K-Path Centrality: A New Centrality Measure in Social Networks

Adriana Iamnitchi

University of South Florida

joint work with Tharaka Alahakoon, Rahul Tripathi, Nicolas Kourtellis and Ramanuja Simha

Centrality Metrics in Social Network Analysis

- Betweenness Centrality
  - how much a node controls the flow between any other two nodes
- Closeness Centrality
  - the extent a node is near all other nodes
- Degree Centrality
  - the number of ties to other nodes
- Eigenvector Centrality
  - the relative importance of a node

Betweenness Centrality

- measures the extent to which a node lies on the **shortest path** between two other nodes
- betweennes $C_B(v)$ of a vertex v is the summation over all pairs of nodes of the fractional shortest paths going through v.

Definition (Betweenness Centrality)

For every vertex $v \in V$ of a weighted graph $G(V, E)$, the betweenness centrality $C_B(v)$ of v is defined by

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

Applications of Betweenness Centrality

- Unstructured peer-to-peer networks:
  - high betweenness nodes can slow down the performance of the entire network: provisioning them well may improve performance
  - high betweenness nodes see a big part of the traffic: they can be used for network monitoring
- Information dissemination via bulletin boards: high betweenness web pages should post important announcements
- Mobile social networks: high betweenness nodes should be updated promptly for security reasons

Computing Betweenness

Deterministic Exact Algorithms

- A naive algorithm
  - $O(n^3)$ time and $O(n^2)$ space on weighted/unweighted graphs
- Brandes' algorithm
  - $O(nm)$ time on unweighted and $O(nm + n^2 \log n)$ time on weighted graphs; $O(n + m)$ space on both

n: number of vertices

m: number of edges

Computing Betweenness

Randomized Approximate Algorithms

- RA-Brandes algorithm
  - $O(\frac{\log n}{\epsilon^2}(m+n))$ time on unweighted and $O(\frac{\log n}{\epsilon^2}(m+n\log n))$ on weighted graphs
  - guarantees computing, for each vertex v, an approximation $\widehat{\mathcal{C}_B}[v]$ that is within $\mathcal{C}_B[v] \pm \epsilon n(n-1)$ with probability $1-1/n^2$.
- AS-Brandes algorithm
  - for constants $\epsilon < 1/2$ and $t \geq 1$, if $\mathcal{C}_B[v] \geq n^2/t$, then with probability $1-2\epsilon$, $\mathcal{C}_B[v]$ can be estimated within $(1 \pm 1/\epsilon) \cdot \mathcal{C}_B[v]$ with ϵt samples of start vertices.

n: number of vertices

m: number of edges

ϵ: a parameter influencing run-time as well as error

The Challenge

- Computing betweenness centrality is computationally infeasible
  - For large networks
  - For dynamic networks
- Accuracy of known approximate algorithms degrades with network size

Intuition

- Actual betweenness centrality values are often irrelevant
  - Only the relative importance of nodes matters
- Approximately identifying categories of nodes is sufficient
  - Identifying top 1% of nodes is more relevant than precisely ordering them based on their relative betweenness centrality
- Distant nodes are unlikely to influence each other
  - The *Horizon of Observability* result
- Influence may not be restricted to shortest paths
  - Biological and technological networks are different than social networks in this respect

Our Contributions

1. A new node centrality metric: κ-path centrality
   - appropriate for large social networks as it limits graph exploration to neighborhood of κ social hops around each node
2. A randomized algorithm that estimates κ-path centrality
   - estimates up to an additive error of at most $n^{1/2+\alpha}$ with probability at least $1 - 1/n^2$
   - $O(\kappa^3 n^{2-2\alpha} \log n)$ time on weighted/unweighted graphs, where $\alpha \in [-1/2, 1/2]$ controls the tradeoff between accuracy and computation time
3. An empirical demonstration on real and synthetic networks that
   - nodes with high κ-path centrality have high betweenness
   - running time of our κ-path centrality computation is orders of magnitude lower than known betweeness algorithms
   - while maintaining higher accuracy especially in very large networks

