grandoni, and Williams 2015).

Preliminaries

Table 1 summarizes the notations that are frequently used throughout our paper. A *distance-decay function* $\alpha : \mathbb{N} \to \mathbb{R}_+$ is a monotone decreasing function such that $\lim_{d\to\infty} \alpha(d) = 0$. We abbreviate $\alpha(d(u,v))$ by $\alpha(u,v)$. Let $C_\alpha = \sum_{v\in V} C_\alpha(v)/n$ be the average of the closeness centralities.

We say that a random variable X is an *unbiased estimator* of $a \in \mathbb{R}$ if $\mathbf{E}[X] = a$. The *root-mean-square error* (*RMSE*) of a random variable X with respect to $a \in \mathbb{R}$ is $\sqrt{\mathbf{E}(X-a)^2}$. When X is an unbiased estimator of a, the RMSE coincides with the standard deviation $\sigma(X)$ of X.

Assumptions on graphs. Throughout this paper, we assume that the input graph is directed and unweighted. This is because we mainly consider applications in social networks and Web graphs, which are usually directed and unweighted. Having said that, we note that our method can be easily adapted for undirected and/or weighted graphs.

Table 1: Frequently used notations throughout this paper.

Symbol	Description
$\mathbb{R}_+$	A set of non-negative real numbers.
$\mathbb{Z}_+$	A set of non-negative integers.
$\sigma(X)$	The standard deviation of X.
$G = (V, E)$	A graph with vertex set V and edge set E.
n	The number of vertices: $\lvert V\rvert$.
m	The number of edges: $\lvert E\rvert$.
$d(u, v)$	The shortest-path distance from u to v.
$\deg^+(u)$	The out-degree of v.
$\alpha(x)$	A distance-decay function.
$\alpha(u, v)$	An abbreviation for $\alpha(d(u, v))$.

Proposed Algorithm

In this section, we present the RRB method. Throughout this section, we fix a directed graph $G = (V, E)$ and a distance-decay function $\alpha : \mathbb{N} \to \mathbb{R}_+$.

Random-Radius Ball Method

We now describe the idea of the RRB method. Suppose the goal is to estimate the closeness centrality $C_\alpha(v)$ of a single vertex $v \in V$. First, we introduce a counter $c(v)$ that is initialized to zero. Then, for each vertex $u \in V \setminus \{v\}$, we increment $c(v)$ with probability $t\alpha(u,v)$, where t is a time-and-accuracy trade-off parameter. Here, we assume $t\alpha(1) \leq 1$. The case when $t\alpha(1) > 1$ is discussed later. Note that $c(v)/t$ is an unbiased estimator of $C_\alpha(v)$; hence, it is outputted as an estimate of $C_\alpha(v)$.

This idea can be extended to estimate the closeness centralities of all the vertices. For each vertex $v \in V$, the counter $c(v)$ is initialized to zero. Then, for each vertex $u \in V$, we assign a random variable r_u uniformly sampled from $[0, 1]$, which is called the *rank* of u. Then, we increment $c(v)$ for every vertex with $d(u,v) \leq \tau$, where $\tau = \lfloor \alpha^{-1}(r_u/t) \rfloor$. Here, $\alpha^{-1}(\cdot)$ exists because α is monotone decreasing. Note that $c(v)$ is incremented when $r_u \in [0, t\alpha(u,v)]$; hence, the probability that $c(v)$ is incremented is $t\alpha(u,v)$. Then, we output $\hat{C}_\alpha(v) := c(v)/t$ as an estimate of $C_\alpha(v)$ for each $v \in V$. See Algorithm 1 for details.

Before analyzing the RRB method, we define the values $\mu_t(v) \in \mathbb{R}_+$ for each $v \in V$ and $k_t \in \mathbb{R}_+$ by

$$\mu_t(v) := \sum_{u \in V\setminus\{v\}} t\alpha(u,v) = tC_\alpha(v),$$

$$k_t := \frac{1}{n}\sum_{v\in V} \mu_t(v) = tC_\alpha. \tag{1}$$

In other words, $\mu_t(v)$ is the expected value of $c(v)$, and k_t is the average of $\mu_t(v)$ over all vertices $v \in V$.

