ABRA: Approximating Betweenness Centrality through Sampling with Rademacher Averages

Matteo Riondato (Two Sigma Investments LP, matteo@twosigma.com) Eli Upfal (Department of Computer Science, Brown University, eli@cs.brown.edu)

22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'16)

1. What problem are we studying?

Motivation: Identifying the most important vertices in a graph allows us to study properties such as network robustness, information diffusion, and influence propagation, that give important insights on the structure and the dynamics of the graph

Example: Find most influential CEOs, most informed journalists, most central locations in a city, most linked websites, etc.

Let $G = (V, E)$ be a graph, with $|V| = n$ nodes and $|E| = m$ edges

To find the most important, i.e., central, vertices in G we need a *measure of importance*, i.e., a centrality measure that assigns a score to each vertex, based on some of concept of importance (using degree, distance, ...)

Betweenness centrality uses information on Shortest Paths (SP) distance to assign the score

For each pair of different nodes $(u, w) \in V \times V$ let:

- σ_{uw}: no. of Shortest Paths (SP) from u to w
- $\sigma_{uw}(v)$: no. of SPs from u to w that go through a node $v \neq u, w$

Definition: Betweenness Centrality (BC) of $v \in V$:

$$\mathsf{b}(v) = \frac{1}{n(n-1)} \sum_{u,w} \frac{\sigma_{uw}(v)}{\sigma_{uw}}$$

Task: Compute the betweenness centrality score $\mathsf{b}(v)$ for all nodes $v \in V$

2. How can we compute all the BC's exactly?

Naïve Algorithm:
All-Pairs SP + Aggregation (sum for each node)
Takes $\Theta(n^3)$ due to the aggregation

Brandes' Algorithm (BA, [Brandes, 2001]):
For each node $v \in V$:
1) Run Single Source SP from v
2) Backtrack along SP DAG,
appropriately incrementing BC of nodes along the SPs.
Takes $O(n^2 m)$ (unweighted G) or $O(n^2 m + n^2 \log n)$ (weighted G)

Issue: Exact algorithms do not scale with $|V|$

3. How can we speed up BC computation?

Intuition: Trade-off accuracy for speed:
Only perform a few SPs computations between pairs of nodes (u, w) sampled at random

Computed BC values are approximate but fast to compute
Approximation is OK: centrality computation is a exploratory task

Key question:
How many pairs to sample to get an approximation of the desired quality?

Our contribution:
ABRA: a fast approximation algorithm for BC that uses Progressive Random Sampling and Rademacher Averages to obtain probabilistic guarantees on the accuracy of the output

4. How do we define an approximation of the BC's?

Let $B = \{\mathsf{b}(v), v \in V\}$ (B is a bag: $|B| = n$), and $\epsilon, \delta \in (0, 1)$

Definition: (ε, δ)-approximation to B:
A set $\widetilde{B} = \{\widetilde{\mathsf{b}}(v), v \in V\}$ s.t.

$$\Pr(\exists v \in V \ : \ |\widetilde{\mathsf{b}}(v) - \mathsf{b}(v)| > \varepsilon) < \delta$$

ABRA returns an (ε, δ)-approximation to B
It uses Progressive Random Sampling to decide how much to sample

5. What is Progressive Random Sampling (PRS)?

How much shall we sample in order to obtain an (ε, δ)-approximation?

State of the art (RK [R. and Kornaropoulos, 2015]):

- sample single SPs at each step: lots of wasted work
- use fixed sample size for worst-case graphs

This work (ABRA):

- samples pairs of nodes, and uses all SPs among them
- PRS: let's start sampling: the data will tell us when to stop

Progressive Random Sampling is an iterative sampling scheme

Outline of a PRS algorithm for BC *centrality*

- Initialize $c(v) = 0$, for all $v \in V$
- Repeat until the stopping condition is satisfied:
  1. sample a pair (u, w) from $V \times V$ uniformly at random
  2. run $s - t$ SP from u to w (e.g., with BFS, Dijkstra, A*, ...)
  3. If w is reached, backtrack from w to u, incrementing $c(v)$ by $\sigma_{uw}(v)/\sigma_{uv}$ for nodes v along the SPs
- Let ℓ be the number of sampled pairs
- Output the collection $\{\widetilde{\mathsf{b}}(v) = c(v)/\ell, v \in V\}$