Random-walk Betweenness Centrality

- Assume the traversal of a message (news, rumor, etc.) from some source s to a destination t in a social network
- Assume each node has only its own local view (i.e., knows only about its neighbors)
- Each intermediate node v forwards the message to one of its outgoing neighbors at random
- The message traversal stops when it reaches the destination node t

Figure: source - Facebook

K-path Centrality

Two additional assumptions on top of the random-walk model:

1. Message traversals are only along simple paths (i.e., paths that do not repeat)
   - intermediate node v on a partially traversed path forwards message to random unvisited neighbor
2. Message traversals are only along paths of at most κ links (edges), where κ is a parameter dependent on the network
   - consistent with the observation that message traversals typically take paths containing few links

K-path Centrality

Definition (K-Path Centrality)

For every vertex v of a graph $G = (V, E)$, the κ-path centrality $C_k(v)$ 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 κ edges.

Algorithm for Estimating K-Path Centrality

Algorithm RA-κpath

- Initialize $\mathrm{count}[v] \leftarrow 0$ for each vertex v
- For $T \leftarrow 2\kappa^2 n^{1-2\alpha} \ln n$ iterations:
  - Choose a start vertex s uniformly at random
  - Choose a walk length $\ell \in [1, \ldots, \kappa]$ uniformly at random
  - Perform a random walk consisting of ℓ edges from s using the assumptions made earlier (thus simulating a message traversal from s)
  - Increment $\mathrm{count}[v]$ for each visited vertex v
- Estimate the κ-path centrality of v as the scaled average of the times v is visited over T walks: $\widehat{C}_\kappa[v] = \kappa n \cdot \frac{\mathrm{count}[v]}{T}$.

Analysis of RA-κpath

Theorem

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

Methodology

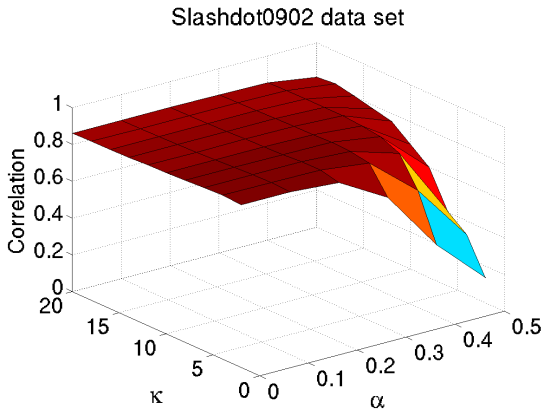
- Implemented Brandes, AS-Brandes, RA-Brandes and RA-κpath algorithms
- Performance metrics:
  - Two performance metrics for accuracy:
    1. Correlation between κ-path and the exact betweenness.
    2. Overlap between the top N% high κ-path and betweenness nodes.
  - Speedup: running time of {RA-κpath, AS-Brandes, RA-Brandes} over running time of Brandes.
- Set of real and synthetically generated social networks
- On AMD Opteron processors at 2.2 GHz and 4GB RAM

Experimental Networks

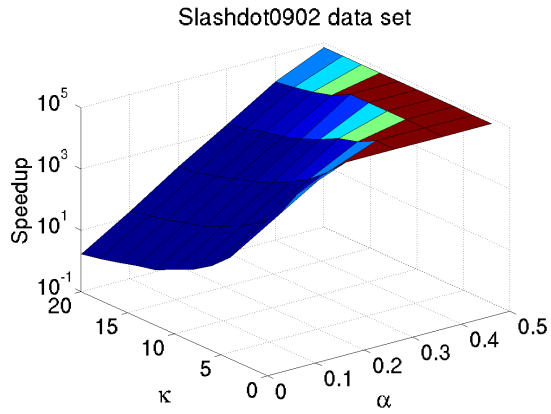
- Real graphs from online sources and our previous research.
- Synthetic social graphs of 1K, 10K, 50K, and 100K nodes.

