Abstract

Processing large graphs is an emerging and increasingly important computation in a variety of application domains, from social networking to genomics and marketing. One of the important and computationally challenging structural graph metrics is node betweenness centrality, a measure of influence of a node in the graph. The best known algorithm for computing exact betweenness centrality runs in time $O(nm + n^2 \log n)$, which makes it infeasible on graphs with millions of nodes and edges. The existing randomized algorithms for estimating betweenness centrality significantly reduce the execution time, but their accuracy decreases considerably with the size of the graph.

This paper proposes an alternative way to identify nodes with high betweenness centrality. It introduces a new metric, $\kappa$-path centrality, and a randomized algorithm for estimating it, and shows empirically that nodes with high $\kappa$-path centrality have high node betweenness centrality. The randomized algorithm runs in time $O(\kappa^3 n^{2-2\alpha} \log n)$ and outputs, for each vertex $v$, an estimate of its $\kappa$-path centrality up to additive error of $\pm n^{1/2+\alpha}$ with probability $1 - 1/n^2$. Experimental evaluations on diverse real and synthetic social networks show improved accuracy in detecting high betweenness centrality nodes and significantly reduced execution time when compared to known randomized algorithms.

Keywords: betweenness centrality, social network analysis, algorithms, experimental evaluation.

1 Introduction

Social network analysis tools have been used in various fields such as physics, biology, genomics, anthropology, economics, organizational studies, psychology, and IT. The recent phenomenal growth of online social networks exacerbates the need for such tools that are scalable for applications in military, government, and for commercial purposes, to name
only a few. Some of the relevant network metrics are local, such as degree centrality, while others capture global structural properties of the graph, such as the betweenness centrality.

This important global graph metric is a centrality index that quantifies the importance of a node or an edge as a function of the number of shortest paths that traverse it.

Node betweenness centrality is relevant to problems such as identifying important nodes that control flows of information between separate parts of the network and identifying causal links to influence other entities behavior, such as genes in genomics or customers in marketing studies. Betweenness centrality has been used to analyze various social or general networks [JMBO01, LEA+01, OHL04, SWSR08], to identify influential nodes surrounded by other influential nodes in social networks [KOKK03], and to measure network traffic in communication networks [SG05].

Node betweenness centrality, however, is computationally expensive. The best known algorithm for computing exact betweenness centrality of all vertices is Brandes’ algorithm [Bra01], which takes time $O(nm)$ on unweighted graphs and $O(nm + n^2 \log n)$ on weighted graphs. Some randomized algorithms for estimating betweenness centrality have been proposed in the literature [JKL+05, BP07, BKMM07], but the accuracy of these randomized algorithms decreases and the execution time increases considerably with the increase in the network size. Variants of betweenness centrality, such as flow betweenness [FBW91] and random-walk betweenness [New05], take computation time at least of the order $nm$. Thus, existing approaches for determining node betweenness centrality, which work well on small networks, are infeasible for networks with millions of nodes and edges.

We introduce a new approach for identifying highly influential nodes based on their betweenness centrality score, according to the following observations. First, we observe that the value of the betweenness centrality is irrelevant: it is the relative “importance” of nodes (as measured by betweenness centrality) that matters. Second, we observe that for the vast majority of applications, it is sufficient to identify categories of nodes of similar importance: thus, identifying the top 1% most important nodes is significantly more relevant than precisely ordering the nodes based on their relative betweenness centrality. Third, we observe that distant nodes in (social) networks are unlikely to influence each other [Fri83]. Finally, we use the observation that influence may not be restricted to shortest paths [SZ89]. Capturing these observations, we introduce a new distance-based centrality index called $\kappa$-path centrality, present a randomized algorithm for estimating it, and show empirically that nodes with high $\kappa$-path centrality have high betweenness centrality.

The $\kappa$-path centrality of a node $v$ is defined as the sum, over all possible source nodes $s$, of the probability that a message originating from $s$ goes through $v$, assuming that the message traversals are along random simple paths of at most $\kappa$ links; the parameter $\kappa$ is problem dependent and specific to the network. We present a randomized algorithm that estimates the $\kappa$-path centrality index up to additive error of at most $n^{1/2+\alpha}$ for all $n$ nodes with probability at least $1 - 1/n^2$ in time $O(\kappa^3 n^{2-2\alpha} \log n)$. Here, $\alpha \in [-1/2, 1/2]$ is a parameter that controls the tradeoff between accuracy and computation time. Thus, for any positive constant $\alpha \approx 0$ and integer $k \approx \ln(m + n)$, we can achieve approximation accuracy within the sublinear limit $\approx n^{1/2}$ in time $O(mn)$. 2
To better understand the practical advantages of our proposed centrality index, we evaluate the computation of our randomized approximation algorithm on a collection of real social graphs of various sizes. We measure the execution time of the algorithm and its accuracy of identifying high betweenness centrality nodes. We compare these results with those produced by two previously proposed randomized algorithms for estimating betweenness centrality as well as with Brandes’ algorithm. We obtain significant advantages in both accuracy and running time: the accuracy of our algorithm is sometimes twice as much as that of the randomized algorithms for estimating betweenness centrality [JKL+05, BP07, BKMM07], while the running time is up to two orders of magnitude lower.

The contributions of this paper are:

- A new node centrality measure, κ-path centrality, that is intuitively more appropriate for very large social networks because it limits graph exploration to a useful neighborhood of κ social hops around each node. The supporting intuition is twofold: first, in social networks, distant nodes are unlikely to influence each other, and thus the (long) shortest path that connects them is irrelevant in practice. Second, shortest paths are not always the choice for information transmission, as information may travel on less optimal paths.

- A randomized algorithm that estimates the κ-path centrality index for all nodes in a network of size n, up to an additive error of at most n^{1/2+α} with probability at least 1 − 1/n^2 in time O(κ^3n^{2−2α}log n), where α ∈ [−1/2, 1/2] controls the tradeoff between accuracy and computation time.

- An empirical demonstration on a set of real and synthetic social networks that nodes with high κ-path centrality have high betweenness centrality.

- An experimental demonstration that the running time of our randomized algorithm for estimating κ-path centrality is orders of magnitude lower than the runtime of the best known algorithms for computing exact or approximate betweenness centrality, while maintaining higher accuracy, especially in very large networks.

2 Node Betweenness Centrality

Node betweenness centrality is a global centrality index that quantifies how much a vertex controls the information flow between all pairs of vertices in a graph. In this section, we review the formal definition of betweenness centrality and briefly overview algorithms used in the experimental evaluation, that compute exact and approximate betweenness of all vertices in a graph.

2.1 Definition and Notations

Let G = (V, E) be any (directed or undirected) graph, described by the set of vertices V and the set of edges E. The number of vertices (edges) in G is denoted by n (respectively,
Let $W$ be a non-negative weight function on the edges of $G$, where we assume without loss of generality that each edge $e$ of $G$ has $W(e) = 1$ if $G$ is unweighted. We define the length of any path $\rho$ in $G$ as the sum of weights of edges in $\rho$. A shortest path from $s$ to $t$ in $G$ is a path of minimum length, and we denote this length by $d_G(s,t)$. Let $P_s(t)$ denote the set of predecessors of a vertex $t$ on shortest paths from $s$ to $t$ in $G$. Let $\sigma_{st}$ denote the number of shortest paths from $s$ to $t$ in $G$ and, for any $v \in V$, let $\sigma_{st}(v)$ denote the number of shortest paths from $s$ to $t$ in $G$ that go through $v$. Note that $d_G(s,s) = 0$, $\sigma_{ss} = 1$, and $\sigma_{st}(v) = 0$ if $v \in \{s,t\}$ or if $v$ does not lie on any shortest path from $s$ to $t$.

The betweenness centrality index of a vertex $v$ is the summation over all pairs of end vertices of the fractional count of shortest paths going through $v$.