In reality, multiple samples are collected at each iteration, according to a sample schedule (a sequence of sample sizes), and the stopping condition is only checked once per iteration

ABRA implements the above scheme for PRS algorithms for BC centrality

6. What are the challenges? What is our contribution?

The challenges are:
1) Developing a stopping condition that

- can be checked fast
- guarantees that the output is a (ε, δ)-approximation
- can be satisfied at small sample sizes

2)Devising a method to choose the next sample size

Our contribution: ABRA, the first algorithm that

- uses PRS to obtain an (ε, δ)-approximation of BC
- uses a stopping condition that only requires the solution of an unconstrained convex minimization problem
- does not need to compute any global property of the graph
- computes the optimal next sample size on the fly
- uses Rademacher Averages in a graph mining setting

Previous contributions using PRS gave no guarantees and used predefined sample sizes

7. What do we really need?

Let $\mathcal{S}$ be the collection of pairs sampled by ABRA up to iteration i

We need a way to compute an $\eta_{\mathcal{S}}$ such that

$$\Pr(\sup_{v \in V} |\widetilde{\mathsf{b}}(v) - \mathsf{b}(v)| > \eta_{\mathcal{S}}) < \delta/2^i$$

(division on the r.h.s. due to the need of Union Bound over all iterations)

If we can compute such an $\eta_{\mathcal{S}}$, then the stopping condition can just be:

"is $\eta_{\mathcal{S}} \leq \varepsilon$?"

Question: How to compute $\eta_{\mathcal{S}}$ with the desired properties?

Issue: to compute $\eta_{\mathcal{S}}$, we can only use information obtained from $\mathcal{S}$

Solution: use upper bounds to Rademacher Averages!

8. What are Rademacher Averages?

The behavior of $\sup_{v \in V} |\widetilde{\mathsf{b}}(v) - \mathsf{b}(v)|$ has been extensively studied using VC-dimension, covering numbers, and Rademacher averages

For any node v and any pair of nodes (u, w), let

$$f_v(u, w) = \frac{\sigma_{uw}(v)}{\sigma_{uw}}$$

Let $\mathcal{S} = \{(u_1, v_1), \ldots, (u_n, v_\ell)\}$ be a random sample of pairs of nodes, and let $\phi_1, \ldots, \phi_\ell$ be independent Rademacher r.v. (1 with prob. 1/2, -1 othw.)

Definition: The Rademacher Average on $\mathcal{S}$ is:

$$\mathsf{R}(\mathcal{S}) = \mathbb{E}_\phi \left[\sup_{v \in V} \frac{1}{\ell} \sum_{i=1}^{\ell} \phi_i f_v(u_i, w_i) \mid \mathcal{S} \right]$$

The expectation is taken w.r.t. the ϕ_i only, i.e., conditionally on $\mathcal{S}$

9. How are we using the Rademacher average?

Theorem (Key result from Statistical Learning Theory): Let

$$\eta_{\mathcal{S}} = 2\mathsf{R}(\mathcal{S}) + \sqrt{\frac{2\ln(2^i 3/\delta)}{\ell}}$$

Then

$$\Pr(\sup_{v \in V} |\widetilde{\mathsf{b}}(v) - \mathsf{b}(v)| \leq \eta_{\mathcal{S}}) < \delta/2^i$$

$\eta_{\mathcal{S}}$ is a sample-dependent upper bound to $\sup_{v \in V} |\widetilde{\mathsf{b}}(v) - \mathsf{b}(v)|$

(We actually use a more refined version of this theorem [Oneto et al., 2013])

Key question: how do we compute or bound $\mathsf{R}(\mathcal{S})$ efficiently using only information from $\mathcal{S}$?