Although, as we will see, k_t is an important value that determines the running time and accuracy of the RRB method, we cannot directly specify k_t in Algorithm 1. We will address this issue later using a bootstrap method, which automatically determines an appropriate t from k_t.

Time and Space Complexity

We now analyze the time and space complexities of the RRB method.

Algorithm 1: Random-radius ball (RRB) method.

```
Input: A graph $G = (V, E)$, a distance-decay function
       $\alpha : \mathbb{N} \to \mathbb{R}_+$, and a parameter $t > 0$
Procedure RRB$(G, \alpha, t)$
1    foreach $v \in V$ do
2        $c(v) \leftarrow 0$.
3    foreach $u \in V$ do
4        $r_u \leftarrow$ a value uniformly sampled from $[0, 1]$.
5        $\tau \leftarrow \lfloor \alpha^{-1}(r_u / t) \rfloor$.
6        Perform a BFS of depth $\tau$ from $u$.
7        foreach visited vertex $v \in V$ with $v \neq u$ do
8            $c(v) \leftarrow c(v) + 1$.
9    foreach $v \in V$ do
10       $\hat{C}_\alpha(v) \leftarrow c(v)/t$.
11   return $\left\{\hat{C}_\alpha(v)\right\}_{v \in V}$.
```

Theorem 1: *Algorithm 1 runs in $O(tm \cdot \max_{v \in V} C_\alpha(v))$ expected time with $O(n \log n)$ space (in addition to the graph itself).*

Proof. The additional space complexity is $O(n \log n)$ because the RRB method can run with counters $\{c(v)\}_{v \in V}$, each of which requires only $O(\log n)$ bits. The time complexity is dominated by the BFS (Line 6). The expected number of scanned vertices is

$$\sum_{v \in V} \mu_t(v) \cdot \deg^+(v) \leq \max_{v \in V}(tC_\alpha(v)) \cdot \sum_{v \in V} \deg^+(v) = tm \cdot \max_{v \in V} C_\alpha(v). \qquad \square$$

A disadvantage of Theorem 1 is that the obtained time complexity relies on an unknown value, i.e., $\max_{v \in V} C_\alpha(v)$. In small-world networks such as social networks and Web graphs, the maximum centrality $\max_{v \in V} C_\alpha(v)$ and average C_α usually differ only by a constant factor. This case leads to the following.

Corollary 2: *If $\max_{v \in V} C_\alpha(v) \leq KC_\alpha$ holds for some $K \in \mathbb{R}_+$, then Algorithm 1 runs in $O(Kk_t m)$ time.*

Proof. By Theorem 1, the time complexity is $O(tm \cdot \max_{v \in V} C_\alpha(v)) = O(tmKC_\alpha) = O(Kk_t m)$. $\square$

Although k_t is again an unknown value, we will show that its value can be controlled using the bootstrap method.

Error Bounds

We now analyze the accuracy of Algorithm 1. We begin with the following fact.

Proposition 3: *For every $v \in V$, $\hat{C}_\alpha(v)$ is an unbiased estimator of $C_\alpha(v)$.*

Proof. The result is immediate from the fact that $\mathbf{E}[\hat{C}_\alpha(v)] = \mathbf{E}[c(v)/t] = \mu_t(v)/t = C_\alpha(v)$. $\square$

Now, we bound the standard deviation of $\hat{C}_\alpha(v)$ as it coincides with the RMSE of $\hat{C}_\alpha(v)$ with respect to $C_\alpha(v)$.

Theorem 4: *For every $v \in V$, we have*

$$\sigma(\hat{C}_\alpha(v)) \leq \sqrt{\frac{C_\alpha(v)}{t}}.$$

Proof. Recall that $c(v)$ is the sum of $n - 1$ independent Bernoulli random variables whose expectations are $t\alpha(u, v)$ $(u \in V \setminus \{v\})$. Then,

$$\sigma(\hat{C}_\alpha(v))^2 = \mathbf{Var}\left[\frac{c(v)}{t}\right] = \frac{1}{t^2} \sum_{u \in V \setminus \{v\}} t\alpha(u, v)(1 - t\alpha(u, v))$$

$$\leq \frac{1}{t} \sum_{u \in V \setminus \{v\}} \alpha(u, v) = \frac{C_\alpha(v)}{t}. \qquad \square$$

Again, a disadvantage of Theorem 4 is that it relies on an unknown value of $C_\alpha(v)$. In most applications, we are only concerned about vertices with high centrality values. For such vertices, we have the following bound.

