ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages

MATTEO RIONDATO, Two Sigma Investments, LP
ELI UPFAL, Brown University

ΑΒΡΑΞΑΣ (ABRAXAS): Gnostic word of mystic meaning.

We present ABRA, a suite of algorithms to compute and maintain probabilistically-guaranteed high-quality approximations of the betweenness centrality of all nodes (or edges) on both static and fully dynamic graphs. Our algorithms use progressive random sampling and their analysis rely on Rademacher averages and pseudodimension, fundamental concepts from statistical learning theory. To our knowledge, this is the first application of these concepts to the field of graph analysis. Our experimental results show that ABRA is much faster than exact methods, and vastly outperforms, in both runtime number of samples, and accuracy, state-of-the-art algorithms with the same quality guarantees.

CCS Concepts: • **Mathematics of computing → Probabilistic algorithms**; • **Human-centered computing → Social networks**; • **Theory of computation → Shortest paths**; **Dynamic graph algorithms**; **Sketching and sampling**; **Sample complexity and generalization bounds**;

ACM Reference format:
Matteo Riondato and Eli Upfal. 2017. ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. *ACM Trans. Knowl. Discov. Data.* 1, 1, Article 1 (July 2017), 35 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Centrality measures are fundamental concepts in graph analysis: they assign to each node or edge a score that quantifies some notion of the importance of the node/edge in the network [38]. Betweenness Centrality (BC) is a very popular centrality measure that, informally, defines the importance of a node or edge z in the network as proportional to the fraction of shortest paths in the network that go through z [2, 18] (see Sect. 3 for formal definitions).

Brandes [13] presented an algorithm (denoted BA) to compute the exact BC values for all nodes or edges in a graph $G = (V, E)$ in time $O(|V||E|)$ if the graph is unweighted, or time $O(|V||E| + |V|^2 \log |V|)$ if the graph has positive weights. The cost of BA is excessive on modern networks with millions of nodes and tens of millions of edges. Moreover, having the exact BC values may often not be needed, given the exploratory nature of the task. A high-quality approximation of the values is usually sufficient, provided it comes with stringent guarantees.

A preliminary version of this work appeared in the proceedings of ACM KDD'16 as [46].
This work was supported in part by NSF grant IIS-1247581 and NIH grant R01-CA180776, and by funding from Two Sigma Investments, LP.
Authors' addresses: Eli Upfal, Department of Computer Science, Brown University, email: eli@cs.brown.edu; Matteo Riondato, Labs, Two Sigma Investments LP, email: matteo@twosigma.com.

1556-4681/2017/7-ART1
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Today's networks are not only large, but also *dynamic*: edges are added and removed continuously. Keeping the BC values up-to-date after edge insertions and removals is a challenging task, and proposed algorithms [20, 27, 31, 32, 36, 37, 42] may improve the running time for some specific class of input graphs and update models, but in general do not offer worst-case time and space complexities better than from-scratch-recomputation using BA. Maintaining a high-quality approximation up-to-date is more feasible and more *sensible*: there is little informational gain in keeping track of exact BC values that change continuously.

Contributions. We focus on developing algorithms for approximating the BC of all nodes and edges in static and dynamic graphs. Our contributions are the following.

- We present ABRA (for "Approximating Betweenness with Rademacher Averages"), the first family of algorithms based on *progressive sampling* for approximating the BC of all nodes in static and dynamic graphs, where node and edge insertions and deletions are allowed. The BC approximations computed by ABRA are *probabilistically guaranteed* to be within an user-specified additive error ε from their exact values. We also present variants using a fixed amount of samples, with relative error (i.e., within a multiplicative factor ε of the true value) for the top-k nodes with highest BC, and variants that use refined estimators to give better approximations with a slightly larger sample size. Additionally, we also show a *fixed* sampling variant that performs exactly as many sample operations as requested by the user.
- Our analysis relies on Rademacher averages [28, 48] and pseudodimension [41], fundamental concepts from the field of statistical learning theory [50]. Building on known and novel results using these concepts, ABRA computes the approximations without having to keep track of any global property of the graph, in contrast with existing algorithms [7, 9, 44]. A byproduct of our analysis are new general results on pseudodimension (Lemmas 3.7 and 3.8) which show properties that can be used to bound the pseudodimension of *any* problem. ABRA performs only "real work" towards the computation of the approximations, without having to obtain such global properties or update them after modifications of the graph. To the best of our knowledge, ours is the first application of Rademacher averages and pseudodimension to graph analysis problems, and the first to use *progressive* random sampling for BC computation. Using pseudodimension, we derive new analytical results on the sample complexity of the BC computation task, generalizing previous contributions [44], and formulating a conjecture on the connection between pseudodimension and the distribution of shortest path lengths. Our work hence also showcases the usefulness of these highly theoretical concepts developed in the setting of supervised learning to develop practical algorithms for important problems in unsupervised settings.
- The results of our experimental evaluation on real networks show that ABRA outperforms, in both speed, number of samples, and accuracy the state-of-the-art methods offering the same guarantees [44], and it is significantly faster than exact methods [13].