**Definition 2.1 (Betweenness Centrality [Ant71, Fre77])** For every vertex $v \in V$ of a weighted graph $G(V,E)$, the betweenness centrality $C_B(v)$ of $v$ is defined by

$$C_B(v) = \sum_{s \neq v, t \neq v, s} \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

A naive algorithm for computing betweenness centrality of all vertices $v$ of a graph $G$ requires determining for every pair $(s, t)$ of vertices (a) the number of shortest paths $\sigma_{st}$ from $s$ to $t$ in $G$ and (b) the number of shortest paths $\sigma_{st}(v)$ from $s$ to $t$ going through $v$. The values $\sigma_{st}$ for all pairs of vertices $(s, t)$ are determined by applying the shortest path algorithm from each source vertex $s$ and using the recurrence relation $\sigma_{st} = \sum_{v \in P_s(t)} \sigma_{sv}$ during the execution of the algorithm. Once all $\sigma_{st}$ values are computed, $\sigma_{st}(v)$ is computed for each vertex $v \notin \{s,t\}$ using the following fact: $\sigma_{st}(v) = \sigma_{sv} \cdot \sigma_{vt}$ if $d_G(s, t) = d_G(s, v) + d_G(v, t)$ and $\sigma_{st}(v) = 0$ otherwise. This algorithm runs in time $O(n^3)$ and takes space $O(n^2)$ on any (weighted or unweighted) graph $G$ with $n$ vertices and $m$ edges.

### 2.2 Brandes’ Algorithm

Brandes’ algorithm [Bra01] for computing betweenness centrality defines the notion of the dependency score of any source vertex $s$ on another vertex $v$ as $\delta_{sv}(v) = \sum_{t \neq s, v} \frac{\sigma_{st}(v)}{\sigma_{st}}$. Notice that the betweenness centrality $C_B(v)$ of any vertex $v$ can be expressed in terms of dependency scores as follows: $C_B(v) = \sum_{s \neq v} \delta_{sv}(v)$. The following recurrence relation on $\delta_{sv}(v)$ is significant to Brandes’ algorithm:

$$\delta_{sv}(v) = \sum_{u : v \in P_s(u)} \frac{\sigma_{su}}{\sigma_{sv}} (1 + \delta_{su}(u)).$$

The pseudocode of Brandes’ algorithm is presented in Algorithm 1. The algorithm takes a graph $G = (V,E)$ and an array $W$ of edge weights as input and outputs the betweenness centrality $C_B[v]$ of every $v \in V$. In each iteration of the loop in Line 3, the dependency score $\delta[v]$ of $s$ on $v$ is computed for all vertices $v \neq s$ by maintaining the following variables: (a) the length $d[v]$ of the shortest paths from $s$ to $v$, (b) the set of predecessors $P[v]$ of $v$ on the shortest paths from $s$ to $v$, and (c) the number of shortest paths $\sigma[v]$ from $s$ to $v$.
As mentioned earlier, the betweenness centrality $C_B[v]$ of any vertex $v$ is the summation of these dependency scores $δ[v]$ over all source vertices $s$.

In this pseudocode, $Q$ denotes a min-priority queue and $S$ denotes a stack. The procedures $\text{Extract-Min}(Q)$ and $\text{Decrement-Key}(Q,v,d[v])$ implement the usual min-priority queue operations of extracting the element (vertex) from $Q$ with the minimum key ($d[·]$-value) and decrementing the key of any vertex $v$ to $d[v]$, respectively.

**Input**: Graph $G = (V,E)$, Array $W$ of edge weights  
**Output**: Array $C_B$ of betweenness values  

begin  
foreach $v \in V$ do  
$C_B[v] \leftarrow 0$ ;  
end  
foreach $s \in V$ do  
foreach $v \in V$ do  
$d[v] \leftarrow \infty$;  
$σ[v] \leftarrow 0$;  
$P[v] \leftarrow \emptyset$;  
$δ[v] \leftarrow 0$;  
end  
d[s] \leftarrow 0;  
$σ[s] \leftarrow 1$;  
$Q \leftarrow \{s\}$;  
$S \leftarrow \emptyset$;  
end  
while ($Q$ is nonempty) do 
$u \leftarrow \text{Extract-Min}(Q)$;  
push $u$ to $S$;  
foreach $v \in V$ such that $(u,v) \in E$ do  
if ($d[v] == d[u] + W[u,v]$) then  
$σ[v] \leftarrow σ[v] + σ[u]$;  
$P[v] \leftarrow P[v] \cup \{u\}$;  
end  
if ($d[v] > d[u] + W[u,v]$) then  
$σ[v] \leftarrow σ[u]$;  
$P[u] \leftarrow \{u\}$;  
$\text{Decrement-Key}(Q,v,d[v])$;  
end  
while ($S$ is nonempty) do  
pop $u \leftarrow S$ ;  
for $v \in P[u]$ do  
$δ[v] \leftarrow δ[v] + \frac{σ[v]}{W[u]} \cdot (1 + δ[u])$ ;  
if $u \neq s$ then  
$C_B[u] \leftarrow C_B[u] + δ[u]$ ;  
end  
return $C_B$;  
end

**Algorithm 1**: Brandes’ algorithm [Bra01]

The running time of Brandes’ algorithm on weighted graphs is $O(nm + n^2 \log n)$ if $Q$ is implemented by a Fibonacci heap. Using BFS instead of Dijkstra’s algorithm when the input graph is unweighted, the running time of Brandes’ algorithm reduces to $O(nm)$. The space complexity of Brandes’ algorithm (on weighted or unweighted graphs) is $O(m + n)$.

### 2.3 Randomized-Approximate Brandes (or RA-Brandes) Algorithm

Adapting the technique of Eppstein and Wang [EW04] for estimating the closeness centrality, Jacob et al. [JKL+05] and, independently, Brandes and Pich [BP07] proposed a randomized approximation algorithm for estimating the betweenness centrality of all vertices in any given graph. This algorithm, which we refer to as Randomized-Approximate Brandes or in short RA-Brandes, is different from Brandes’ algorithm in only one main respect: Brandes’ algorithm considers dependency scores $δ_n(·)$ of all $n$ start vertices (also
called pivots) $s$, whereas RA-Brandes considers dependency scores of only a multiset $S$ of $\Theta((\log n)/\epsilon^2)$ pivots. The multiset $S$ of pivots is selected by choosing vertices uniformly at random with replacement. The estimated betweenness centrality $\hat{C}_B[v]$ of any vertex $v$ is then defined as the scaled average of these scores: $\hat{C}_B[v] = \frac{\delta}{|S|} \sum_{s \in S} \delta_{ss}(v)$.

The running time of RA-Brandes on unweighted graphs is $O\left(\log n \epsilon^2 (m + n)\right)$, and on weighted graphs is $O\left(\log n \epsilon^2 (m + n \log n)\right)$ if the min-priority queue $Q$ is implemented by a Fibonacci heap. Its space usage (on weighted or unweighted graphs) is $O(m + n)$. The algorithm guarantees computing, for each vertex $v$, an approximation $\hat{C}_B[v]$ that is within $C_B[v] \pm \epsilon n (n - 1)$ with high probability (i.e., probability $1 - 1/n \Omega(1)$).

### 2.4 Adaptive-Sampling Brandes (or AS-Brandes) Algorithm

Bader et al. [BKMM07] proposed a randomized approximation algorithm for estimating the betweenness centrality of all vertices in any given graph. Their algorithm is based on the adaptive sampling technique of Lipton and Naughton [LN89] used in an algorithm for estimating the size of the transitive closure of a directed graph. The adaptive sampling technique requires selecting a multiset of start vertices by sampling vertices adaptively in the sense that the number of vertices chosen varies with the information gained from each sample. Because of its similarity to Brandes’ algorithm and application of adaptive sampling technique, we refer to this algorithm as *Adaptive-Sampling Brandes* or in short *AS-Brandes*.

