Computing Classic Closeness Centrality, at Scale

Edith Cohen
Joint with: Thomas Pajor, Daniel Delling, Renato Werneck

Microsoft Research
Very Large Graphs

- Model many types of relations and interactions (edges) between entities (nodes)
  - Call detail, email exchanges, Web links, Social Networks (friend, follow, like), Commercial transactions,…

- Need for scalable analytics:
  - **Centralities/Influence** (power/importance/coverage of a node or a set of nodes): ranking, viral marketing,…
  - **Similarities/Communities** (how tightly related are 2 or more nodes): Recommendations, Advertising, Prediction
Centrality

Centrality of a node measures its importance.

Applications: ranking, scoring, characterize network properties.

Several structural centrality definitions:

- **Betweenness**: effectiveness in connecting pairs of nodes
- **Degree**: Activity level
- **Eigenvalue**: Reputation
- **Closeness**: Ability to reach/influence others.
Closeness Centrality

Importance measure of a node that is a function of the distances from a node to all other nodes.

**Classic Closeness Centrality [(Bavelas 1950, Beaucahmp 1965, Sabidussi 1966)]**

(Inverse of) the average distance to all other nodes

\[ B^{-1}(v) = \frac{n - 1}{\sum_{u \in V} d_{uv}} \]

Maximum centrality node is the 1-median
Computing Closeness Centrality

- Run Dijkstra’s algorithm from source \( v \).
- Compute sum of distances \( \sum_{u \in V} d_{uv} \) from \( v \) to all other nodes

\[
B(v) = \frac{\sum_{u \in V} d_{uv}}{n - 1}
\]

!! Does not scale when we want \( B(v) \) for many or all nodes in a large graph
Centrality of $\nu$ using Dijkstra

Exact, but does not scale for many nodes on large graphs.
Goals

- Scalable algorithm to compute/estimate centrality scores of all nodes
- Accurate: Small relative error: within $(1 + \epsilon)$ with high probability
- Scalable:
  - Processing cost $O(|G|)$ (can depend on $\epsilon^{-1}$)
  - Constant memory per node, independent of $\epsilon$
Algorithmic Overview

- **Approach I:** *Sampling*
  - Properties: good for “close” distances (concentrated around mean)

- **Approach II:** *Pivoting*
  - Properties: good for “far” distances (heavy tail)

- **Hybrid:** Best of all worlds
Approach I: Sampling

- uniform sample $C$ of $k$ nodes
- Ran Dijkstra from each $u \in C$ (Gives us exact $B(u)$ for $u \in C$)
- For $v \in V \setminus C$ estimate $B(v)$ by the average distance to sampled nodes

\[
\hat{B}(v) = \frac{\sum_{u \in C} d_{uv}}{k}
\]
Sampling
$B(\nu)$ ?
Sampling Estimator $\hat{B}(\nu)$
Sampling: Properties

- Unbiased
- Can have large variance -- uniform sample can miss heavy (far) items. Estimate quality depends on distribution of distances from $\nu$

Works well! sample average captures population average
Sampling: Properties

- Unbiased
- Can have large variance -- uniform sample can miss heavy (far) items. Estimate quality depends on distribution of distances from \( v \)

Heavy tail -- sample average has high variance
Approach II: Pivoting

- Uniform sample $C$ of $k$ nodes
- Ran Dijkstra from each $u \in C$ (Gives us exact $B(u)$ for $u \in C$)

For $v \in V \setminus C$, find closest sample node “pivot” $c(v) \in C$.

Estimate using pivot average distance

$$\hat{B}(v) = B(c(v))$$
Pivoting

\[ B(u_1) \]
Pivoting

\[ B(u_2) \]

\[ B(u_1) \]
Pivoting

\[ B(u_1) \]

\[ B(u_2) \]

\[ B(u_3) \]

\[ B(u_k) \]

\[ B(u_4) \]
Pivoting $\hat{B}(\nu)$

Inherit centrality of pivot (closest sampled node)
Pivoting: properties

Estimate is within $\pm d_{vc(v)}$ of true $B(v)$

Proof:

- Triangle inequality: for all $z$,
  \[ d_{c(v)z} - d_{vc(v)} \leq d_{vz} \leq d_{c(v)z} + d_{vc(v)} \]
- Therefore $|B(v) - B(c(v))| \leq d_{vc(v)}$
Estimate is within $\pm d_{vc}(v)$ of true $B(v)$

**WHP upper bound**

$\hat{B}(v) \equiv d_{vc}(v) + B(c(v))$

satisfies

$$B(v) \leq \hat{B}(v) \leq 4B(v)$$

**Proof:** WHP pivot is one of the $\frac{n}{k} \log n$ closest nodes

$$\Rightarrow B(v) \geq \left(1 - \frac{\log n}{k}\right) d_{vc}(v)$$

$$\hat{B}(v) = d_{vc}(v) + B(c(v)) \leq 2d_{vc}(v) + B(v)$$

WHP

$$\leq B(v) \cdot \left(1 + \frac{2}{\log n}\right)$$
Pivoting: properties

- Estimate is within $\pm d_{vc(v)}$ of true $B(v)$

- **WHP upper bound** $B(v) = d_{vc(v)} + B(c(v))$ satisfies

  $$B(v) \leq \hat{B}(v) \leq 4B(v)$$

Bounded relative error for any instance! A property we could not obtain with sampling.
Pivoting vs. Sampling

- Same computation/information:
  - $k$ Dijkstras from a uniform sample
- Different properties on estimate quality:
  - Sampling accurate when distance distribution is concentrated.
  - Pivoting accurate with heavier tail.

But neither gives us a small relative error!

\[
\hat{B}(v) = \frac{\sum_{u \in C} d_{uv}}{k}
\]

\[
\hat{B}(v) = B(c(v))
\]
Hybrid Estimator !!

- Same computation/information as sampling/pivoting ($k$ Dijkstra's from a uniform sample)
- Use sampling to estimate distances from $v$ to “close” nodes
- Use pivot to estimate distances to “far” nodes

How to partition close/far?

Idea: Look at distances of nodes from the pivot $c(v)$ (we have all these distances!)
Hybrid

Partition nodes according to their distance to the pivot $c(v)$:

- **Far nodes**: Nodes $> d_{vc(v)}/\epsilon$ from pivot, use distance to pivot.
  - We have error at most $\pm d_{vc(v)}$ which is at most $1/(\frac{1}{\epsilon} - 1) \approx \epsilon$ contribution to relative error

- **Close nodes**: Nodes within $d_{vc(v)}/\epsilon$ from pivot, estimate using exact distances to sampled nodes
  - Intuition: We “cut off” the heavy tail that was bad for sampling
Hybrid $\hat{B}(v)$

Close nodes $c(v)$

Far nodes
Hybrid $\hat{B}(v)$

Close nodes

c(v)

$v$

6 close nodes (we know how many). Estimate using exact distance from $v$ to 2 closer sampled nodes
11 far nodes (we know which and how many). Estimate using distance from pivot \( c(\nu) \)
Analysis

How to set sample size $k$ ?

Theory: (worse-case distance distribution)

$k \approx \epsilon^{-3} / 2$

($\times \log n$) for small error WHP for all nodes
Analysis (worst case)

- **Far nodes**: Nodes $> d_{\nu C(\nu)}/\epsilon$ from pivot $\approx \epsilon$ contribution to relative error

- **Close nodes**: We need $k \approx \epsilon^{-3}/2$ samples so that NRMSE (normalized standard error) at most $\epsilon$

**Idea**: We estimate $\sum_{\{u \text{ close}\}} d_{uv}$ by $\frac{n}{k} \sum_{\{u \text{ close in } C\}} d_{uv}$

- Each $u \in C$ is sampled with $p = k/n \Rightarrow \text{var} \left( \sum_{\{u \text{ close}\}} d_{\{uv\}} \right) \leq \frac{n}{k} \sum_{\{u \text{ close}\}} d_{\{uv\}}^2$

- Look at worst-case values $d_{uv} \in \left[0, \frac{d_{\nu C(\nu)}}{\epsilon}\right]$ that maximize $\sqrt{\text{var}} / \sum_u d_{uv}$
Analysis

How to set sample size $k$?