Corollary 5: *Let $v \in V$ be a vertex with $C_\alpha(v) \geq C_\alpha$. Then, we have*

$$\frac{\sigma(\hat{C}_\alpha(v))}{C_\alpha(v)} \leq \frac{1}{\sqrt{k_t}}.$$

Proof. By Theorem 4, $\sigma(\hat{C}_\alpha(v))/C_\alpha(v) \leq 1/\sqrt{tC_\alpha(v)} \leq 1/\sqrt{tC_\alpha} = 1/\sqrt{k_t}$. $\square$

Handling Large t

Theorem 4 states that a small standard deviation can be obtained by increasing the value of t. However, the RRB method is only defined when $t\alpha(1) \leq 1$, which may not always be the case. To address this issue, when $t\alpha(1) > 1$, we consider the following strategy. First, choose a large constant $N \in \mathbb{N}$ with $t\alpha(1)/N \leq 1$ and run the RRB method N times with parameter t/N. Let $\hat{C}^1_\alpha(v), \dots, \hat{C}^N_\alpha(v)$ be the obtained estimates for $C_\alpha(v)$. Then, the output is the average $\hat{C}_\alpha(v) := \sum_{i=1}^N \hat{C}^i_\alpha(v)/N$.

Since the random variables $\hat{C}^1_\alpha(v), \dots, \hat{C}^N_\alpha(v)$ are independent, by Theorem 4,

$$\sigma^2(\hat{C}_\alpha(v)) = \mathbf{Var}\Big[\frac{1}{N}\sum_{i=1}^N \hat{C}^i_\alpha(v)\Big] \leq \frac{N}{N^2}\frac{C_\alpha(v)}{t/N} = \frac{C_\alpha(v)}{t}$$

as desired (see the correspondence with Theorem 4).

Bootstrap Method

If the value of k_t is set to desired $k^* \in \mathbb{R}_+$, the time complexity and standard deviations can be controlled using Corollaries 2 and 5, respectively. However, the RRB method takes the parameter t; it is hard to determine t such that $k_t = k^*$. To address this issue, we present a "bootstrap" method, which estimates k_t by running Algorithm 1 on small t values and then uses the estimates to choose an appropriate t.

From the counters $\{c(v)\}_{v \in V}$ computed by Algorithm 1 and with parameter t, we adopt $\hat{k}_t = \sum_{v \in V} c(v)/n$ as an estimator of k_t.

Algorithm 2: Random-radius ball (RRB) method with bootstrapping.

```
Input: A graph $G = (V, E)$, a distance-decay function
       $\alpha : \mathbb{N} \to \mathbb{R}_+$, parameters $k^* > 0$ and $s > 0$
Procedure RRB-BOOTSTRAP($G, \alpha, k^*, s$)
1   $t_0 \leftarrow \alpha(1) \cdot k^*/n$.
2   for $i \in \mathbb{Z}_+$ do
3       $t_i \leftarrow 2^i t_0$.
4       $\{\hat{C}_\alpha(v)\}_{v\in V} \leftarrow \mathrm{RRB}(G, \alpha, t_i)$.
5       $\hat{k}_{t_i} \leftarrow \sum_{v\in V} \hat{C}_\alpha(v) \cdot t_i/n$.
6       if $\hat{k}_{t_i} \geq k^* + s\sqrt{k^*}$ then
7           Break the loop.
8   return $\{\hat{C}_\alpha(v)\}_{v\in V}$.
```

Proposition 6: *$\hat{k}_t$ is an unbiased estimator of k_t.*

Proof. The result follows immediately from the definition of k_t. □

Next, we bound the standard deviation of $\hat{k}_t$. Although $c(u)$ and $c(v)$ may not be independent even when $u \neq v$, a careful discussion gives us the following bound on $\sigma(\hat{k}_t)$.