The present paper extends the conference version [46] along multiple directions. The most significant new contributions are the following:

- a revised version of the algorithm with a new proof of correctness, where we fixed a subtle mistake in the algorithm presented in conference version due to the presence of randomly stopped sequences of random variables;
- a new variant of the algorithm using a fixed amount of samples (instead of progressive sampling), which returns much better approximations than previously existing algorithms using a fixed amount of samples [14, 44].

- a stricter bound to the maximum approximation error, which allows ABRA's stopping condition to be satisfied at smaller sample sizes than before;
- a completely new upper bound to the number of samples needed by ABRA to compute an approximation of the desired quality, which allows ABRA to deterministically stop after the number of samples suggested by the upper bound. This upper bound is based on pseudodimension, and we show upper and, in some cases, matching lower bounds to the pseudodimension of the problem of estimating betweenness centralities, shedding new light on its sample complexity. We additionally formulate an open conjecture (see Conjecture 4.11) that we show true for fundamental specific cases, and, if proved true, would greatly reduce the needed number of samples.
- We also reworked our algorithm for relative-error approximation of the top-k highest betweenness values, improving its stopping condition so it will use fewer samples.
- We also present all the proofs of our theoretical results, and additional experimental results, which give insights to the betweenness estimation problem and to the behavior of our algorithms. Moreover, we have added examples throughout the text, with the goal of improving the clarity of the presentation and to make the paper more self-contained.

Outline. We discuss related works in Sect. 2. The formal definitions of the concepts we use in the work can be found in Sect. 3. Our algorithms for approximating BC on static graphs are presented in Sect. 4, while the dynamic case is discussed in Sect. 5. The results of our extensive experimental evaluation are presented in Sect. 6. We draw conclusions in Sect. 7.

2 RELATED WORK

The definition of Betweenness Centrality comes from the sociology literature [2, 18], but the study of efficient algorithms to compute it started only when graphs of substantial size became available to the analysts, following the emergence of the Web. The BA algorithm by Brandes [13] is currently the asymptotically fastest algorithm for computing the exact BC values for all nodes in the network. A number of works also explored heuristics to improve BA [17, 47], but retained the same worst-case time complexity.

The use of random sampling to approximate the BC values in static graphs was proposed independently by Jacob et al. [24] and Brandes and Pich [14], and successive works explored the tradeoff space of sampling-based algorithms [7–9, 44]. Other works focused on estimating the betweenness centrality of a single target node, rather than on obtaining uniform guarantees for all the nodes [5, 25]. We focus here on related works that offer approximation guarantees similar to ours. For an in-depth discussion of previous contributions approximating BC on static graphs but not offering guarantees, we refer the reader to the comments by Riondato and Kornaropoulos [44, Sect. 2]. Table 1 shows a comparison of the sample space, sample size, and analysis techniques for the different works discussed in this section.

Riondato and Kornaropoulos [44] present algorithms that employ the Vapnik-Chervonenkis (VC) dimension [50] to compute what is currently the tightest upper bound on the sample size sufficient to obtain guaranteed approximations of the BC of all nodes in a static graph. Their algorithms offer the same guarantees as ABRA but, to compute the sample size, they need to compute an upper bound on a characteristic quantity of the graph (the vertex-diameter, namely the maximum number of nodes on any shortest path). A progressive sampling algorithm based on the vertex-diameter was recently introduced [10]. Thanks to our use of Rademacher averages in a progressive random sampling setting, ABRA does not need to compute any characteristic quantity of the graph, and instead uses an efficient-to-evaluate stopping condition to determine when the approximated BC

Table 1. Comparison of sample-based algorithms for BC estimation on graphs.

