Almost Linear-Time Algorithms for Adaptive Betweenness Centrality Using Hypergraph Sketches

Yuichi Yoshida
National Institute of Informatics, and Preferred Infrastructure, Inc.
• **Centrality** of a vertex ≈ how **important** it is
  Important notion in network science

• Centralities based on shortest paths
  Many s.p. pass through ⇒ the vertex is important
  – Coverage centrality
  – Betweenness centrality
  – Closeness centrality
A pair \((s, t)\) is covered by \(v\): \(s\)-\(t\) s.p. passes through \(v\)

**Coverage centrality** \(C(v)\):

How many fraction of pairs are covered by \(v\)
A pair \((s, t)\) is **covered** by \(v\): \(s\)-\(t\) s.p. passes through \(v\)

**Coverage centrality** \(C(v)\):

How many fraction of pairs are covered by \(v\)
A pair \((s, t)\) is **covered** by \(v\): \(s\)-\(t\) s.p. passes through \(v\)

**Coverage centrality** \(C(v)\):

How many fraction of pairs are covered by \(v\)
A pair \((s, t)\) is covered by \(v\): \(s\)-\(t\) s.p. passes through \(v\).

**Coverage centrality** \(C(v)\):

How many fraction of pairs are covered by \(v\).

**Betweenness centrality** considers how much fraction of \(s\)-\(t\) s.p. pass through \(v\).
Applications of CC / BC

Community detection [Girvan-Newman’s alg]

- Choose the vertex of highest BC.
Community detection [Girvan-Newman’s alg]

- Choose the vertex of highest BC.
Community detection [Girvan-Newman’s alg]

- Choose the vertex of highest BC.
- Remove the vertex
Applications of CC / BC

Community detection [Girvan-Newman’s alg]

• Choose the vertex of highest BC.
• Remove the vertex
• Repeat, say, k times
Community detection [Girvan-Newman’s alg]

- Choose the vertex of highest BC.
- Remove the vertex
- Repeat, say, k times
Community detection [Girvan-Newman’s alg]

- Choose the vertex of highest BC.
- Remove the vertex
- Repeat, say, k times
Applications of CC / BC

- Distance oracle [Akiba et al. SIGMOD’13]
- Targeted immunization

No good to compute CC / BC in the original graph. Seems we need to compute CC / BC adaptively:

for i = 1 to k :

\[ v_i \leftarrow \text{vertex of highest CC / BC} \]

No longer consider pairs covered by \( v_i \)
Adaptive centrality

Previous methods take at least quadratic time...

<table>
<thead>
<tr>
<th></th>
<th>Exact</th>
<th>Approximate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive CC</td>
<td>$O(kn^2m)$</td>
<td>$\tilde{O}(knm/\varepsilon^2)$</td>
</tr>
<tr>
<td>Adaptive BC</td>
<td>$O(knm)$ [Bra01]</td>
<td>$\tilde{O}(km/\varepsilon^2)$ [BP08]</td>
</tr>
</tbody>
</table>

Approximate = approximate CC / BC to within $\pm \varepsilon$
Our contributions

Compute the ordering based on adaptive centrality in almost linear time!

**Time complexity**

- Adaptive CC: $O((n + m)\log n / \varepsilon^2)$
- Adaptive BC: $O((n + m)\log n / \varepsilon^2)$ (in practice)

($\varepsilon$ : error parameter)
Our contributions

Compute the ordering based on adaptive centrality in almost linear time!

Theoretical guarantee
The output is a (1-1/e)-approximation to a certain optimization problem.

Experiments
• CC/BC of our method ≈ CC/BC of exact method.
• sometimes 1000x faster than previous algorithms
Rephrase the problem

• **Coverage centrality** $C(S)$ of a vertex set $S \subseteq V$ := fraction of pairs covered by a vertex in $S$

• $C(v \mid S) := C(\{v\} \cup S) – C(S)$

Want to compute the following ordering

```plaintext
for i = 1 to k:
    $v_i \leftarrow \arg\max_v C(v \mid v_1,\ldots,v_{i-1}).$
```

= **Greedy method** for maximizing $C(\bullet)$
Our method: sketching by hypergraph

Construct a hypergraph $H$ as follows

$$\text{for } i = 1 \text{ to } M = O(\log n/\epsilon^2):$$

- Sample a pair $(s, t)$ uniformly at random.
- Add the hyperedge comprising $s$-$t$ s.p to $H$.

Time complexity: $O((n+m)M)$. 
Our method: sketching by hypergraph

Find the ordering as follows.

\[
\text{for } i = 1 \text{ to } k : \\
v_i \leftarrow \arg\max_v d_{H\setminus\{v_1,\ldots,v_{i-1}\}}(v).
\]

Suffices to keep degrees in a hypergraph
⇒ Time complexity: $O((n+m)M)$. 
Correctness

- $E_{H}[d_{H}(v) / M] = Pr_{st}[d_{H}(v) \text{ increases}] = C(v)$
  \[\Rightarrow C(v) \text{ can be approximated by } d_{H}(v)\]

- $E_{H}[d_{H-S}(v) / M] = Pr_{st}[d_{H-S}(v) \text{ increases}] = C(v \mid S)$
  \[\Rightarrow C(v \mid S) \text{ can be approximated by } d_{H-S}(v)\]
Experiments
Centrality compared to the exact method

Dataset: p2p-Gnutella

\( k \) vs CC

\( k \) vs BC

\( M = 4K \) or \( 16K \) \Rightarrow \text{up to 1\% of error.}
Running time of computing adaptive CC

- $k = 50$

![Graph showing running times for $k = 50$.](chart1.png)

- $k = n$

![Graph showing running times for $k = n$.](chart2.png)

Exact method did not finish in 12 hours.
Running time of computing adaptive CC

- $k = n$

(# of edges < 10M)

$M = 1K$
Summary

- Almost linear time algorithm for adaptive BC/CC.
- Provable guarantee: \((1-1/e)\)-approximation.
- Confirmed accuracy and efficiency by experiment.
- Adaptive BC/CC are now available in applications involving large networks.

Future work
- Even faster
- Other centralities, say, closeness centrality