Theory: (worse-case distance distribution)
$k \approx \epsilon^{-3}/2$
($\times \log n$) for small error WHP for all nodes

Practice: $k \approx \epsilon^{-2}$ works well.

What about the guarantees (want confidence intervals)?
Adaptive Error Estimation

Idea: We use the information we have on the actual distance distribution to obtain tighter confidence bounds for our estimate than the worst-case bounds.

- **Far nodes:** Instead of using error $\pm d_{vc}(v)$, use sampled far nodes to determine if errors “cancel out.” (some nodes closer to pivot $c(v)$ but some closer to $v$.

- **Close nodes:** Estimate population variance from samples.
Extension: Adaptive Error Minimization

For a given sample size (computation investment), and a given node, we can consider many thresholds for partitioning into closer/far nodes.

- We can compute an adaptive error estimate for each threshold (based on what we know on distribution).
- Use the estimate with smallest estimated error.
Efficiency

Given the $kn$ distances from sampled nodes to all others, how do we compute the estimates efficiently?

- Partition “threshold” is different for different nodes with the same pivot (since it depends on distance to pivot).
- Can compute “suffix sums” of distances with Dijkstra from each pivot, to compute estimates for all nodes in $O(k)$ time per node.
Scalability: Using +O(1)/node memory

- We perform $k$ Dijkstra’s but do not want to store all $kn$ distances.
- In our implementation, we reduce the additional storage to $O(1)$ per node by first mapping nodes to their closest pivots. This is equivalent to performing one more Dijkstra.
**Experimental Evaluation**

| type  | instance   | $|E| \cdot 10^3$ | Exact time ≈ | Sampling err. | Sampling time | Pivoting err. | Pivoting time | Hybrid err. | Hybrid time |
|-------|------------|----------------|--------------|---------------|---------------|---------------|---------------|-------------|-------------|
| road  | fla-t      | 1 344         | 60           | 5.4           | 24.4          | 3.2           | 21.6          | 2.5         | 28.3        |
|       | usa-t      | 28 854        | 44 222       | 2.9           | 849.4         | 3.7           | 736.4         | 2.0         | 2 344.3     |
| grid  | grid20     | 2 095         | 71           | 4.3           | 26.5          | 3.5           | 26.8          | 2.9         | 29.2        |
| trian | buddha-w   | 1 631         | 21           | 3.5           | 16.4          | 2.6           | 15.5          | 2.2         | 18.5        |
|       | del20-w    | 3 146         | 72           | 2.7           | 27.4          | 3.6           | 26.7          | 2.6         | 32.6        |
| game  | FrozenSea  | 2 882         | 38           | 3.0           | 22.1          | 4.1           | 20.2          | 2.1         | 24.0        |
| sensor| rgg20-w    | 6 894         | 160          | 1.6           | 61.2          | 3.8           | 57.1          | 2.1         | 73.3        |
| comp  | Skitter    | 11 094        | 248          | 0.7           | 59.7          | 14.3          | 55.2          | 0.7         | 61.6        |
|       | MetroSec   | 21 643        | 270          | 0.6           | 52.1          | 2.3           | 47.5          | 0.6         | 53.2        |
| social| rws20      | 3 146         | 114          | 0.9           | 45.6          | 3.0           | 41.3          | 0.9         | 49.4        |
|       | rba20      | 6 291         | 133          | 0.8           | 56.8          | 9.7           | 48.4          | 0.8         | 60.2        |
|       | Hollywood  | 56 307        | 227          | 1.0           | 86.5          | 14.6          | 81.8          | 1.0         | 85.7        |
|       | Orkut      | 117 185       | 2 973        | 1.7           | 377.4         | 7.2           | 367.6         | 1.7         | 376.4       |

Hybrid slightly slower, but more accurate than sampling or pivoting.
Experimental Evaluation

- Sampling: less accurate for “high diameter” graphs.
- Pivoting: less accurate for “low diameter” graph.
- Hybrid: Consistently good results (best of both).
Graph: Road network of Florida with travel time metric.
Example Centrality Distribution

**Graph:** Road network of Florida with travel time metric.
Example Centrality Distribution

**Graph:** Road network of Florida with travel time metric.

![Graph showing the distribution of estimated centrality with a peak at around 0.35 and a lower frequency at 0.4]

**Y-axis:** Frequency

**X-axis:** Estimated Centrality
Directed graphs

(Classic Closeness) Centrality is defined as (inverse of) average distance to *reachable* (outbound distances) or *reaching* (inbound distances) nodes only.