Works	Sample Space	Sample Size for ε-approximation * with confidence $\geq 1-\delta$	Analysis Techniques
[14, 23, 24]	nodes	$O\left(\frac{1}{\varepsilon^2}\left(\ln\lvert V\rvert + \ln\frac{1}{\delta}\right)\right)$	Hoeffding's inequality, Union bound
[8, 44]	shortest paths	$O\left(\frac{1}{\varepsilon^2}\left(\log_2 \mathrm{VD}(G) + \ln\frac{1}{\delta}\right)\right)$ †	VC-Dimension
This work	pairs of nodes	Variable, at most $O\left(\frac{1}{\varepsilon^2}\left(\log_2 \mathsf{L}(G) + \ln\frac{1}{\delta}\right)\right)$ ‡ but usually much less	Rademacher Averages, Pseudodimension

* See Def. 3.2 for the formal definition.
† $\mathrm{VD}(G)$ is the vertex-diameter of the graph G.
‡ $\mathsf{L}(G)$ is the size of the largest weakly connected component of G. See Sect. 4.2 for tighter bounds.

values are close to the exact ones. This allows ABRA to use smaller samples and be much faster than the algorithms by Riondato and Kornaropoulos [44].

A number of works [20, 27, 31, 32, 36, 37, 42] focused on computing the *exact* BC for all nodes in a dynamic graph, taking into consideration different update models. None of these algorithm is provably asymptotically faster than a complete computation from scratch using Brandes' algorithm [13] on general graphs (some of them are faster than BA on some specific classes of input and under some specific update models), and they all require significant amount of space (more details about these works can be found in [7, Sect. 2]). In contrast, Bergamini and Meyerhenke [7, 8] built on the work by Riondato and Kornaropoulos [44] to derive an algorithm for maintaining high-quality approximations of the BC of all nodes when the graph is dynamic and both additions and deletions of edges are allowed. Due to the use of the algorithm by Riondato and Kornaropoulos [44] as a building block, the algorithm must keep track of the vertex-diameter after an update to the graph. Our algorithm for dynamic graphs, instead, does not need this piece of information, and therefore can spend more time in computing the approximations, rather than in keeping track of global properties of the graph. Moreover, our algorithm can handle directed graphs, which is not the case for the algorithms by Bergamini and Meyerhenke [7, 8].

Hayashi et al. [23] recently proposed a data structure called *Hypergraph Sketch* to maintain the shortest path DAGs between pairs of nodes following updates to the graph. Their algorithm uses random sampling and this novel data structure allows them to maintain a high-quality, probabilistically guaranteed approximation of the BC of all nodes in a dynamic graph. Their guarantees come from an application of the simple uniform deviation bounds (i.e., the union bound) to determine the sample size, as previously done by Jacob et al. [24] and Brandes and Pich [14]. As a result, the resulting sample size is excessively large, as it depends on the *number of nodes in the graph*. Our improved analysis using the Rademacher averages allows us to develop an algorithm that uses the Hypergraph Sketch with a much smaller number of samples, and is therefore faster.

Progressive random sampling with Rademacher Averages has been used by Elomaa and Kääriäinen [16] and Riondato and Upfal [45] in completely different settings, i.e., to train classification trees and to mine frequent itemsets respectively.

3 PRELIMINARIES

We now introduce the formal definitions and basic results that we use throughout the paper.

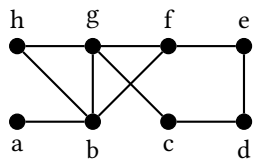
3.1 Graphs and Betweenness Centrality

Let $G = (V, E)$ be a graph. G may be directed or undirected and may have non-negative weights on the edges. For any ordered pair (u, v) of different nodes $u \neq v$, let $\mathcal{S}_{uv}$ be the set of *Shortest Paths* (SPs) from u to v, and let $\sigma_{uv} = |\mathcal{S}_{uv}|$. Given a path p between two nodes $u, v \in V$, a node $w \in V$ is *internal to* p if and only if $w \neq u$, $w \neq u$, and p goes through w. We denote as $\sigma_{uv}(w)$ the number of SPs from u to v that w is internal to.