Networks	Vertices	Edges	Directed/ Weighted	Type
Kazaa	2,424	13,354	D/W	Interest-sharing
Email-Enron	36,692	367,662	U/U	Email
Soc-Epinions1	75,879	508,837	D/U	Social
Soc-Slashdot0922	82,168	948,464	D/U	Social
SciMet	2,729	10,416	U/U	Citation
HepPh	34,546	421,578	D/U	Citation
CondMat	23,133	186,936	U/U	Co-authorship

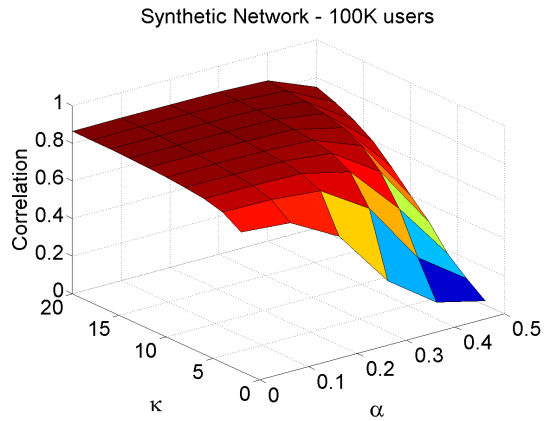
Accuracy and Speedup vs. κ and α



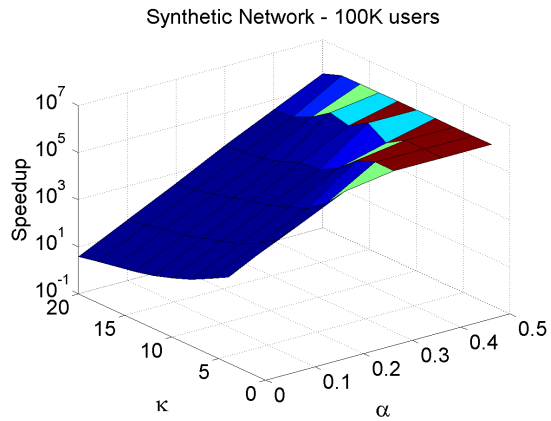
(a) Correlation



(b) Speedup



(c) Correlation

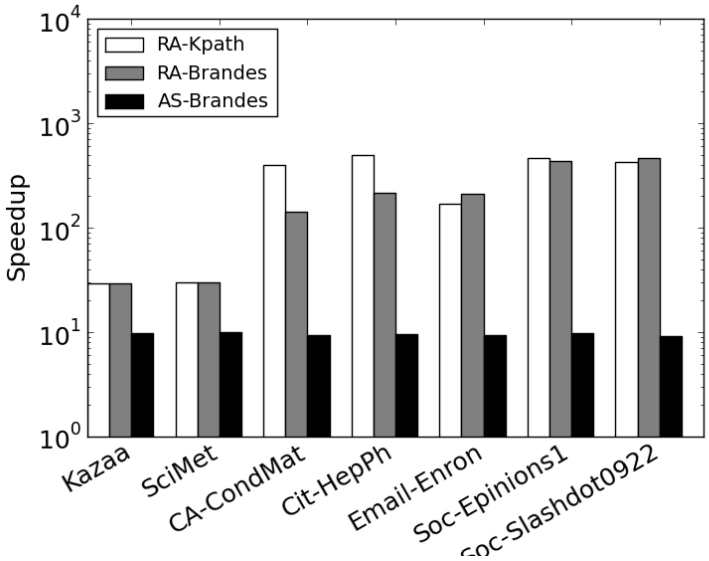


(d) Speedup

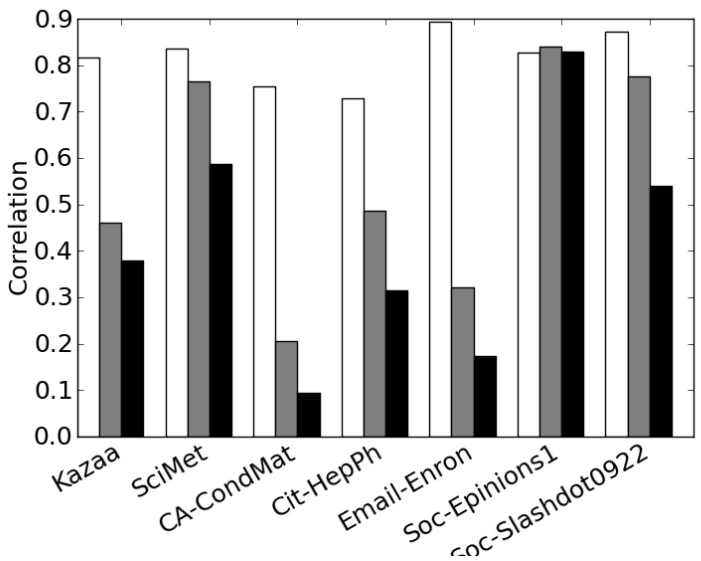
Accuracy in Identifying Top Betweenness Nodes

Network	RA-K 1%	RA-B 1%	AS-B 1%	RA-K 5%	RA-B 5%	AS-B 5%