- Sampling works (same properties) *when graph is strongly connected*.
- Pivoting breaks, even with strong connectivity. Hybrid therefore also breaks.
- When graph is not strongly connected, basic sampling also breaks – we may not have enough samples from each reachability set.

We design a new sampling algorithm...
...Directed graphs

(Classic Closeness) Centrality is defined as (inverse of) average distance to *reachable* (outbound distances) or *reaching* (inbound distances) nodes only.

Algorithm computes for each node $v$ its average distance to a uniform sample of $k$ nodes from its reachability set. $\tilde{O}(k|G|)$ based on reachability sketches [C’ 1994].

- Process nodes $u$ in random permutation order
- Run Dijkstra from $u$, prune at nodes already visited $k$ times

$$\hat{B}(v) = \text{sum of distances from visiting nodes} / \#\text{visitors}$$
Directed graphs: Reachability sketch based sampling is orders of magnitude faster with only a small error.

| type      | instance    | $|V|_{\cdot10^3}$ | $|E|_{\cdot10^3}$ | time $\approx$ [h:m] | err. [%] | time [sec] |
|-----------|-------------|-----------------|-----------------|-----------------------|---------|------------|
| road      | eur-t       | 18 010          | 42 189          | 28 399:47            | 3.2     | 655.9      |
| web       | NotreDame   | 326             | 1 470           | 0:54                 | 2.4     | 1.5        |
|           | Indo        | 1 383           | 16 540          | 58:46                | 4.1     | 21.1       |
|           | Indochina   | 7 415           | 191 607         | 2 884:19             | 4.7     | 174.7      |
| comp      | Gnutella    | 63              | 148             | 0:02                 | 2.8     | 0.6        |
| social    | Epinions    | 76              | 509             | 0:07                 | 5.4     | 1.1        |
|           | Slashdot    | 82              | 870             | 0:18                 | 2.2     | 2.2        |
|           | Flickr      | 1 861           | 22 614          | 227:01               | 4.3     | 65.1       |
|           | WikiTalk    | 2 394           | 5 021           | 22:01                | 0.5     | 5.4        |
|           | Twitter     | 457             | 14 856          | 28:16                | 1.2     | 26.1       |
|           | LiveJournal | 4 848           | 68 475          | 2 757:01             | 1.9     | 276.8      |
Extension: Metric Spaces

Basic hybrid estimator applies in *any metric space*: Using $k$ single-source computations from a random sample, we can estimate centrality of all points with a small relative error.

**Application**: Centrality with respect to *Round-trip distances in directed strongly connected graphs*:

- Perform both a forward and back Dijkstra from each sampled node.
- Compute roundtrip distances, sort them, and apply estimator to that.
Extension: Node weights

**Weighted centrality:** Nodes are heterogeneous. Some are more important. Or more related to a topic. Weighted centrality emphasizes more important nodes.

\[
B(v) = \frac{\sum_{u \in V} w(u) d_{uv}}{\sum_{u \in V} w(u)}
\]

Variant of Hybrid with same strong guarantees uses a weighted (VAROPT) instead of a uniform nodes sample.
Closeness Centrality

- Classic (penalize for far nodes)

\[ C(i) = (n - 1)/ \sum_j d_{ij} \beta(j) \]

- Distance-decay (reward for close nodes)

\[ C(i) = \sum_j \alpha(d_{ij}) \beta(j) \]

Different techniques required: All-Distances Sketches [C’94] work for approximating distance-decay but not classic.
Summary

- Undirected graphs (and metric spaces): We combine sampling and pivoting to estimate classic closeness centrality of all nodes within a small relative error using $k$ single-source computations.
- Directed graphs: Sampling based on reachability sketches
- Implementation: minutes on real-world graphs with hundreds of millions of edges
Thank you!