Definition 3.1 (Betweenness Centrality (BC) [2, 18]). Given a graph $G = (V, E)$, the *Betweenness Centrality (BC) of a node* $w \in V$ is defined as

$$\mathrm{b}(w) = \frac{1}{|V|(|V|-1)} \sum_{\substack{(u,v)\in V\times V \\ u\neq v}} \frac{\sigma_{uv}(w)}{\sigma_{uv}} \quad (\in [0,1]) \ .$$

An example of a graph and the associated values, taken from [44, Sect. 3] is shown in Fig. 1.



(a) Example graph

(b) Betweenness values

Vertex v	a	b	c	d	e	f	g	h
$\mathrm{b}(v)$	0	0.250	0.125	0.036	0.054	0.080	0.268	0

Fig. 1. Example of betweenness values

Many variants of BC have been proposed in the literature, including, e.g., one for edges [38] and one limited to random walks of a fixed length [30]. Our results can be extended to many of these variants, following the same discussion as in [44, Sect. 6].

In this work we focus on computing an *ε-approximation* of the collection $B = \{\mathrm{b}(w), w \in V\}$.

Definition 3.2 (ε-approximation). Given $\varepsilon \in (0, 1)$, an *ε-approximation to B* is a collection

$$\widetilde{B} = \left\{\widetilde{\mathrm{b}}(w), w \in V\right\}$$

such that, for all $w \in V$,

$$\left|\tilde{\mathrm{b}}(w) - \mathrm{b}(w)\right| \leq \varepsilon \ .$$

In Sect. 4.5 we discuss a relative (i.e., multiplicative) error variant for the top-k nodes with highest BC.

3.2 Rademacher Averages

Rademacher Averages [28] are fundamental concepts to study the rate of convergence of a set of sample averages to their expectations. They are at the core of statistical learning theory [50] but their usefulness extends way beyond the learning framework [45]. We present here only the definitions and results that we use in our work and we refer the readers to, e.g., the book by Shalev-Shwartz and Ben-David [48] for in-depth presentation and discussion.

While the Rademacher complexity can be defined on an arbitrary measure space, we restrict our discussion here to a sample space that consists of a finite domain $\mathcal{D}$ and the uniform distribution

over the elements of $\mathcal{D}$. Let $\mathcal{F}$ be a family of functions from $\mathcal{D}$ to $[0,1]$,[1] and let $\mathcal{S} = \{s_1, \ldots, s_\ell\}$ be a collection of ℓ independent uniform samples from $\mathcal{D}$. For each $f \in \mathcal{F}$, define

$$\mathsf{m}_{\mathcal{D}}(f) = \frac{1}{|\mathcal{D}|}\sum_{c\in\mathcal{D}} f(c) \quad (= \mathbb{E}[f]) \quad \text{and} \quad \mathsf{m}_{\mathcal{S}}(f) = \frac{1}{\ell}\sum_{i=1}^{\ell} f(s_i) \quad (\mathbb{E}\left[\mathsf{m}_{\mathcal{S}}(f)\right] = \mathsf{m}_{\mathcal{D}}(f)) \ . \tag{1}$$

Given $\mathcal{S}$, we are interested in bounding the *maximum deviation of* $\mathsf{m}_{\mathcal{S}}(f)$ *from* $\mathsf{m}_{\mathcal{D}}(f)$ *among all* $f \in \mathcal{F}$, i.e., the quantity

$$\sup_{f\in\mathcal{F}} |\mathsf{m}_{\mathcal{S}}(f) - \mathsf{m}_{\mathcal{D}}(f)| \ . \tag{2}$$