The algorithm AS-Brandes considers dependency scores of only a multiset $S$ of at most $T$ pivots. It estimates betweenness centrality of any vertex $v$ by noting how fast the sum of dependency scores for $v$ reach a threshold $cn$, where $c \geq 2$ is supplied to the algorithm. To this end, for each vertex $v$, the algorithm maintains a running sum $RS[v]$ of dependency scores $\delta_{ss}(v)$ for pivots $s$ and it records in a variable $k[v]$, the number of pivots used for $v$ until $RS[v]$ becomes greater than $cn$; $k[v]$ is set to $T$ if $RS[v]$ never exceeds $cn$. The estimated betweenness centrality $\hat{C}_B[v]$ of any vertex $v$ is then defined as the scaled average of these scores over $k[v]$ samples: $\hat{C}_B[v] = n \cdot \frac{RS[v]}{k[v]}$.

Since AS-Brandes considers only $T$ pivots while Brandes’ algorithm considers all $n$ pivots, AS-Brandes should be roughly $\Omega(n/T)$ times faster than Brandes’ algorithm. The space usage of AS-Brandes (on weighted or unweighted graphs) is $O(m + n)$. The algorithm guarantees that, for $0 < \epsilon < 0.5$, if the betweenness centrality $C_B[v]$ of a vertex $v$ is at least $n^2/t$ for some constant $t \geq 1$, then with probability at least $1 - 2\epsilon$, its estimated betweenness centrality $\hat{C}_B[v]$ is within $(1 \pm 1/\epsilon) \cdot C_B[v]$ using $ct$ pivots. Thus, with high probability, AS-Brandes gives a good estimate of high betweenness scores (i.e., those of magnitude $\Omega(n^2)$).
**Input** : Graph $G = (V,E)$, Array $W$ of edge weights, cut-off $T$, constant $c \geq 2$

**Output** : Array $\hat{C}_B$ of betweenness estimates

begin
1    foreach $v \in V$ do
2        $\hat{C}_B[v] \leftarrow 0$; $RS[v] \leftarrow 0$; Flag[$v$] $\leftarrow$ false;
3    /* $S$ is a stack and $Q$ is a min-priority queue */
4    for $i$ $\leftarrow 1$ to $T$ do
5        $s$ $\leftarrow$ a vertex chosen uniformly at random from $V$;
6    foreach $v \in V$ do
7        $d[v] \leftarrow \infty$; $\sigma[v] \leftarrow 0$; $P[v] \leftarrow \emptyset$; $\delta[v] \leftarrow 0$;
8        $d[s] \leftarrow 0$; $\sigma[s] \leftarrow 1$; $Q \leftarrow \{s\}$; $S \leftarrow \emptyset$;
9    while $(Q$ is nonempty$)$ do
10       $u$ $\leftarrow$ Extract-Min$(Q)$;
11       push $u$ to $S$;
12       foreach $v \in V$ such that $(u,v) \in E$ do
13          if $(d[v] == d[u] + W[u,v])$ then
14             $\sigma[v] \leftarrow \sigma[v] + \sigma[u]$;
15             $P[v] \leftarrow P[v] \cup \{u\}$;
16          if $(d[v] > d[u] + W[u,v])$ then
18             $\sigma[v] \leftarrow \sigma[u]$; $P[v] \leftarrow \{u\}$;
19             DECREMENT-Key$(Q,v,d[v])$;
20    while $(S$ is nonempty$)$ do
21       pop $u$ $\leftarrow S$ ;
22       for $v \in P[u]$ do
23          $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\frac{n}{|P[u]|}} \cdot (1 + \delta[u])$ ;
24          if ($u \neq s$ and !Flag[$u$]) then
25             $RS[u] \leftarrow RS[u] + \delta[u]$;
26          if $(RS[u] > c \cdot n)$ then
27             $k[u] \leftarrow i$;
28             $\hat{C}_B[u] \leftarrow \frac{n}{k[u]} \cdot RS[u]$;
29             Flag[$u$] $\leftarrow$ true;
30    foreach $v \in V$ do
31          if (!Flag[$v$]) then
32             $k[v] \leftarrow T$; $\hat{C}_B[v] \leftarrow \frac{n}{k[v]} \cdot RS[v]$;
33    return $\hat{C}_B$;
end

Algorithm 2: The algorithm AS-Brandes [BKMM07]

2.5 Variations of Betweenness Centrality

A number of variants of betweenness centrality have been defined that address the fact that information propagates through small paths or that it may spread along paths that need not be the shortest. Some such variants are as follows:

**Bounded-Distance Betweenness or $k$-betweenness**: Borgatti and Everett [BE06] suggested the idea of limiting the length of shortest paths in the definition of betweenness centrality, as they argued that long paths are seldom used for propagation of influence in
some networks. They defined \( k \)-betweenness centrality as an index in which, for each vertex \( v \), its centrality (similar to the case of betweenness) is the sum of dependency scores \( \delta_{s*}(v) \) of all start vertices \( s \) on \( v \), but the dependency scores account for only those shortest paths that are of length at most \( k \).

**Definition 2.2 \((k\text{-Betweenness Centrality})\)** For every vertex \( v \in V \) of a graph \( G = (V, E) \), the \( k \)-betweenness centrality \( C_{B(k)}(v) \) of \( v \) is defined as

\[
C_{B(k)}(v) = \sum_{s,t \in V : d_G(s,t) \leq k} \frac{\sigma_{st}(v)}{\sigma_{st}}.
\]

The \( k \)-betweenness centrality of all vertices of a graph can be computed using Brandes’ algorithm where we stop the underlying single-source shortest path search when a vertex of distance \( k \) from the source is reached. In the worst case, if the shortest path distances from every vertex to all other vertices are no more than \( k \), then the algorithmic complexity will be identical to Brandes’ algorithm.

**Random-Walk Betweenness:** This notion of betweenness, introduced by Newman [New05], assumes that message transmission between any two individuals in a social network follows a random path and, in particular, does not need to take the shortest path. Suppose that a message originating from a source \( s \) is to reach a destination \( t \) via some arbitrary path in an undirected network. The path the message takes can be modeled as an absorbing random walk from \( s \) to \( t \), i.e., a walk from \( s \) to \( t \) that follows a random edge from each current node on the walk until it reaches \( t \), when it terminates. The net flow of this random walk on an edge \( \{x,y\} \) is defined as the absolute difference between the probability that the walk goes from \( x \) to \( y \) and the probability that it goes from \( y \) to \( x \). The net flow of the random walk through vertex \( x \) is defined as one-half of the sum of the net flows on the edges incident to \( x \). The net flow (along an edge or a vertex) is defined in this way so as to discount the possibility that a random walk repeats a vertex or an edge multiple times. The random-walk betweenness of a vertex \( v \) is the expected net flow of a random walk from source \( s \) to destination \( t \) through \( v \), where the expectation is over all possible pairs \((s,t)\).

Newman [New05] showed that the random-walk betweenness is identical to another variant of betweenness called current-flow betweenness. Current-flow betweenness models an undirected network as an electric circuit where nodes represent junctions and edges represent wires of unit resistance. Imagine now that a unit current is injected into the circuit from a junction \( s \) and is extracted from another junction \( t \). It is known that each wire in this circuit will have a unique current flowing through it. The current-flow betweenness of a vertex \( v \) is defined as the amount of current that flows through \( v \) when unit current is injected from \( s \) and extracted from \( t \), averaged over all possible pairs \((s,t)\).

The best known algorithm for computing random-walk betweenness of all vertices takes time \( O(I(n - 1) + mn \log n) \), where \( I(n) = O(n^3) \) is the time for computing the inverse of an \( n \times n \)-matrix [BF05].

**Flow Betweenness:** Freeman et al. [FBW91] introduced flow betweenness based on the concept of maximum flow in a flow network. This centrality notion models any directed
network as a flow network where edges represent pipes that can carry up to unit amount of flow. Imagine a flow that is generated at a source node $s$, transmitted across edges, and absorbed at a sink node $t$. The value of a flow is defined as the total amount of flow generated at $s$ (or, equivalently, the total amount of flow absorbed at $t$) and the amount of flow through any vertex $x$ is the total amount of flow leaving $x$. Although each edge has unit capacity for carrying flow, the maximum possible flow can be greater than one as multiple edges incident on any node can transmit the flow. A natural question is to determine the quantity of the flow through a particular vertex $v$ assuming that the flow transmitting from $s$ to $t$ has the maximum possible value. (In case this quantity is not unique because more than one solutions exist for the $st$-maximum flow problem, then we seek for the maximum flow through $v$ over all possible solutions.) The flow betweenness of a vertex $v$ is defined as the average of this quantity over all possible source-sink pairs $(s, t)$.