Theorem 7: *It holds that*

$$\sigma(\hat{k}_t) \leq \sqrt{k_t}.$$

Proof. For vertices $u, v \in V$, let $I(u, v) \in \{0, 1\}$ be the indicator random variable corresponding to the event that $c(v)$ is incremented when processing u. For a vertex $u \in V$, let $X(u) = \sum_{v\in V\setminus\{u\}} I(u, v)$. It follows that $\hat{k}_t n = \sum_{v\in V} c(v) = \sum_{u\in V} X(u)$. Since $X(u)$ depends only on the rank of u, $X(u)$ and $X(u')$ are independent when $u \neq u'$. Therefore, if the variance of $X(u)$ is obtained, we can calculate the variance of $\hat{k}_t n$ by simply summing its values.

Thus, the variance of $X(u)$ is bounded. An issue here is that $I(u, v)$ and $I(u, v')$ may not be independent again even when $v \neq v'$. Indeed, if $\alpha(u, v) \leq \alpha(u, v')$ and $I(u, v) = 1$, we always have $I(u, v') = 1$. Thus, we consider the covariance between $I(u, v)$ and $I(u, v')$. For any $v, v' \in V$ with $v \neq u \neq v'$, it follows that

$$\mathbf{Cov}[I(u, v), I(u, v')] \leq t \cdot \min\{\alpha(u, v), \alpha(u, v')\}$$

using the fact that the covariance between two (dependent) Bernoulli random variables with success probabilities p_1 and p_2 does not exceed $\min\{p_1, p_2\}$. Therefore,

$$\begin{aligned}\mathbf{Var}[X(u)] &= \sum_{v,v'\in V: v\neq u\neq v'} \mathbf{Cov}[I(u, v), I(u, v')] \\ &\leq (n-1) \sum_{v\in V\setminus\{u\}} t\alpha(u, v) \leq n \cdot \mathbf{E}[X(u)].\end{aligned}$$

By summing these values, we obtain $\mathbf{Var}[\hat{k}_t n] \leq k_t n^2$, and it follows that $\sigma(\hat{k}_t) = \sqrt{k_t}$. □

Recall that k^* is the desired value for k_t, so let t^* be the value such that $k_{t^*} = k^*$. We now consider the following *doubling-t strategy* to approximate t^*. The idea of this strategy is to repeatedly run Algorithm 1 by doubling t until the value of the obtained $\hat{k}_t$ becomes sufficiently larger than k^*. To realize this idea, we need to address two issues, which are described below.

The first issue is how to choose the initial value of t. In fact, any initial t will suffice as long as the corresponding k_t is smaller than k^*. Hence, for example, we can start with $t_0 := \alpha(1) \cdot k^*/n$. For notational simplicity, we let $t_i = 2^i t_0$.

The second issue is how we decide that $\hat{k}_t$ is sufficiently large. Since the standard deviation of $\hat{k}_t$ is bounded by $\sqrt{k_t}$ by Theorem 7, a reasonable solution is to regard $\hat{k}_t$ as sufficiently large when $\hat{k}_t \geq k^* + s\sqrt{k^*}$ for some parameter $s > 0$. Algorithm 2 summarizes the discussion above.

We analyze the quality of the k_{t_i} value used to estimate the closeness centrality.

Lemma 8: *When Algorithm 2 breaks the loop at Line 7, the integer i satisfies $k_{t_i} \geq k^*$ with probability at least $1 - 4/k^* - 1/s^2$.*

Proof. By Chebyshev's inequality, the probability that $\hat{k}_t > k_t + p\sqrt{k_t}$ is at most $1/p^2$. Hence, as long as i satisfies $k_{t_i} < k^*$, the probability that we break the loop at Line 7 is at most $\min\{k_{t_i}/(k^* - k_{t_i})^2, 1/s^2\}$.

Let $i^* \in \mathbb{Z}_+$ be the first i such that $k_{t_i} \geq k^*$. By the union bound, the failure probability is at most

$$\begin{aligned}&\sum_{i<i^*} \Pr[\hat{k}_{t_i} \geq k^* + s\sqrt{k^*}] \leq \sum_{i<i^*} \min\left\{\frac{k_{t_i}}{(k^* - k_{t_i})^2}, \frac{1}{s^2}\right\} \\ &\leq \sum_{i<i^*-1} \frac{k_{t_i}}{(k_{t_{i^*}}/2)^2} + \frac{1}{s^2} = \sum_{i<i^*-1} \frac{1}{2^{i^*-i}} \frac{4}{2^{i^*} t_0 C_\alpha} + \frac{1}{s^2} \qquad \text{(By (1))} \\ &\leq \frac{4}{k^*} + \frac{1}{s^2}.\end{aligned}$$

In the last inequality, we used $2^{i^*} t_0 C_\alpha = k_{t_{i^*}} \geq k^*$. □

By Combining Corollary 5 and Lemma 8, we obtain the following. Note that we can determine the values of k^* and s by ourselves as opposed to t. Hence, we can make the success probability arbitrarily close to one.