Computing $\mathsf{R}(\mathcal{S})$ efficiently allows us to compute $\eta_{\mathcal{S}}$ and check our stopping condition efficiently

10. How do we bound the Rademacher Average?

Given $\mathcal{S}$, for any node $v \in V$, let $\mathsf{q}_{\mathcal{S}}(v)$ be the ℓ-dimensional vector

$$\mathsf{q}_{\mathcal{S}}(v) = (f_v(u_1, w_1), \ldots, f_v(u_\ell, w_\ell)),$$

and let $Q_{\mathcal{S}} = \{\mathsf{q}_{\mathcal{S}}(v), v \in V\}$ ($Q_{\mathcal{S}}$ is a set, hence $|Q_{\mathcal{S}}| \leq n$)

Theorem (Variant of Massart's Lemma):
Let $w : \mathbb{R}^+ \to \mathbb{R}^+$ be the function

$$\rho(s) = \frac{1}{s} \ln \sum_{\mathsf{q} \in Q_{\mathcal{S}}} \exp(s^2 \|\mathsf{q}\|^2 / (2\ell^2))$$

Then

$$\mathsf{R}(\mathcal{S}) \leq \min_{s \in \mathbb{R}^+} \rho(s)$$

Since $\rho(s)$ is convex, its global minimum can be found efficiently

Question:
How to keep track of $Q_{\mathcal{S}}$ and the $\mathsf{q}_{\mathcal{S}}(v)$ efficiently?

Things to note:

- We actually only need the bag of the norms $\|\mathsf{q}\|$ for $\mathsf{q} \in Q_{\mathcal{S}}$
- For each $v \in V$, $\|\mathsf{q}_{\mathcal{S}}(v)\|/\ell = \widetilde{\mathsf{b}}(v)$, so a PRS algorithm is already implicitly tracking $\mathsf{q}_{\mathcal{S}}(v)$
- For each $\mathsf{q} \in Q_{\mathcal{S}}$ there is (at least) one v such that $\mathsf{q} = \mathsf{q}_{\mathcal{S}}(v)$. Hence $Q_{\mathcal{S}}$ induces a partitioning of V
- At the beginning of the algorithm, $Q_{\mathcal{S}} = \{\mathsf{q}_0\}$ with $\mathsf{q}_0 = ()$ (empty vector)
- As more samples are taken, the partitioning is refined by splitting some of the partitions into multiple parts
- When a pair of nodes is sampled, only partitions involving vertices along the computed SPs between the sampled nodes may be split

Solution:
Keep track of the partitioning of V as the algorithm samples more and more pairs, using a map from vertices to sets of vertices

- Only requires $\widetilde{O}(n)$ space
- Can be updated efficiently at each sample fast while backtracking along the SPs

11. How do we choose the next sample size?

We can compute the next sample size on the fly, using information from $\mathcal{S}$

First iteration: Use a sample of size at least $2\ln(3/\delta)\varepsilon^{-2}$
Why? It is impossible that $\eta_{\mathcal{S}} \leq \varepsilon$ at smaller sample sizes

Successive iterations:
multiply the sample size ℓ from the previous iteration by $(2\eta_{\mathcal{S}}/\varepsilon)^2$

Intuition:
If the upper bound $\min_{s \in \mathbb{R}^+} \rho(s)$ to the Rademacher average $R(\mathcal{S})$ in this iteration is also an upper bound to the Rademacher average for the next iteration, then the stopping condition will be satisfied at the next iteration

12. Is there an upper bound to the number of samples taken by ABRA?

Theorem: Let θ be the *logarithm of the size of the largest WCC in* G. Then we can obtain an (ε, δ)-approximation with a sample $\mathcal{S}$ of size

$$|\mathcal{S}| = \frac{1}{\varepsilon^2} (\theta + \ln(1/\delta))$$

Intuition: θ is an upper bound to the pseudodimension (VC-dimension for real-valued functions)