The flow betweenness of all vertices can be computed in time $O(m^2n)$ as reported in [New05].

3 $\kappa$-Path Centrality

As introduced in [New05], the random-walk betweenness centrality is based on the traversal of the network with absorbing random walks. Assume the traversal of a message (e.g., news or rumor) originating from some source $s$ over a network and intending to finally reach some destination $t$ in the network along a path, and assume that each node in the network has only its own local view (i.e., has information only of its outgoing neighbors). Thus, when the message is at a current node $v$, the node $v$ forwards the message based on its local view to one of its outgoing neighbors chosen uniformly at random. The message continues to travel in this manner until it reaches the destination node $t$, and then stops.

The notion of $\kappa$-path centrality is based on a similar assumption regarding the random traversal of a message from a source $s$. However, we make two further assumptions in order to reduce the computation time without deviating much from the above random walk model. First, we consider message traversals along simple paths only, i.e., paths in which vertices do not repeat. As non-simple paths do not correspond to the intuitive notion of ideal message traversals in a social network, their consideration in the computation of centrality indices is a noisy factor. To discount non-simple paths, we assume that each intermediate node $v$ on a partially traversed path forwards the message to a neighbor chosen randomly, with probability inversely proportional to edge weights, from the current set of unvisited neighbors; the message traversal is assumed to stop if all the outgoing neighbors of the current node $v$ already appear in the path up to $v$. Although choosing a random neighbor in this manner at each step requires the premise that the message carries the history of the path traversed so far, this premise is needed to express the average contribution of any simple path in the overall information flow and to efficiently simulate such random simple paths. Second, we assume that the message traversals are only along paths of at most $\kappa$ links (edges), where $\kappa$ is a parameter dependent on the network. It has been found in many studies on social networks that message traversals typically take paths containing
few links [Fri83], and so this seems to be a reasonable assumption in the context of social networks. Based on these assumptions, we define \( \kappa \)-path centrality:

**Definition 3.1 (\( \kappa \)-Path Centrality)** For every vertex \( v \) of a graph \( G = (V, E) \), the \( \kappa \)-path centrality \( C_\kappa(v) \) of \( v \) is defined as the sum, over all possible source nodes \( s \), of the probability that a message originating from \( s \) goes through \( v \), assuming that the message traversals are only along random simple paths of at most \( \kappa \) edges.

### 3.1 A Formal Analysis

Consider an arbitrary simple path \( \rho_{s,\ell} \) with start vertex \( s \) and having \( \ell \) links. Let \( s, u_1, u_2, \ldots, u_{\ell-1}, u_\ell \) denote the vertices in the order they appear in \( \rho_{s,\ell} \) and \( s = u_0 \) for convenience. For every \( 0 \leq i \leq \ell \), let \( (s, u_1, \ldots, u_{i-1}, u_i) \) denote the subpath from \( s \) to \( u_i \), and let \( \Pr[\rho_{s,i}] \) denote the probability that a message originating from \( s \) traversed through the path \( \rho_{s,i} \). The probability \( \Pr[\rho_{s,\ell}] \), as shown below, is equal to the product of individual probabilities associated with the random transitions of the message between successive nodes of \( \rho_{s,\ell} \). The exact expression of \( \Pr[\rho_{s,\ell}] \) depends on whether the graph is weighted or unweighted; so we consider these two cases separately.

Consider the case of an unweighted, directed graph in which \( \rho_{s,\ell} \) is a simple path from \( s \) to \( u_\ell \). For every \( 0 \leq i \leq \ell \), let \( N(u_i) \) denote the set of outgoing neighbors of \( u_i \). The expression for \( \Pr[\rho_{s,\ell}] \) is given by the following recurrence relation:

\[
\Pr[\rho_{s,i}] = \begin{cases} 
\Pr[\rho_{s,i-1}] \times \Pr[\text{edge } (u_{i-1}, u_i) \text{ is chosen given } \rho_{s,i-1}] & \text{if } i \geq 2 \\
1/|N(s)| & \text{if } i = 1.
\end{cases}
\]

Here, \( \Pr[\text{edge } (u_{i-1}, u_i) \text{ is chosen given } \rho_{s,i-1}] \) denotes the conditional probability that the message is forwarded from \( u_{i-1} \) to \( u_i \), given that the path traversed up to \( u_{i-1} \) is \( \rho_{s,i-1} \). This probability is equal to \( 1/|N(u_{i-1}) - \{s, u_1, u_2, \ldots, u_{i-2}\}| \), since, by our assumption, each node \( u_i \) forwards the message to a node chosen uniformly at random from the unvisited neighbors of \( u_i \). The above recurrence relation easily leads to the following solution:

\[
\Pr[\rho_{s,\ell}] = \prod_{i=1}^{\ell} \frac{1}{|N(u_{i-1}) - \{s, u_1, u_2, \ldots, u_{i-2}\}|}.
\]

Notice from the above expression that the larger the outdegree of a node is, the smaller the probability of the message being forwarded through a specific link is. This observation corresponds to the intuition that, if the intermediates nodes of a path have a high outdegree, then it is less likely for a message originating from the source to take that path in its entirety.

Next consider the case of a weighted, directed graph in which \( \rho_{s,\ell} \) is a simple path from \( s \) to \( u_\ell \). In this case, each edge \( (u_{i-1}, u_i) \) in \( \rho_{s,\ell} \) has a weight \( W(u_{i-1}, u_i) \). Intuitively, the weight of the edge \( (u_{i-1}, u_i) \) quantifies how easily any information from \( u_{i-1} \) can pass to \( u_i \): the smaller the weight of an edge is, the more accessible the endpoint of the edge is. Thus, it is more likely for a message to be forwarded on to a lower weight edge than to be
forwarded on to a higher weight edge from any node. This intuition suggests the following analog of Eq. (1) for the case of weighted graphs:

$$\Pr[\rho_{s,\ell}] = \prod_{i=1}^{\ell} \frac{1/W(u_{i-1}, u_i)}{\sum_{v \in N(u_{i-1}) - \{s, u_1, u_2, \ldots, u_{i-2}\}} 1/W(u_{i-1}, v)}.$$  (2)

Here, the conditional probability that the message is forwarded from $u_{i-1}$ to $u_i$, given that the path traversed up to $u_{i-1}$ is $\rho_{s,i-1}$, is given by the expression within the product symbol. In this expression, the numerator $1/W(u_{i-1}, u_i)$ corresponds to the intuition that the probability of the message traversing the edge $(u_{i-1}, u_i)$ is inversely proportional to the weight of this edge and the denominator is only a normalization factor so that the probabilities sum to one.

With the above expression for $\Pr[\rho_{s,\ell}]$, we now formalize the notion of $\kappa$-path centrality. For any simple path $\rho_{s,\ell}$ originating from $s$ and any $v \neq s$, let

$$\chi[v \in \rho_s] = \begin{cases} 1 & \text{if } v \text{ lies on } \rho_s, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the probability that the message originating from $s$ goes through any vertex $v$ as per our assumptions is given by

$$\sum_{1 \leq \ell \leq \kappa} \sum_{\rho_{s,\ell}:|\rho_{s,\ell}| = \ell} \chi[v \in \rho_{s,\ell}] \cdot \Pr[\rho_{s,\ell}].$$

The first summation is over the link counts $\ell$ of any simple path and the second summation is over all simple paths $\rho_{s,\ell}$ whose link count is exactly $\ell$. In these summations, the contribution $\Pr[\rho_{s,\ell}]$ of any simple path $\rho_{s,\ell}$ is included if and only if $v$ lies on $\rho_{s,\ell}$, as indicated by the expression $\chi[v \in \rho_{s,\ell}] \cdot \Pr[\rho_{s,\ell}]$. Thus, we get an alternative formulation of $\kappa$-path centrality.