Theorem 9: *Let $v \in V$ be a vertex with $C_\alpha(v) \geq C_\alpha$ and let $\hat{C}_\alpha(v)$ be the estimate of $C_\alpha(v)$ obtained by Algorithm 2. Then, we have*

$$\frac{\sigma(\hat{C}_\alpha(v))}{C_\alpha(v)} \leq \frac{1}{\sqrt{k^*}}.$$

with probability at least $1 - 4/k^* - 1/s^2$.

An advantage of the doubling-t strategy is that its total time complexity remains low.

Theorem 10: *If $\max_{v\in V} C_\alpha(v) \leq KC_\alpha$ holds for some $K \in \mathbb{R}_+$, then Algorithm 2 runs in $O(K(k^* + (2 + s)\sqrt{k^*})m)$ time.*

Proof. By Chebyshev's inequality, the probability that $\hat{k}_t < k_t - 2\sqrt{k_t}$ is at most $1/4$. Hence, once i satisfies $k_{t_i} > k^* + (s+2)\sqrt{k^*}$, we break the loop at Line 7 with probability at least $1/4$ (in every subsequent iteration).

Let $i^* \in \mathbb{Z}_+$ be the first i such that $k_{t_i} > k^* + (2+s)\sqrt{k^*}$. Then by Corollary 2, the expected time complexity is

$$O\Big(K \sum_{i \le i^*} k_{t_i} m\Big) + O\Big(K \sum_{i > i^*} k_{t_i} m \big(\frac{1}{4}\big)^{i-i^*}\Big)$$

$$= O\Big(K \sum_{i \le i^*} t_i C_\alpha m\Big) + O\Big(K \sum_{i > i^*} t_i C_\alpha m \big(\frac{1}{4}\big)^{i-i^*}\Big) \quad \text{(By (1))}$$

$$= O(K t_{i^*} C_\alpha m) + O\Big(K \sum_{i > i^*} t_{i^*} C_\alpha m \big(\frac{1}{2}\big)^{i-i^*}\Big)$$

$$= O(K t_{i^*} C_\alpha m) = O(K(k^* + (2+s)\sqrt{k^*})m). \qquad \square$$

Optimality of the RRB method

In Algorithm 1, the counter $c(v)$ is incremented by $u \in V$ with probability $t\alpha(u,v)$. A natural question that arises is whether Algorithm 1 can be improved by changing the probability. In this section, we will show that the strategy used in Algorithm 1 is optimal in a certain framework.

We change the probability that the counter $c(v)$ is incremented by $u \in V$ from $t\alpha(u,v)$ to $\beta(u,v)$, where $\beta : \mathbb{N} \to \mathbb{R}_+$ is an arbitrary monotone decreasing function such that $\lim_{d\to\infty} \beta(d) = 0$. For each $u \in V$, such a strategy can be achieved by incrementing $c(v)$ for every vertex with $d(u,v) \le \tau$, where $\tau = \lfloor \beta^{-1}(r_u) \rfloor$ and r_u is a random value sampled from $[0,1]$. Let $\beta(u,v)$ stand for $\beta(d(u,v))$, and let $S(v) \subseteq V$ be the random set of vertices u such that $c(v)$ is incremented by u. As long as there does not exist a vertex u with $\alpha(u,v) > 0$ and $\beta(u,v) = 0$, an unbiased estimator can be given as $\hat{C}^\beta_\alpha(v) = \sum_{u \in S(v)} \alpha(u,v)/\beta(u,v)$. We call this method the *β-RRB method*.

When discussing the optimality of a strategy, it is natural to fix the value of $\sum_{u \in V\setminus\{v\}} \beta(u,v)$ to some $T_v \in \mathbb{R}_+$ for each $v \in V$ because T_v corresponds to the expected value of $c(v)$ and will affect the overall time complexity of the β-RRB method. The following theorem states that the estimator $\hat{C}^\beta_\alpha$ has the smallest standard deviation when β is proportional to α.