Conjecture: Let ℓ be the maximum positive integer for which there exists a set $L = \{(u_1, v_1), \ldots, (u_\ell, v_\ell)\}$ of ℓ distinct pairs of distinct vertices such that

$$\sum_{i=1}^{\ell} \sigma_{u_i v_i} \geq \binom{\ell}{\lfloor \ell/2 \rfloor}$$

then the pseudodimension is at most ℓ

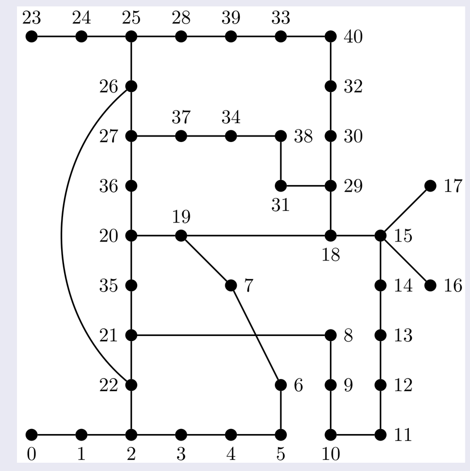


Figure: The conjecture is true for $\ell \leq 4$: this graph satisfies the conjecture for $\ell = 4$ and has pseudodimension 4

13. Experimental evaluation

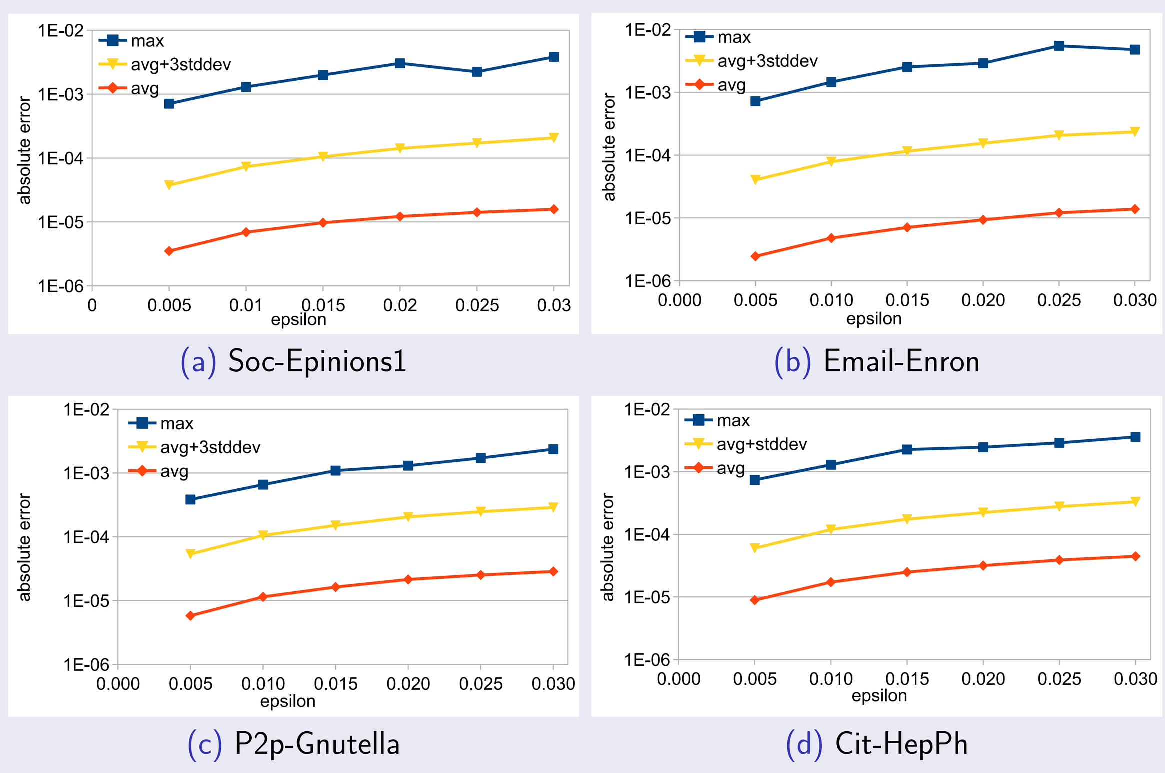
- We implemented ABRA in C++ as extension to NetworKit
- We tested ABRA on datasets from the SNAP repository
- Run on AMD Phenom II X4 955 processor and 16GB of RAM, running FreeBSD 11