**Proposition 3.2 ($\kappa$-Path Centrality)** For every vertex $v$ of a graph $G = (V, E)$, the $\kappa$-path centrality $C_k(v)$ of $v$ is given by

$$C_k(v) = \sum_{s \neq v} \sum_{1 \leq \ell \leq k} \sum_{\rho_{s,\ell}:|\rho_{s,\ell}| = \ell} \chi[v \in \rho_{s,\ell}] \cdot \Pr[\rho_{s,\ell}],$$

where $\Pr[\rho_{s,\ell}]$ is described by Eq. (1) if $G$ is unweighted and by Eq. (2) if $G$ is weighted.

### 3.2 Estimating $\kappa$-Path Centrality

We present a randomized approximation algorithm for estimating the $\kappa$-path centrality of all vertices in any graph. The algorithm takes as input a graph $G = (V, E)$, a non-negative weight function $W$ on the edges of $G$, and parameters $\alpha \in [-1/2, 1/2]$ and integer $\kappa = f(m, n)$, and runs in time $O(\kappa^3 n^2 - 2\alpha \ln n)$. For each vertex $v$, it outputs an estimate of $C_k(v)$ up to an additive error of $\pm n^{1/2 + \alpha}$ with probability at least $1 - 1/n^2$. We refer to
this algorithm as Randomized-Approximate $\kappa$-path or in short RA-$\kappa$-path. The pseudocode of RA-$\kappa$-path is presented in Algorithm 3.

The algorithm performs $T = 2\kappa^2 n^{1-2\alpha} \ln n$ iterations in Line 5 (the expression for $T$ comes from the analysis of the algorithm). In each iteration, a start vertex $s \in V$ and a walk length $\ell \in [1, \kappa]$ are chosen uniformly at random. In every iteration, a random walk consisting of $\ell$ edges from $s$ is performed, which essentially simulates a message traversal from $s$ in $G$ using the assumption made in Definition 3.1. The number of times any vertex $v$ is visited over all the random walks is recorded in a variable $\text{count}[v]$. The estimated $\kappa$-path centrality $\hat{C}_\kappa[v]$ of any vertex $v$ is then defined as the scaled average of the times $v$ is visited over $T$ walks: $\hat{C}_\kappa[v] = \kappa n \cdot \frac{\text{count}[v]}{T}$.

**Algorithm 3**: The algorithm RA-$\kappa$-path

Theorem 3.3 The algorithm RA-$\kappa$-path runs in time $O(\kappa^3 n^{2-2\alpha} \log n)$, takes space $O(m + n)$, and outputs, for each vertex $v$, an estimate $\hat{C}_\kappa[v]$ of $C_\kappa[v]$ up to an additive error of $\pm n^{1/2+\alpha}$ with probability $1 - 1/n^2$. 

Proof  Fix an arbitrary vertex $v \in V$, real $\alpha \in [-1/2, 1/2]$, and integer $\kappa \geq 1$. We define random variables $X_i$, for $1 \leq i \leq T$, corresponding to the $T$ iterations in Line 5 as follows:

$$X_i = \begin{cases} 
1 & \text{if the } i\text{'th random simple path goes through } v, \\
0 & \text{otherwise.}
\end{cases}$$

It is easy to see that when the algorithm terminates $\text{count}[v] = \sum_{i=1}^{T} X_i$. Let us now evaluate the expected value $E[X_i]$ of $X_i$, for any $1 \leq i \leq T$. Since $X_i$ is an indicator random variable, we have $E[X_i] = \text{Pr}[X_i = 1]$, and, by the definition of $X_i$, $\text{Pr}[X_i = 1]$ equals the probability that the $i$'th random simple path goes through $v$. The algorithm chooses a random start vertex $s$ and a random link count $\ell \in [1, \kappa]$, where both are distributed uniformly over their respective sample sets. Thus, for any vertex $s$ and link count $\ell \in [1, \kappa]$, $s$ is chosen as a start vertex and $\ell$ is chosen as a link count with probability $1/\kappa n$. Once $s$ and $\ell$ are fixed, then a path $\rho_{s,\ell}$ of $\ell$ link counts originating from $s$ is traversed with probability $\text{Pr}[\rho_{s,\ell}]$, described by Eq. (1) if $G$ is unweighted and by Eq. (2) if $G$ is weighted. It follows that

$$E[X_i] = \frac{1}{\kappa n} \sum_{s \neq v} \sum_{1 \leq \ell \leq \kappa} \sum_{\rho_{s,\ell} \ni v} \chi[v \in \rho_{s,\ell}] \cdot \text{Pr}[\rho_{s,\ell}],$$

$$= \frac{1}{\kappa n} \mathcal{C}_\kappa[v] \quad \text{(by Proposition 3.2).} \quad \text{(3)}$$

Let us define random variables $Y_i$, for $1 \leq i \leq T$, as $Y_i = \kappa n X_i$. Note that $Y_i$s are independent random variables and each $Y_i$ takes value either 0 or $\kappa n$. Also, note that the estimate of $\mathcal{C}_\kappa[v]$ returned by RA-Kpath algorithm is $\hat{\mathcal{C}}_\kappa[v] = \kappa n \cdot \text{count}[v] = \sum_{i=1}^{T} Y_i / T$. Thus, by linearity of expectation, we get

$$E\left[\frac{\sum_{i=1}^{T} Y_i}{T}\right] = \frac{\kappa n}{T} E\left[\sum_{i=1}^{T} X_i\right]$$

$$= \kappa n \cdot E[X_i]$$

$$= \mathcal{C}_\kappa(v) \quad \text{(by Eq. 3).}$$

Application of Hoeffding bound\(^1\) gives

$$\text{Pr}\left[\left|\frac{\sum_{i=1}^{T} Y_i}{T} - \mathcal{C}_\kappa(v)\right| \geq \xi\right] \leq 2e^{-2T^2\xi^2/(\kappa^2 n^2)}$$

$$= 2e^{-2T\xi^2/(\kappa^2 n^2)}.$$

\(^1\)The Hoeffding bound [Hoe63], a classical result in probability theory, states the following: Let $X_1, X_2, \ldots, X_T$ be independent random variables, such that each $X_i$ ranges over the real interval $[a_i, b_i]$, and let $\mu = E\left[\sum_{i=1}^{T} X_i/T\right]$ denote the expected value of the average of these variables. Then, for every $\xi > 0$, $\text{Pr}\left[\left|\frac{\sum_{i=1}^{T} X_i}{T} - \mu\right| \geq \xi\right] \leq 2e^{-2T^2\xi^2/\sum_{i=1}^{T} (b_i-a_i)^2}$. 

13
Keep the error margin $\xi$ to $n^{1/2+\alpha}$ results in $\Pr[|\hat{C}_\kappa[v] - C_\kappa(v)| \geq \xi] \leq 2e^{-2T/(\kappa^2n^{1-2\alpha})}$. This probability can be made at most $1/n^3$ if $T \geq 2\kappa^2n^{1-2\alpha} \ln n$. Thus, setting $T$ to $2\kappa^2n^{1-2\alpha} \ln n$ yields, for every vertex $v$, an estimate $\hat{C}_\kappa[v]$ of $C_\kappa[v]$ up to an additive error of $\pm n^{1/2+\alpha}$ with probability at least $1 - 1/n^2$.

In each of the $T$ iterations in Line 5, the time spent is $O(\kappa n)$. Thus, the running time of the algorithm is $O(T\kappa n) = O(\kappa^3n^{2-2\alpha} \ln n)$. The space complexity of the algorithm is $O(m + n)$.

4 Experimental Evaluation

In this section, with the aid of experiments we compare the performance of the algorithm RA-$\kappa$-path with that of Brandes’ algorithm in Section 4.4 and also with that of the two betweenness centrality approximation algorithms (RA-Brandes and AS-Brandes) in Section 4.5. These comparisons are carried out to evaluate the accuracy of $\kappa$-path centrality in estimating the relative importance of a node in a network and also exhibit the runtime benefits of RA-$\kappa$-path. These objectives are achieved by comparing the experimental results using the performance metrics defined in Section 4.2. The environment and the data used in our experiments are detailed in Sections 4.1 and 4.3, respectively.