For $1 \le i \le \ell$, let λ_i be a Rademacher r.v., i.e., a r.v. that takes value 1 with probability $1/2$ and -1 with probability $1/2$. The r.v.'s λ_i are independent. Consider the quantity

$$\mathsf{R}_{\mathcal{F}}(\mathcal{S}) = \mathbb{E}_{\lambda}\left[\sup_{f\in\mathcal{F}} \frac{1}{\ell}\sum_{i=1}^{\ell}\lambda_i f(s_i)\right], \tag{3}$$

where the expectation is taken only w.r.t. the Rademacher r.v.'s, i.e., conditioning on $\mathcal{S}$. The quantity $\mathsf{R}_{\mathcal{F}}(\mathcal{S})$ is known as the *(conditional) Rademacher average of* $\mathcal{F}$ *on* $\mathcal{S}$.[2]

The connection between $\mathsf{R}_{\mathcal{F}}(\mathcal{S})$ and the maximum deviation (2) is a key result in statistical learning theory. Classically, e.g., in textbooks and surveys, the connection has been presented using suboptimal bounds that are useful for conveying the intuition behind the connection, but inappropriate for practical use (see, e.g., [48, Thm. 26.5], and compare the bounds presented therein with the ones presented in the following.). Tighter although more complex bounds are available [39, 40]. Specifically, we use Thm. 3.3, which is an extension of [39, Thm. 3.11] to a probabilistic tail bound for the supremum of the *absolute value* of the deviation for functions with co-domain $[0,1]$.

Theorem 3.3. *Let* $\mathcal{S}$ *be a collection of* ℓ *independent uniform samples from* $\mathcal{D}$. *Let* $\eta \in (0,1)$. *Then, with probability at least* $1-\eta$,

$$\sup_{f\in\mathcal{F}} |\mathsf{m}_{\mathcal{S}}(f) - \mathsf{m}_{\mathcal{D}}(f)| \le 2\mathsf{R}_{\mathcal{F}}(\mathcal{S}) + \frac{\ln\frac{3}{\eta} + \sqrt{\left(\ln\frac{3}{\eta} + 4\ell\mathsf{R}_{\mathcal{F}}(\mathcal{S})\right)\ln\frac{3}{\eta}}}{2\ell} + \sqrt{\frac{\ln\frac{3}{\eta}}{2\ell}} \ . \tag{4}$$

Even more refined bounds than the ones presented above are available [40] but, as observed by Oneto et al., in practice they do not seem perform better than the one presented in (4).

Computing, or even estimating, the expectation in (3) w.r.t. the Rademacher r.v.'s is not straightforward and can be computationally expensive, requiring a time-consuming Monte Carlo simulation [11]. For this reason, *upper bounds to the Rademacher average* are usually employed in (4) in place of $\mathsf{R}_{\mathcal{F}}(\mathcal{S})$. A powerful and efficient-to-compute bound is presented in Thm. 3.4. Given $\mathcal{S}$, consider, for each $f \in \mathcal{F}$, the vector $\mathbf{v}_{f,\mathcal{S}} = (f(s_1), \ldots, f(s_\ell))$, and let $\mathcal{V}_{\mathcal{S}} = \{\mathbf{v}_{f,\mathcal{S}}, f \in \mathcal{F}\}$ be the *set* of such vectors ($|\mathcal{V}_{\mathcal{S}}| \le |\mathcal{F}|$, as there may be distinct functions of $\mathcal{F}$ with identical vectors).

Theorem 3.4. *Let* $\mathsf{w} : \mathbb{R}^+ \to \mathbb{R}^+$ *be the function*

$$\mathsf{w}(r) = \frac{1}{r}\ln\left(\sum_{\mathbf{v}\in\mathcal{V}_{\mathcal{S}}} \exp\left[\frac{r^2\|\mathbf{v}\|_2^2}{2\ell^2}\right]\right), \tag{5}$$

[1] The fact that the co-domain of the functions in $\mathcal{F}$ is $[0,1]$ is of crucial importance, as many of the results presented in this section are valid *only* for such functions, although they can be extended to general *non-negative* functions.