Results

- In all runs, the maximum error was smaller than ε
- Average error 100x smaller than ε
- 95th percentile error 10x smaller than ε
- ABRA requires up to 4x fewer samples than state-of-the art RK
- ABRA is up to 7x faster than RK, 40x faster than BA
- 99.7% of the running time is spent doing useful work (sampling)
- 0.1% spent checking the stopping condition
- The automatic sample schedule outperforms fixed geometric ones

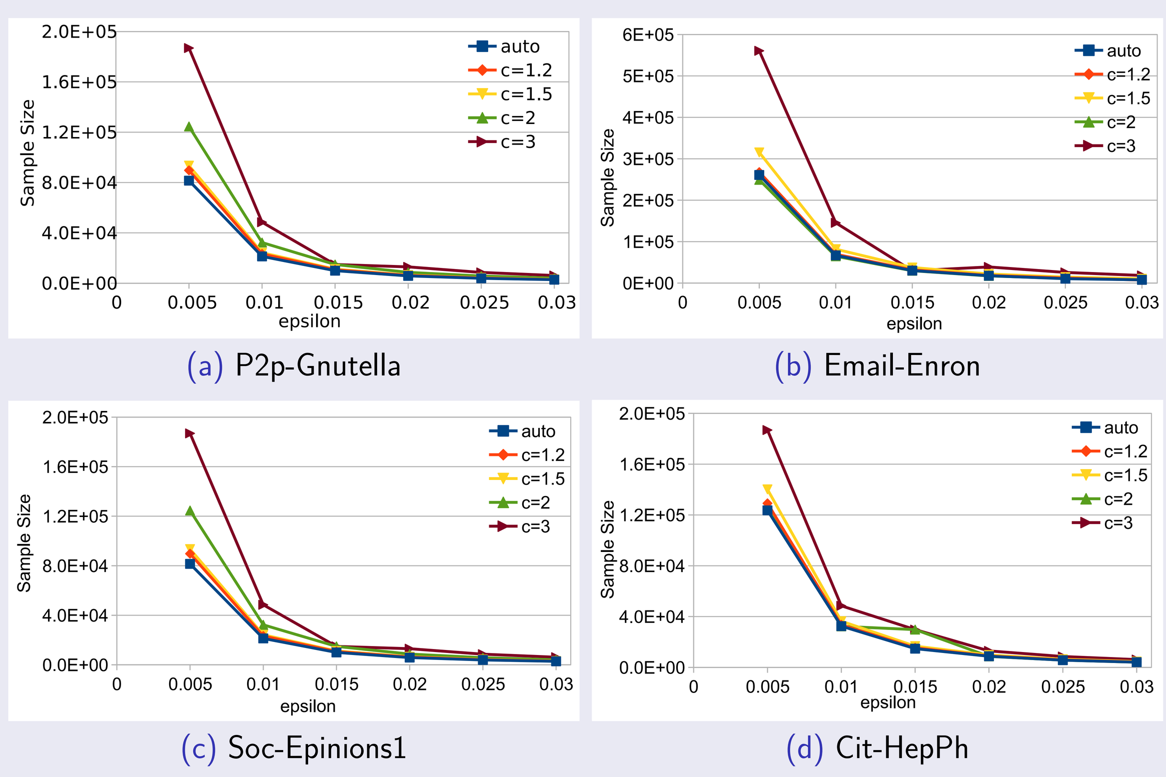
Graph	ε	Runtime (sec.)	Speedup w.r.t. BA	Speedup w.r.t. RK	Runtime Breakdown (%) Sampling	Runtime Breakdown (%) Stop Cond.	Runtime Breakdown (%) Other	Sample Size	Reduction w.r.t. RK	Absolute Error ($\times 10^5$) max	Absolute Error ($\times 10^5$) avg	Absolute Error ($\times 10^5$) stddev
Soc-Epinions1 Directed $\|V\| = 75,879$ $\|E\| = 508,837$	0.005	483.06	1.36	2.90	99.983	0.014	0.002	110,705	2.64	70.84	0.35	1.14
	0.010	124.60	5.28	3.31	99.956	0.035	0.009	28,601	2.55	129.60	0.69	2.22
	0.015	57.16	11.50	4.04	99.927	0.054	0.018	13,114	2.47	198.90	0.97	3.17
	0.020	32.90	19.98	5.07	99.895	0.074	0.031	7,614	2.40	303.86	1.22	4.31
	0.025	21.88	30.05	6.27	99.862	0.092	0.046	5,034	2.32	223.63	1.41	5.24
	0.030	16.05	40.95	7.52	99.827	0.111	0.062	3,668	2.21	382.24	1.58	6.37
P2p-Gnutella31 Directed $\|V\| = 62,586$ $\|E\| = 147,892$	0.005	100.06	1.78	4.27	99.949	0.041	0.010	81,507	4.07	38.43	0.58	1.60
	0.010	26.05	6.85	4.13	99.861	0.103	0.036	21,315	3.90	65.76	1.15	3.13
	0.015	11.91	14.98	4.03	99.772	0.154	0.074	9,975	3.70	109.10	1.63	4.51
	0.020	7.11	25.09	3.87	99.688	0.191	0.121	5,840	3.55	130.33	2.15	6.12
	0.025	4.84	36.85	3.62	99.607	0.220	0.174	3,905	3.40	171.93	2.52	7.43