4.1 Experimental Setup

We performed experiments on both real and synthetic networks. The real networks were selected from various online sources. They cover a wide range of application domains and have sizes in the orders of $10^3$ to $10^5$ nodes. The synthetic networks were created using a synthetic social network generator based on the model introduced in [Váz03] and refined later in [SCW+10]. The details of these networks are given in Section 4.3. All the experiments were done on a cluster with identical nodes with two AMD Opteron processors at 2.2 GHz and 4GB of memory. Each job runs on its own processor on a machine, but the jobs on the same machine share access to disk and I/O.

4.2 Performance Metrics

Our comparison with Brandes’ algorithm is done along two metrics. First, we evaluate how accurate the $\kappa$-path centrality metric is in estimating the relative importance of a node as per the betweenness centrality index. For this, we present the correlation between the approximate $\kappa$-path centrality values computed by RA-$\kappa$-path and the exact betweenness centrality values computed by Brandes’ algorithm, which we refer to as RA-$\kappa$-path correlation. We also present the percentage overlap between the top $N\%$ nodes as returned by RA-$\kappa$-path and the top $N\%$ nodes with the highest betweenness centrality as identified by Brandes’ algorithm, which we refer to as top $N\%$ RA-$\kappa$-path. Second, we evaluate the runtime benefits of RA-$\kappa$-path by measuring its speedup over our implementation of Brandes’ algorithm. The speedup of RA-$\kappa$-path on a given network (i.e., RA-$\kappa$-path speedup) is the
ratio of the running time of Brandes’ algorithm and that of RA-κ-path on the same network. In all of our experiments, we vary the parameters, \( \alpha \) and \( \kappa \), used by RA-κ-path.

For our comparison with the two betweenness centrality approximation algorithms described in Section 2 (RA-Brandes and AS-Brandes), we present relative accuracy and running time performances of RA-κ-path, AS-Brandes, and RA-Brandes. The relative accuracy is evaluated in terms of the correlation of the values and of the percentage overlap of the top \( N\% \) nodes as computed by the three algorithms with respect to the exact betweenness centrality values as computed by Brandes’ algorithm on the same network. We show a comparison among RA-κ-path correlation, RA-Brandes correlation, and AS-Brandes correlation. RA-Brandes correlation and AS-Brandes correlation represent the correlations between the values computed by their respective algorithms with the exact betweenness centrality values computed by Brandes’ algorithm on the same network.

We also present a comparison among top \( N\% \) RA-κ-path, top \( N\% \) RA-Brandes, and top \( N\% \) AS-Brandes. Top \( N\% \) RA-Brandes and top \( N\% \) AS-Brandes represent the percentage overlap between the top \( N\% \) nodes as returned by their respective algorithms and the top \( N\% \) nodes with the highest betweenness centrality as identified by Brandes’ algorithm on the same network.

The running time performance is in terms of the speedup of each of the three algorithms over our implementation of Brandes’ algorithm. The speedup of RA-Brandes and AS-Brandes are defined similarly to that of RA-κ-path, and are referred to as RA-Brandes speedup and AS-Brandes speedup, respectively.

4.3 Experimental Data

Real Networks: In our experiments, we considered the following real networks summarized in Table 1.

- **Kazaa** [IRF04]: an interest-sharing graph that is built based on queries in the Kazaa peer-to-peer file-sharing network. In this graph, vertices are Kazaa users and edges connect users who downloaded the same files over some time interval.

- **Email-Enron** [Les09]: an email communication network generated from around half million emails sent within the Enron company. Vertices are email addresses in the data set and undirected edges represent at least one email exchange between the end vertices.

- **P2P-Gnutella31** [Les09]: a sequence of snapshots of the Gnutella peer-to-peer file sharing network from August 31, 2002. Nodes represent hosts in the Gnutella network topology and edges represent connections between the Gnutella hosts.

- **Soc-Epinions1** [Les09]: a who-trust-whom online social network of the general consumer review site “Epinions.com.” Members of the site can decide whether to “trust” each other. Nodes represent the members and edges represent the trust relationships among members.
• Soc-Slashdot0922 [Les09]: a network containing friend/foe links between users of Slashdot, a user-contributed, technology-related news website. Users can tag each other as friends or foes using the “Slashdot Zoo” feature.

• Citation Networks: networks in which vertices represent articles and edges represent citation relationships. We used the following citation networks:
  – SciMet [BM06]: contains data on articles published in or citing Scientometrics in the period from 1978 to 2000. This network is represented as an undirected and an unweighted graph and is relatively sparse.
  – Kohonen [BM06]: contains data on articles with topic “self-organizing maps” or references to “Kohonen T”. This network is represented as an undirected and an unweighted graph and is relatively dense.
  – Cite-HepPh [Les09]: contains data on articles in the e-print arXiv with topic “High Energy Physics Phenomenology” in the period from January 1993 to April 2003 (124 months). This network is represented as a directed graph where a directed edge from $i$ to $j$ represents that article $i$ cited article $j$.

• Collaboration Networks: networks in which vertices are authors and edges represent co-authorship relationships. We used the following collaboration networks that have vastly varying network structure in terms of the number of vertices and the number of edges:
  – Geom [BM06]: contains data on articles from computational geometry. This network is represented as an undirected and a weighted graph where the weight of an edge represents the number of co-authored articles by the end vertices.
  – CA-AstroPh [Les09]: contains data on articles in the e-print arXiv with topic “Astro Physics” in the period from January 1993 to April 2003. This network is represented as an undirected and an unweighted graph.
  – CA-CondMat [Les09]: contains data on articles in the e-print arXiv with topic “Condensed Matter” in the period from January 1993 to April 2003. This network is represented as an undirected and an unweighted graph.
  – Cond-Mat-2005 [New08]: contains data on articles in the e-print arXiv with topic “Condensed Matter” in the period from 1995 to 2005. This network is represented as an undirected and a weighted graph where the weight of an edge represents the number of co-authored articles by the end vertices.

Synthetic Social Networks: In addition to the real networks presented above, we generated a set of synthetic social networks. These synthetic social networks are random networks with properties similar to real social networks. In order to test the performance of RA-κpath on social graphs that maintain consistent social properties with increase in their size, we created a set of synthetic networks using a synthetic social network generator based on the model introduced in [Váz03] and refined in [SCW+10]. We used this generator to produce networks with 1,000, 10,000, 50,000 and 100,000 nodes. The pseudocode of this algorithm is shown in Algorithm 4 in Appendix.
<table>
<thead>
<tr>
<th>Real Networks</th>
<th>Number of Vertices</th>
<th>Number of Edges</th>
<th>Directed/Undirected</th>
<th>Weighted/Unweighted</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kazaa</td>
<td>2,424</td>
<td>13,354</td>
<td>Undirected</td>
<td>Weighted</td>
<td>[IRF04]</td>
</tr>
<tr>
<td>SciMet</td>
<td>2,729</td>
<td>10,416</td>
<td>Undirected</td>
<td>Unweighted</td>
<td>[BM06]</td>
</tr>
<tr>
<td>Kohonen</td>
<td>3,772</td>
<td>112,731</td>
<td>Undirected</td>
<td>Unweighted</td>
<td>[BM06]</td>
</tr>
<tr>
<td>Geom</td>
<td>6,158</td>
<td>11,898</td>
<td>Undirected</td>
<td>Weighted</td>
<td>[BM06]</td>
</tr>
<tr>
<td>CA-AstroPh</td>
<td>18,772</td>
<td>396,160</td>
<td>Undirected</td>
<td>Unweighted</td>
<td>[Les09]</td>
</tr>
<tr>
<td>CA-CondMat</td>
<td>23,133</td>
<td>186,936</td>
<td>Undirected</td>
<td>Unweighted</td>
<td>[Les09]</td>
</tr>
<tr>
<td>Cit-HepPh</td>
<td>34,546</td>
<td>421,578</td>
<td>Directed</td>
<td>Unweighted</td>
<td>[Les09]</td>
</tr>
<tr>
<td>Email-Enron</td>
<td>36,692</td>
<td>367,662</td>
<td>Undirected</td>
<td>Unweighted</td>
<td>[Les09]</td>
</tr>
<tr>
<td>Cond-Mat-2005</td>
<td>40,421</td>
<td>175,693</td>
<td>Undirected</td>
<td>Weighted</td>
<td>[New08]</td>
</tr>
<tr>
<td>P2P-Gnutella31</td>
<td>62,586</td>
<td>147,892</td>
<td>Directed</td>
<td>Unweighted</td>
<td>[Les09]</td>
</tr>
<tr>
<td>Soc-Epinions1</td>
<td>75,879</td>
<td>508,837</td>
<td>Directed</td>
<td>Unweighted</td>
<td>[Les09]</td>
</tr>
<tr>
<td>Soc-Slashdot0922</td>
<td>82,168</td>
<td>948,464</td>
<td>Directed</td>
<td>Unweighted</td>
<td>[Les09]</td>
</tr>
</tbody>
</table>

Table 1: Summary information of the real networks used in this study.