Theorem 11: *Let $v \in V$ be a vertex and let $\beta : \mathbb{N} \to \mathbb{R}_+$ be a monotone decreasing function. If $\sum_{u \in V\setminus\{v\}} \beta(u,v) = T_v$, then $\sigma(\hat{C}^\beta_\alpha(v))$ takes the minimum value when $\beta(d) = \alpha(d) \cdot T_v / C_\alpha(v)$.*

Thus, Algorithm 1 is optimal in this sense. The proof of Theorem 11 is omitted due to the space limitation.

Experiments

In this section, we show our experimental results and the superiority of the RRB method(s).

Methods. We consider four methods: the RRB method (RRB, Algorithm 1), the RRB method with bootstrapping (RRB-BS, Algorithm 2), HyperBall (HB) (Boldi and Vigna 2013), and All-Distances Sketches (ADS) (Cohen 2015). For RRB-BS, we set $s = 3$. For all of these methods, we can control the trade-off between the time complexity and accuracy by varying select parameters. Throughout all of the experiments, we estimated the harmonic centrality of the vertices, i.e., the distance-decay function $\alpha(x)$ was set to $1/x$. To evaluate the (normalized) RMSE of an estimate, we ran an algorithm 100 times with different random seeds and then took the average.

The *permutation-rank technique* (Cohen 2015) is the technique of randomly and bijectively assigning values from an n-sized set to vertices, instead of uniformly and independently sampling values from $[0,1]$. It is known that this technique improves the performance of ADS although there is no theoretical guarantee. We can also adopt this technique when assigning ranks to vertices in RRB and RRB-BS; that is, we randomly and bijectively assign values from $\left\{\frac{1}{2n}, \frac{3}{2n}, \ldots, \ldots, \frac{2n-1}{2n}\right\}$ to the vertices. We call these techniques RRB (PR) and RRB-BS (PR), respectively.

Environment. All of the experiments were conducted on a machine with two Intel Xeon E5540 processors and 48 GiB of main memory. Each processor has four cores (2.53GHz), but all of the programs ran as single-threaded programs. We

Table 2: Networks used in our experiments.

Name	Type	n	m
Wiki-Vote	Social (d)	7,115	103,689
email-Enron	Social (u)	36,692	367,662
soc-Slashdot0902	Social (d)	82,168	948,464
web-NotreDame	Web (d)	325,729	1,497,134
web-Google	Web (d)	875,713	5,105,039
com-youtube	Social (u)	1,134,891	5,975,248
dblp-2011	Social (u)	933,258	6,707,236
ego-Gplus	Social (d)	107,614	13,673,453
in-2004	Web (d)	1,382,870	16,917,053
soc-Pokec	Social (d)	1,632,803	30,622,564
soc-LiveJournal1	Social (d)	4,847,571	68,993,773
enwiki-2013	Social (d)	4,203,325	101,355,853

Table 3: Runtime and the average of the normalized RMSEs over vertices $v \in V$ with $C_\alpha(v) \ge C_\alpha$ of RRB-BS (PR) with $k^* = 100$.

Graph	Time (s)	Normalized RMSE
Wiki-Vote	0.16	1.95%
email-Enron	0.38	2.87%
soc-Slashdot0902	1.28	2.99%
web-NotreDame	2.23	5.50%
web-Google	24.69	3.90%
com-youtube	20.99	2.38%
dblp-2011	29.65	2.92%
ego-Gplus	10.10	4.83%
in-2004	27.49	5.84%
soc-Pokec	107.93	3.67%
soc-LiveJournal1	278.03	2.54%
enwiki-2013	223.64	1.72%

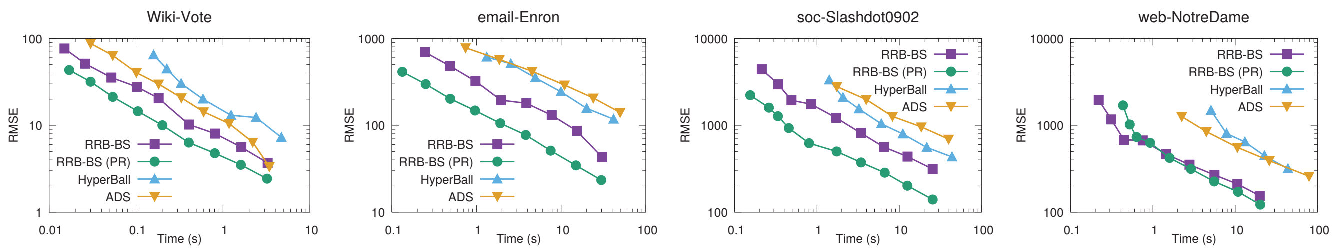

Figure 1: Trade-off between runtime and accuracy.