Kazaa	**79.2**	58.3	58.3	**72.7**	64.5	66.9
SciMet	**85.2**	48.1	44.4	**77.9**	66.2	64.0
CondMat	**74.5**	48.1	48.9	**76.6**	73.2	72.4
HepPh	**71.3**	53.9	47.8	**66.1**	61.2	61.4
Email-Enron	75.1	**79.0**	76.8	63.8	88.5	**89.1**
Soc-Epinions1	**80.6**	70.2	71.0	75.0	**90.2**	90.0
Soc-Slashdot0922	**85.9**	67.4	67.7	85.2	**88.8**	88.3
synth/1K	**83.0**	70.0	65.0	**82.4**	70.6	69.6
synth/10K	**88.3**	58.0	58.4	**82.4**	67.8	67.8
synth/50K	**86.6**	61.6	60.8	**81.7**	76.5	77.0
synth/100K	**87.5**	61.0	60.4	**81.4**	79.7	79.8

Speedup of Randomized Algorithms over Brandes' (1)

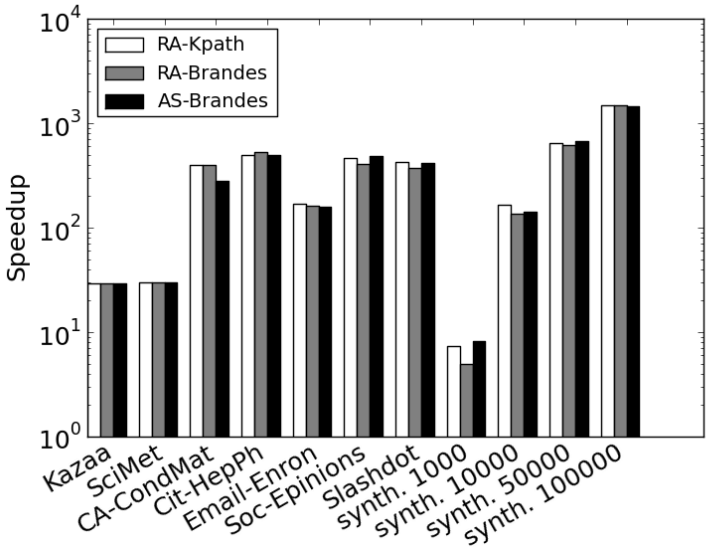


(e) Speedup(unmatched)