[2] In this work, we deal, for the most part, with the conditional Rademacher average, rather than with its expectation over the possible samples (which is known as the "Rademacher average", without specializing adjectives). Hence we usually omit the specification "conditional", unless it is needed to avoid confusion.

where $\|\cdot\|_2$ *denotes the* ℓ_2*-norm (Euclidean norm). Then*

$$\mathrm{R}_{\mathcal{F}}(\mathcal{S}) \leq \min_{r \in \mathbb{R}^+} \mathrm{w}(r) \ . \tag{6}$$

This result is obtained from a careful reading of the proof of Massart's Lemma [48, Lemma 26.8].

The function w is convex, continuous in $\mathbb{R}^+$, and has first and second derivatives w.r.t. r everywhere in its domain, so it is possible to minimize it efficiently using standard convex optimization methods [12]. More refined bounds can be derived but are more computationally expensive to compute [1].

3.2.1 Rademacher averages for relative-error approximation. In this section we discuss how to obtain an upper bound the supremum of a specific relative (i.e., multiplicative) deviation of sample means from their expectations, for a family $\mathcal{F}$ of functions from a domain $\mathcal{D}$ to $[0, 1]$.

Let $\mathcal{S} = \{s_1, \ldots, s_\ell\}$ be a collection of ℓ elements from $\mathcal{D}$. Given a parameter $\theta \in (0, 1]$, we are interested specifically in giving probabilistic bounds to the quantity

$$\sup_{f \in \mathcal{F}} \frac{|\mathrm{m}_{\mathcal{D}}(f) - \mathrm{m}_{\mathcal{S}}(f)|}{\max\{\theta, \mathrm{m}_{\mathcal{D}}(f)\}} \ . \tag{7}$$

Li et al. [35] used *pseudodimension* to study the quantity

$$\sup_{f \in \mathcal{F}} \frac{|\mathrm{m}_{\mathcal{D}}(f) - \mathrm{m}_{\mathcal{S}}(f)|}{\mathrm{m}_{\mathcal{D}}(f) + \mathrm{m}_{\mathcal{S}}(f) + \theta} \ . \tag{8}$$

Har-Peled and Sharir [21] derived their concept of relative (θ, ε)-approximation from this quantity and were only concerned with binary functions. The quantity in (8) has been studied often in the literature of statistical learning theory [22], [3, Sect. 5.5], [11, Sect. 5.1], , while other works [4, 6, 15], [11, Sect. 5.1] focused on the quantity

$$\sup_{f \in \mathcal{F}} \frac{|\mathrm{m}_{\mathcal{D}}(f) - \mathrm{m}_{\mathcal{S}}(f)|}{\sqrt{\mathrm{m}_{\mathcal{D}}(f)}} \ .$$

We study the quantity in (7) because it is more useful in our specific case.

It is easy to see that

$$\sup_{f \in \mathcal{F}} \frac{|\mathrm{m}_{\mathcal{D}}(f) - \mathrm{m}_{\mathcal{S}}(f)|}{\max\{\theta, \mathrm{m}_{\mathcal{D}}(f)\}} \leq \sup_{f \in \mathcal{F}} \frac{|\mathrm{m}_{\mathcal{D}}(f) - \mathrm{m}_{\mathcal{S}}(f)|}{\theta} \ . \tag{9}$$

Therefore, a bound to the r.h.s. of this equation implies a bound to the quantity from (7) that we are interested in. We can use Thm. 3.3 to obtain a bound to the supremum of the absolute deviations of the sample means from their expectations for the functions in $\mathcal{F}$, and then divide this bound by θ.

Theorem 3.5. *Let* $\eta \in (0, 1)$*. Let* $\mathcal{S}$ *be a collection of* ℓ *elements of* $\mathcal{D}$ *sampled independently. Then, with probability at least* $1 - \eta$*,*

$$\sup_{f \in \mathcal{F}} \frac{|\mathrm{m}_{\mathcal{D}}(f) - \mathrm{m}_{\mathcal{S}}(f)|}{\max\{\theta, \mathrm{m}_{\mathcal{D}}(f)\}} \leq \frac{1}{\theta}\left(2\mathrm{R}_{\mathcal{F}}(\mathcal{S}) + \frac{\ln(3/\eta) + \sqrt{\ln(3/\eta)\left(4\ell \mathrm{R}_{\mathcal{F}}(\mathcal{S}) + \ln(3/\eta)\right)}}{2\ell} + \sqrt{\frac{\ln(3/\eta)}{2\ell}}\right) .$$

3.3 Pseudodimension

Before introducing the pseudodimension, we must recall some notions and results about the Vapnik-Chervonenkis (VC) dimension. We refer the reader to the books by Shalev-Shwartz and Ben-David [48] and by Anthony and Bartlett [3] for an in-depth exposition of VC-dimension and pseudodimension.