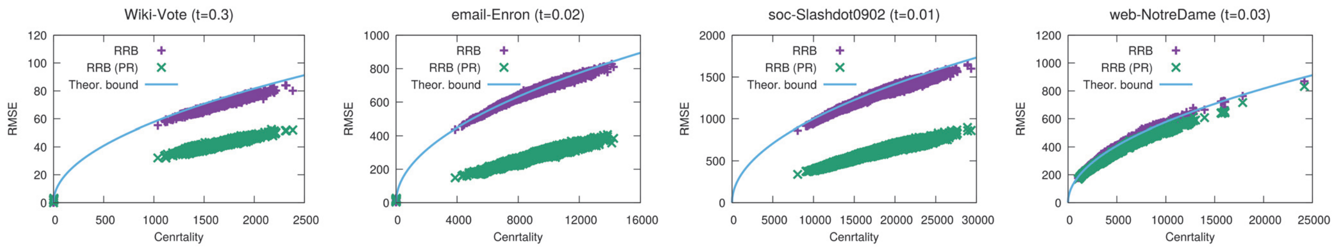

Figure 2: Comparison of the error distribution and theoretical bound.

implemented RRB, RRB-BS, and ADS in C++11 and compiled them with gcc 4.8.2. For HB, we used the WebGraph framework, which is the Java program provided by the authors (Boldi and Vigna 2004).

Networks. All of the networks were collected from the Stanford Large Network Dataset Collection (Leskovec and Krevl 2014) and Laboratory for Web Algorithms (Boldi and Vigna 2004; Boldi et al. 2011). Table 2 summarizes the details of the networks used in the experiments. Here, (d) and (u) denote the corresponding network is directed and undirected, respectively. When counting the number of edges, we regarded an undirected edge as two directed edges.

Comparison with Existing Methods. Figure 1 shows the trade-off between the runtime and the average of the RMSEs over all the vertices when varying the parameters on the four smallest networks. Observe that RRB-BS always outperforms HB and ADS. RRB-BS (PR) shows even better performance. For example, achieving the same accuracy, it runs approximately 20 times faster than HB and ADS on the email-Enron network.

Scalability of RRB-BS. Table 3 shows the runtime of RRB-BS (PR) for $k^* = 100$ on all of the networks. Observe that the runtime increases almost linearly with the number of edges, which demonstrates the high scalability of RRB-BS. In particular, it can handle the enwiki-2013 network, which has 100 million edges, in just four minutes.

Accuracy of RRB-BS. Theorem 9 guarantees that the normalized RMSEs obtained by RRB-BS when $k^* = 100$ for high centrality vertices are at most 0.1. Table 3 shows the average of the normalized RMSEs over vertices $v \in V$ with $C_\alpha(v) \geq C_\alpha$. Notice that it is always at most $0.1 = 10\%$.

Tightness of Theorem 4. Since all of the error bounds are derived from Theorem 4, it is important to investigate their tightness. Figure 2 shows the curve of the RMSEs given by Theorem 4 and the actual error distributions obtained by running RRB and RRB (PR). On all of the networks, the plots for RRB almost lie on the curves given by Theorem 4. Moreover, notice that the RMSE decreases significantly by adopting the permutation-rank technique, except for on the web-NotreDame network. The reason for this exception is unclear, although it may be related to the fact that it is a Web graph, whereas the other networks are social networks.