	0.030	3.41	52.38	3.66	99.495	0.262	0.243	2,810	3.28	236.36	2.86	8.70
Email-Enron Undirected $\|V\| = 36,682$ $\|E\| = 183,831$	0.010	202.43	1.18	1.10	99.984	0.013	0.003	66,882	1.09	145.51	0.48	2.46
	0.015	91.36	2.63	1.09	99.970	0.024	0.006	30,236	1.07	253.06	0.71	3.62
	0.020	53.50	4.48	1.05	99.955	0.035	0.010	17,676	1.03	290.30	0.93	4.83
	0.025	31.99	7.50	1.11	99.932	0.052	0.016	10,589	1.10	548.22	1.21	6.48
	0.030	24.06	9.97	1.03	99.918	0.061	0.021	7,923	1.02	477.32	1.38	7.34
Cit-HepPh Undirected $\|V\| = 34,546$ $\|E\| = 421,578$	0.010	215.98	2.36	2.21	99.966	0.030	0.004	32,469	2.25	129.08	1.72	3.40
	0.015	98.27	5.19	2.16	99.938	0.054	0.008	14,747	2.20	226.18	2.49	5.00
	0.020	58.38	8.74	2.05	99.914	0.073	0.013	8,760	2.08	246.14	3.17	6.39
	0.025	37.79	13.50	2.02	99.891	0.091	0.018	5,672	2.06	289.21	3.89	7.97
	0.030	27.13	18.80	1.95	99.869	0.108	0.023	4,076	1.99	359.45	4.45	9.53

Table: Runtime, speedup, breakdown of runtime, sample size, reduction, and absolute error



(a) Soc-Epinions1 (b) Email-Enron (c) P2p-Gnutella (d) Cit-HepPh

Figure: Absolute error evaluation. The vertical axis has a logarithmic scale



(a) P2p-Gnutella (b) Email-Enron (c) Soc-Epinions1 (d) Cit-HepPh

Figure: Final sample size for different sample schedules

14. Future work

- Extend ABRA to other centrality measures
- Improve bounds to Rademcher averages
- Develop different sampling schemes to use all compute SPs
- Use Martingales theory rather than Union Bound over iterations

15. Acknowledgements

This work was supported, in part, by NSF grant IIS-1247581 and NIH grant R01-CA180776

16. Extended version

Available from http://bit.ly/abra-betweenness