Let D be a domain and let $\mathcal{R}$ be a collection of subsets of D ($\mathcal{R} \subseteq 2^D$). We call $\mathcal{R}$ a *rangeset on* D. Given $A \subseteq D$, the *projection of* $\mathcal{R}$ *on* A is $P_{\mathcal{R}}(A) = \{R \cap A \ : \ R \in \mathcal{R}\}$. When $P_{\mathcal{R}}(A) = 2^A$, we say that A is *shattered* by $\mathcal{R}$. Given $B \subseteq D$, the *empirical VC-dimension* of $\mathcal{R}$, denoted as $\mathsf{EVC}(\mathcal{R}, B)$ is the size of the largest subset of B that can be shattered. The *VC-dimension* of $\mathcal{R}$, denoted as $\mathsf{VC}(\mathcal{R})$ is defined as $\mathsf{VC}(\mathcal{R}) = \mathsf{EVC}(\mathcal{R}, D)$.

Let $\mathcal{F}$ be a class of functions from some domain D to $[0, 1]$. Consider, for each $f \in \mathcal{F}$, the subset R_f of $D \times [0, 1]$ defined as

$$R_f = \{(x, t) \ : \ t \leq f(x)\} \ .$$

We define a rangeset $\mathcal{F}^+$ on $D \times [0, 1]$ as

$$\mathcal{F}^+ = \{R_f, f \in \mathcal{F}\} \ .$$

The *empirical pseudodimension* [41] of $\mathcal{F}$ on a subset $B \subseteq D$, denoted as $\mathsf{EPD}_{\mathcal{F}}(B)$, is the empirical VC-dimension of $\mathcal{F}^+$:

$$\mathsf{EPD}_{\mathcal{F}}(B) = \mathsf{EVC}(\mathcal{F}^+, B) \ .$$

The pseudodimension of $\mathcal{F}$, denoted as $\mathsf{PD}(\mathcal{F})$ is the VC-dimension of $\mathcal{F}^+$ [3, Sect. 11.2]:

$$\mathsf{PD}(\mathcal{F}) = \mathsf{VC}(\mathcal{F}^+) \ .$$

The fundamental result that we use is that having an upper bound on the pseudodimension allows to bound the supremum of the deviations from (2), as stated in the following theorem.

Theorem 3.6 ([35]). *Let D be a domain and $\mathcal{F}$ be a family of functions from D to $[0, 1]$. Let $\mathsf{PD}(\mathcal{F}) \leq d$. Given $\varepsilon, \eta \in (0, 1)$, let $\mathcal{S}$ be a collection of elements sampled independently and uniformly at random from D, with size*

$$|\mathcal{S}| = \frac{c}{\varepsilon^2}\left(d + \log\frac{1}{\eta}\right) \ . \tag{10}$$

Then

$$\Pr\left(\exists f \in \mathcal{F} \text{ s.t. } |\mathsf{m}_D(f) - \mathsf{m}_{\mathcal{S}}(f)| > \varepsilon\right) < \eta \ .$$

The constant c is universal.

The following two technical but completely general lemmas are, to the best of our knowledge, new. They presents constraints on the sets that can actually be shattered by a rangeset, allowing the analyst to focus only on such set to prove bounds on the pseudodimension. We use them later to show upper bounds to the number of samples needed by ABRA.