Conclusions

In this paper, we presented new approximation algorithms, namely, the RRB and RRB-BS methods, for the closeness centrality. RRB-BS allows us to control the trade-off between the time complexity and accuracy. Our experimental results demonstrated that RRB-BS outperforms existing methods such as HyperBall (Boldi, Rosa, and Vigna 2011; Boldi and Vigna 2013) and All-Distances Sketches (ADS) (Cohen 1997; 2015). Although they have a certain guarantee on the relative error for vertices with small centrality values, it is at the cost of a large runtime. In most real-world applications such as vertex ranking, we are mainly concerned about vertices with large centrality values; the RRB-BS method is the best solution in such a case.

Acknowledgement. Y. Y. is supported by JSPS Grant-in-Aid for Young Scientists (B) (No. 26730009), MEXT Grant-in-Aid for Scientific Research on Innovative Areas (No. 24106003), and JST, ERATO, Kawarabayashi Large Graph Project.

References

Abboud, A.; Grandoni, F.; and Williams, V. V. 2015. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1681–1697.

Akiba, T., and Yano, Y. 2016. Compact and scalable graph neighborhood sketching. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 685–694.

Bavelas, A. 1950. Communication patterns in task-oriented groups. *Journal of the Acoustical Society of America* 22(6):271–282.

Bergamini, E.; Borassi, M.; Crescenzi, P.; Marino, A.; and Meyerhenke, H. 2016. Computing top-k closeness centrality faster in unweighted graphs. In *Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*, 68–80.

Boldi, P., and Vigna, S. 2004. The WebGraph framework I: Compression techniques. In *Proceedings of the 13th International Conference on World Wide Web (WWW)*, 595–602.

Boldi, P., and Vigna, S. 2013. In-core computation of geometric centralities with HyperBall: A hundred billion nodes and beyond. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, 621–628.

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

Boldi, P.; Rosa, M.; Santini, M.; and Vigna, S. 2011. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, 587–596.

Boldi, P.; Rosa, M.; and Vigna, S. 2011. HyperANF: Approximating the neighbourhood function of very large graphs on a budget. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, 625–634.

Borgatti, S. P., and Everett, M. G. 2006. A graph-theoretic perspective on centrality. *Social Networks* 28(4):466–484.

Brandes, U., and Pich, C. 2007. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos* 17(7):2303–2318.

Chechik, S.; Cohen, E.; and Kaplan, H. 2015. Average distance queries through weighted samples in graphs and metric spaces: High scalability with tight statistical guarantees. In *Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM)*, 659–679.

Cohen, E.; Delling, D.; Pajor, T.; and Werneck, R. F. 2014. Computing classic closeness centrality, at scale. In *Proceedings of the 2nd ACM conference on Online Social Networks (COSN)*, 37–50.

Cohen, E. 1997. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences* 55(3):441–453.

Cohen, E. 2015. All-distances sketches, revisited: HIP estimators for massive graphs analysis. *Knowledge and Data Engineering, IEEE Transactions on* 27(9):2320–2334.

Dangalchev, C. 2006. Residual closeness in networks. *Physica A: Statistical Mechanics and its Applications* 365(2):556–564.

Eppstein, D., and Wang, J. 2004. Fast approximation of centrality. *Journal of Graph Algorithms and Applications* 8(1):39–45.

Flajolet, P.; Fusy, É.; Gandouet, O.; and Meunier, F. 2007. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. In *Proceedings of the 13th Conference on Analysis of Algorithm (AofA)*, 127–146.

Geisberger, R., and Schultes, D. 2008. Better approximation of betweenness centrality. In *Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*, 90–100.

Hayashi, T.; Akiba, T.; and Yoshida, Y. 2016. Efficient algorithms for spanning tree centrality. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 3733–3739.

Heule, S.; Nunkesser, M.; and Hall, A. 2013. HyperLogLog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology (EDBT)*, 683–692.

Leskovec, J., and Krevl, A. 2014. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data.

Mavroforakis, C.; Garcia-Lebron, R.; Koutis, I.; and Terzi, E. 2015. Spanning edge centrality: Large-scale computation and applications. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, 732–742.

Opsahl, T.; Agneessens, F.; and Skvoretz, J. 2010. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks* 32(3):245–251. https://toreopsahl.com/2010/03/20/.

Riondato, M., and Kornaropoulos, E. M. 2014. Fast approximation of betweenness centrality through sampling. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM)*, 413–422.

Riondato, M., and Upfal, E. 2016. ABRA: Approximating betweenness centrality in static and dynamic graphs with rademacher averages. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1145–1154.