4.4 Performance Comparison with Brandes’ Algorithm

We present the correlation and the speedup results of RA-$\kappa$-path with respect to Brandes’ algorithm for the twelve real networks as well as for the four sizes of synthetic social networks and for $\kappa$ varying from 2 to 20 in the increments of 2 and $\alpha$ varying from 0 to 0.5 in the increments of 0.1. Each of our result for synthetic social networks is averaged over ten independent network instances of the same size.

4.4.1 Real Networks

The correlation and the speedup speedup results for the real networks are presented in Figures 1 to 4, in a series of 3-D graphs illustrating correlation and speedup in comparison to Brandes’ algorithm, for the various values of $\alpha$ and $\kappa$. Overall, we observe a better correlation of RA-$\kappa$-path with respect to Brandes’ algorithm as the value of $\alpha$ is reduced. The best correlation results are found for $\alpha$ equals 0. Nevertheless, at this value of $\alpha$, the speedup of RA-$\kappa$-path in comparison to Brandes’ algorithm suffers the most.

We notice that, for smaller networks (i.e., those with < 4,000 nodes), the maximum RA-$\kappa$-path correlation is between 0.80 and 0.95 and the maximum RA-$\kappa$-path speedup is between 30 and 80. For larger networks (i.e., those with > 10,000 nodes), the maximum correlation is somewhat lower, to about 0.70 to 0.90. Nevertheless, RA-$\kappa$-path speedup is in the order of $10^2$ to $10^5$.

4.4.2 Synthetic Networks

Figures 5 and 6 show the distribution of RA-$\kappa$-path correlation and RA-$\kappa$-path speedup for the four synthetic networks and for the various values of $\alpha$ and $\kappa$. For any fixed $\alpha$, the improvement in RA-$\kappa$-path correlation across different values of $\kappa$ shows that the length of
Figure 1: Correlations and speedups in the Kazaa, the SciMet, and the Kohonen networks
Figure 2: Correlations and speedups in the Geom, the CA-AstroPh, and the CA-CondMat networks
Figure 3: Correlations and speedups in the Cit-HepPh, the Email-Enron, and the Cond-Mat-2005 networks
Figure 4: Correlations and speedups in the P2P-Gnutella31, the Soc-Epinions1, and the Soc-Slashdot0922 networks.
the path \( \kappa \) is crucial to achieve better results. We observe that as \( \alpha \) decreases the correlation increases, and that the lower the value of \( \alpha \) is, the smaller RA-\( \kappa \)-path speedup is. These patterns are evident across all network sizes. The maximum correlation achieved is in the range of 0.70 to 0.93 and the maximum speedup achieved is in the range of \( 10^2 \) to \( 10^6 \), depending on the values of \( \alpha, \kappa \), and the size of the network.

4.5 Performance Comparison with RA-Brandes and AS-Brandes

A general observation from the previous set of experiments is that we can achieve a near optimal performance of RA-\( \kappa \)-path in both correlation and speedup performance metrics when \( \alpha \) is set to 0.2 and \( \kappa \) is set to \( \ln(n+m) \), where \( n \) and \( m \) are the number of vertices and the number of edges in the network. We use these values of \( \alpha \) and \( \kappa \) in the following experiments that compare the performance of RA-\( \kappa \)-path with RA-Brandes and AS-Brandes.

4.5.1 Real Networks

Figure 7 shows the correlation and the speedup results of the three algorithms (RA-\( \kappa \)-path, RA-Brandes, and AS-Brandes) with respect to Brandes’ algorithm. In these results, we set \( \alpha \) to 0.2 and \( \kappa \) to \( \ln(n+m) \) for RA-\( \kappa \)-path, \( \epsilon \) to 0.5 for RA-Brandes, and \( s \) to 20 and \( c \) to 5 for AS-Brandes. This choice of parameters for AS-Brandes has also been used in [BKMM07]. The results demonstrate the superiority of RA-\( \kappa \)-path over the other two algorithms in both performance metrics for most of the real networks examined, when the parameter values are set as above.

We believe that the choice of parameter values \( \epsilon = 0.5 \) and \( s = 20 \) are not suitable for the sizes of the networks we examined. For this reason, we decided to match the speedups of the three algorithms before comparing their correlations. We performed several experiments with various values of \( \epsilon \) (for RA-Brandes) and \( s \) (for AS-Brandes). Finally, we settled on a heuristic that helped us find a set of values for these parameters that closely matched the speedups of the three algorithms with respect to Brandes’ algorithm. This heuristic defined the following choice of values for the two parameters:

\[
\epsilon = \sqrt{4 \times (\text{RA-}\kappa\text{-path speedup}) \times \ln(n)/n} \quad \text{and} \quad s = 2 \times \text{RA-}\kappa\text{-path speedup}.
\]

The intuition for this choice of \( \epsilon \) is as follows: RA-Brandes considers dependency scores of \( \Theta((\ln n)/\epsilon^2) \) pivots while Brandes’ algorithm considers these scores of all \( n \) pivots, and so RA-Brandes speedup can be estimated to \( \Theta(ne^2/\ln n) \); setting this estimate to RA-\( \kappa \)-path speedup yields the above expression for \( \epsilon \). The intuition for the choice of \( s \) follows a similar reasoning.

The results from the matching of speedups are shown in Figure 8. It can be seen from this figure that the speedups of RA-\( \kappa \)-path, RA-Brandes, and AS-Brandes closely match each other whereas their correlation degrades fast. This degradation in correlation is expected as there is an inherent trade-off between speedup and accuracy of results in these algorithms. Nonetheless, RA-\( \kappa \)-path outperforms the other two algorithms in correlation metric with respect to Brandes’ algorithm by a significant factor.
Figure 5: Correlations and speedups in synthetic networks of sizes 1K, 10K and 50K nodes
Figure 6: Correlations and speedups in synthetic networks of size 100K nodes

Figure 7: Speedups and correlations of RA-\(\kappa\)-path, RA-Brandes, and AS-Brandes with respect to Brandes’ algorithm for the real networks. The parameters used are \(\alpha = 0.2\), \(\kappa = \ln(n + m)\), \(\epsilon = 0.5\), \(s = 20\), and \(c = 5\).
Figure 8: Speedups and correlations of RA-$\kappa$path, RA-Brandes, and AS-Brandes with respect to Brandes’ algorithm for the real networks. The speedups of the three algorithms were first matched to set the parameter values and then the algorithms with the parameter values set were run for their correlation.

Table 2: Percentage overlap (on real networks) of the top $N\%$ nodes as computed by the three algorithms with respect to the exact betweenness centrality values. The speedups of the three algorithms were first matched to set the parameter values and then the algorithms with the parameter values set were run for their percentage overlap of the top $N\%$ nodes. The values in bold denote the highest in the respective ($N$-value) category.
Figure 9: Speedups and correlations of RA-κ-path, RA-Brandes, and AS-Brandes with respect to Brandes’ algorithm for various sizes of synthetic social networks. The speedups of the three algorithms were first matched to set the parameter values and then the algorithms with the parameter values set were run for their correlation.