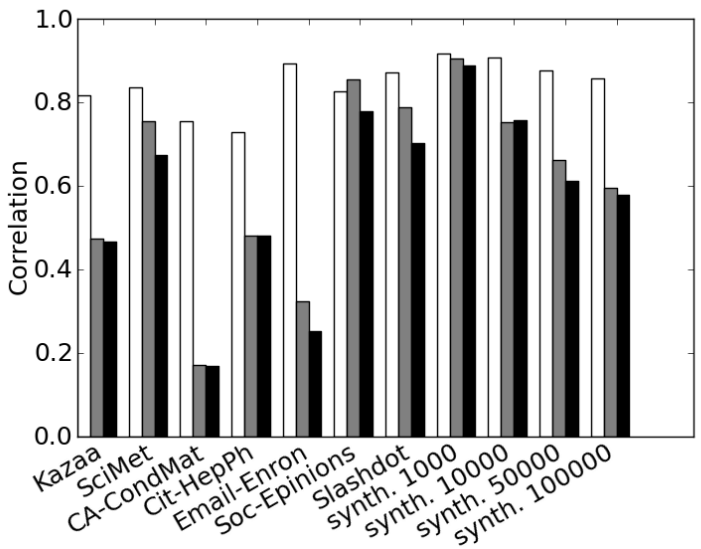


(f) Correlation(unmatched)

Speedup of Randomized Algorithms over Brandes' (2)



(g) Speedup(matched)



(h) Correlation(matched)

Summary

We introduced an alternative centrality metric for betweeness centrality, κ-path centrality, that:

- identifies with high accuracy the top betweenness centrality nodes in a graph;
  - correlation between 0.7 and 0.95 for all networks tested
- is computationally much faster than known algorithms (deterministic or randomized) that compute betweenness centrality;
  - measured speedup of up to 6 orders of magnitude higher than Brandes' for networks of more than 10K nodes

Thank you.
Questions?

Algorithm for Estimating K-Path Centrality

Input : Graph $G = (V, E)$, Array W of edge weights, $\alpha \in [-1/2, 1/2]$, and integer κ
Output: Array $\widehat{C}_\kappa$ of κ-path centrality estimates

begin
  foreach $v \in V$ **do** $\text{count}[v] \leftarrow 0$; $\text{Explored}[v] \leftarrow \text{false}$;
  /* S is a stack and $n = |V|$ */
  $T \leftarrow 2\kappa^2 n^{1-2\alpha} \ln n$; $S \leftarrow \emptyset$;
  for $i \leftarrow 1$ **to** T **do**
    /* simulate a message traversal from s containing ℓ edges */
    $s \leftarrow$ a vertex chosen uniformly at random from V;
    $\ell \leftarrow$ an integer chosen uniformly at random from $[1, \kappa]$;
    $\text{Explored}[s] \leftarrow$ true; push s to S; $j \leftarrow 1$;
    while ($j \leq \ell$ *and* $\exists (s, u) \in E$ *such that* $!\text{Explored}[u]$) **do**
      $v \leftarrow$ a vertex chosen randomly from $\{u \mid (s, u) \in E$ and $!\text{Explored}[u]\}$ with probability proportional to $1/W(s, v)$;
      $\text{Explored}[v] \leftarrow$ true; push v to S; $\text{count}[v] \leftarrow \text{count}[v] + 1$; $s \leftarrow v$; $j \leftarrow j + 1$;
    end
    /* reinitialize $\text{Explored}[v]$ to false */
    while S *is nonempty* **do** pop $v \leftarrow S$; $\text{Explored}[v] \leftarrow \text{false}$;
  end
  foreach $v \in V$ **do**
    $\widehat{C}_\kappa[v] \leftarrow \kappa n \cdot \frac{\text{count}[v]}{T}$;
  end
  return $\widehat{C}_\kappa$;
end