Table 2 shows top $N\%$ RA-κ-path, top $N\%$ RA-Brandes, and top $N\%$ AS-Brandes for all the twelve real networks and for $N = 1, 5,$ and $10$. The results shown are obtained after the algorithms were matched in speedup as mentioned earlier. The results demonstrate a high variability depending on the size and the type of the network. For smaller networks (i.e., those with $< 4,000$ nodes), RA-κ-path outperforms the other two algorithms in identifying the top 5\% and sometimes the top 10\% most important nodes. Similarly, for average-sized networks (i.e., those with 10,000 to 40,000 nodes), RA-κ-path performs better in identifying the top 5\% most important nodes. For larger networks (i.e., those with $> 40,000$ nodes), RA-κ-path performs poorly compared to RA-Brandes and AS-Brandes when top 10\% is considered. This performance deterioration with increasing $N$ could be due to the arbitrary ordering of low κ-path centrality nodes arising from closeness in their values. The nodes that lie in the lower region of the top $N\%$ values given by RA-κ-path tend to have very close values as $N$ gets larger. Thus, the arrangement of nodes in the lower region of the top $N\%$ values is somewhat arbitrary. This possibly accounts for a deterioration in the performance of RA-κ-path with increasing $N$ when compared to RA-Brandes and AS-Brandes.

4.5.2 Synthetic Networks

We repeated our methodology presented in Section 4.5.1 on the synthetic networks. Initially, we set $\alpha$ to 0.2 and $\kappa$ to $\ln(n + m)$ for RA-κ-path, $\epsilon$ to 0.5 for RA-Brandes, and $s$ to 20 and $c$ to 5 for AS-Brandes. But since the parameter values $\epsilon = 0.5$ and $s = 20$ were not suitable for the network sizes we examined, the heuristics defined in Section 4.5.1 were used to fix the values of $\epsilon$ and $s$. In this way, the speedups of the three algorithms were matched. The result from the matching of speedups are shown in Figures 9. It can be seen from this figure
Table 3: Percentage overlap (on synthetic networks) of the top $N\%$ nodes as computed by the three algorithms with respect to the exact betweenness centrality values. The speedups of the three algorithms were first matched to set the parameter values and then the algorithms with the parameter values set were ran for their percentage overlap of the top $N\%$ nodes. The values in bold denote the highest in the respective ($N\%$-value) category.

<table>
<thead>
<tr>
<th>Graph Size</th>
<th>$\kappa$-path</th>
<th>RA-B</th>
<th>AS-B</th>
<th>$\kappa$-path</th>
<th>RA-B</th>
<th>AS-B</th>
<th>$\kappa$-path</th>
<th>RA-B</th>
<th>AS-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>83.0</td>
<td>70.0</td>
<td>65.0</td>
<td>82.4</td>
<td>70.6</td>
<td>69.6</td>
<td>77.3</td>
<td>70.1</td>
<td>69.7</td>
</tr>
<tr>
<td>10000</td>
<td>88.3</td>
<td>58.0</td>
<td>58.4</td>
<td>82.4</td>
<td>67.8</td>
<td>67.8</td>
<td>78.7</td>
<td>78.5</td>
<td>78.5</td>
</tr>
<tr>
<td>50000</td>
<td>86.6</td>
<td>61.6</td>
<td>60.8</td>
<td>81.7</td>
<td>76.5</td>
<td>77.0</td>
<td>77.5</td>
<td>83.5</td>
<td>83.5</td>
</tr>
<tr>
<td>100000</td>
<td>87.5</td>
<td>61.0</td>
<td>60.4</td>
<td>81.4</td>
<td>79.7</td>
<td>79.8</td>
<td>77.1</td>
<td>84.4</td>
<td>84.6</td>
</tr>
</tbody>
</table>

that the speedups of RA-$\kappa$-path, RA-Brandes, and AS-Brandes closely match each other. The three algorithms were executed ten times on ten independent instances of each of the four different network sizes (thus, a total of $10 \times 10 \times 4 = 400$ executions were made). The values shown in Figure 9 and Table 3 for each network size are the average of the values over these multiple independent executions on instances of the same size.

We notice that RA-$\kappa$-path outperforms the other two algorithms in correlation, especially when the size of the network is large (i.e., has $> 10,000$ nodes). Table 3 shows top $N\%$ RA-$\kappa$-path, top $N\%$ RA-Brandes, and top $N\%$ AS-Brandes for the four sizes of synthetic social networks. We observe that the performance of RA-$\kappa$-path is significantly better than that of RA-Brandes and AS-Brandes for small $N$. However, as in the case real networks, there is a slight drop in the performance of RA-$\kappa$-path for larger percentage $N$. This can be attributed to the existence of many nodes in the lower region of the top $N\%$ values given by RA-$\kappa$-path, which results in an arbitrary ordering of these nodes as explained in Section 4.5.1. But, overall, RA-$\kappa$-path performs significantly better than RA-Brandes and AS-Brandes.

5 Summary

In this paper, we introduced a new graph centrality index called $\kappa$-path centrality and presented a randomized algorithm RA-$\kappa$-path for estimating its value for all vertices. Our experimental evaluation demonstrates that this centrality metric can be used to scalably estimate the relative importance of nodes as per the betweenness centrality index: the correlation between the two centrality indices reaches from 0.70 to 0.95 for all network sizes for a speedup gain of up to 6 orders of magnitude for networks with more than 10,000 nodes. Our experiments show that RA-$\kappa$-path is very effective in identifying the top 1% or the top 5% nodes in the exact betweenness score, outperforming previously known approximate betweenness centrality algorithms AS-Brandes and RA-Brandes. The near optimal performance of RA-$\kappa$-path in both correlation and speedup performance metrics can be achieved when its parameters are set to $\alpha = 0.2$ and $\kappa = \ln(n + m)$, where $n$ and $m$
are the number of nodes and the number of edges in the network, respectively.

Through our experiments, we have shown that $\kappa$-path centrality can be used as an alternative to node betweenness centrality since (a) $\kappa$-path centrality closely models the spread of information in a network and allows to quantify the influence of any node in the network and (b) the speedup performance of RA-$\kappa$-path for estimating $\kappa$-path centrality surpasses those achieved by existing methods of computing exact or approximate betweenness centrality values.

Acknowledgment This research was partially supported by the National Science Foundation under Grants No. CNS-0831785 and CNS-0952420. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. The authors would also like to acknowledge the use of the services provided by Research Computing, University of South Florida.

References


A Appendix I

The pseudocode for the generation of synthetic social networks following the algorithm refined in [SCW+10] is given in Algorithm 4 below.

**Input**: The probability \( u (= 0.8 \text{[SCW+10]}) \) of attachment with an existent node, the number of pairs \( \kappa (= 1) \) of existing nodes to be connected, and the desired size \( N \) of the synthetic social network.

**Output**: A graph \( G = (V, E) \) which is a synthetic network with social network power-law properties.

```plaintext
begin
Let \( G = (V, E) \) be a graph with a single node \( x \in V \) (thus, \( V \leftarrow \{x\} \) and \( E \leftarrow \emptyset \));
while \( |V| \leq N \) do
\( u' \leftarrow \) a real number chosen uniformly at random from \([0, 1]\);
if \( u' \leq 1 - u \) then
\( z \leftarrow \) a node chosen uniformly at random from \( V \);
Let \( y \) be a new node;
\( V \leftarrow V \cup \{y\} \);
\( E \leftarrow E \cup \{(y, z)\} \);
Let \( E' \) be the set of potential edges, i.e., the set of all pairs \( (a, b) \) such that \( (a, b) \notin E \);
for \( i \leftarrow 1 \) to \( \kappa \) do
\( (a, b) \leftarrow \) an edge chosen uniformly at random from \( E' \);
\( E \leftarrow E \cup \{(a, b)\} \);
\( E' \leftarrow E' \setminus \{(a, b)\} \);
end
else
\( (a, b) \leftarrow \) an edge chosen uniformly at random from \( E' \);
\( E \leftarrow E \cup \{(a, b)\} \);
end
end
Algorithm 4: Pseudocode for the generation of synthetic social networks following the algorithm refined in [SCW+10].
```