Lemma 3.7. *Let $B \subseteq D \times [0, 1]$ be a set that is shattered by $\mathcal{F}^+$. Then B can contain* at most one *element $(d, x) \in D \times [0, 1]$ for each $d \in D$.*

Proof. Let $d \in D$ and consider any two distinct values $x_1, x_2 \in [0, 1]$. Let, w.l.o.g., $x_1 < x_2$ and let $B = \{(\tau, x_1), (\tau, x_2)\}$. From the definitions of the ranges, there is no $R \in \mathcal{F}^+$ such that $R \cap B = \{(d, x_1)\}$, therefore B can not be shattered, and so neither can any of its supersets, hence the thesis. □

Lemma 3.8. *Let $B \subseteq D \times [0, 1]$ be a set that is shattered by $\mathcal{F}^+$. Then B does not contain* any *element in the form $(d, 0)$, for any $d \in D$.*

Proof. For any $d \in D$, $(d, 0)$ is contained in every $R \in \mathcal{F}^+$, hence given a set $B = \{(d, 0)\}$ it is impossible to find a range $R_\emptyset$ such that $B \cap R_\emptyset = \emptyset$, therefore B can not be shattered, nor can any of its supersets, hence the thesis. □

4 APPROXIMATING BETWEENNESS CENTRALITY IN STATIC GRAPHS

We now present and analyze ABRA-s, our *progressive sampling algorithm* that computes, with probability at least $1 - \delta$, an ε-approximation to the collection of exact BC values in a *static* graph. Many of the details and properties of ABRA-s are shared with the other ABRA algorithms we present in later sections.

Progressive Sampling. Progressive sampling algorithms are intrinsically *iterative*. At a high level, they work as follows. At iteration i, the algorithm extracts an approximation of the values of interest (in our case, of the BC of all nodes) from a collection $\mathcal{S}_i$ of $S_i = |\mathcal{S}_i|$ random samples from a suitable domain $\mathcal{D}$ (in our case, the samples are pairs of different nodes). Then, the algorithm checks a specific *stopping condition* that uses information obtained from the sample $\mathcal{S}_i$ and from the computed approximation. If the stopping condition is satisfied, then the approximation has, with at least the required probability (in our case $1 - \delta$), the desired quality (in our case, it is an ε-approximation). The approximation is then returned in output and the algorithm terminates. If the stopping condition is not satisfied, the algorithm builds a collection $\mathcal{S}_{i+1}$ by adding random samples to $\mathcal{S}_i$ until it has size S_{i+1}. Then it computes a new approximation from the so-created collection $\mathcal{S}_{i+1}$, and checks the stopping condition again and so on.

There are two main challenges for the designer of progressive sampling algorithm: deriving a "good" stopping condition and determining good choices for the initial sample size S_1 and the subsequent sample sizes S_{i+1}, $i \geq 1$.

An ideal stopping condition is such that:

(1) when satisfied, it guarantees that the computed approximation has the desired quality properties (in our case, the approximation is, with probability at least $1 - \delta$, an ε-approximation); and
(2) it can be evaluated efficiently; and
(3) it is "weak", in the sense that is satisfied at small sample sizes.

The stopping condition for ABRA-s (presented in the following) is based on Thm. 3.3 and Thm. 3.4, and has all the above desirable properties.

The second challenge is determining the *sample schedule* $(S_i)_{i>0}$. Any monotonically increasing sequence of positive numbers can act as sample schedule, but the goal in designing a good sample schedule is to minimize the number of iterations that are needed before the stopping condition is satisfied, while minimizing the sample size S_i at the iteration i at which this happens. The sample schedule is fixed in advance, but an *adaptive approach* allows to find a resonable initial sampling size and then skip directly to a sampling size at which the stopping condition is likely satisfied. We developed such a general adaptive approach which can be used also in other progressive sampling algorithms and is not specific to ABRA (see Sect. 4.1.2.)

4.1 Algorithm Description and Analysis

ABRA-s takes as input a graph $G = (V, E)$, which may be directed or undirected and may have non-negative weights on the edges, a sample schedule $(S_i)_{i \geq 1}$, and two parameters $\varepsilon, \delta \in (0, 1)$. It outputs a collection $\widetilde{B} = \left\{\widetilde{\mathrm{b}}(w), w \in V\right\}$ that is, with probability at least $1 - \delta$, an ε-approximation of the betweenness centralities $B = \{\mathrm{b}(w), w \in V\}$. Let $\mathcal{D} = \{(u, v) \in V \times V, u \neq v\}$. For each node

$w \in V$, let $f_w : \mathcal{D} \to [0, 1]$ be the function

$$f_w(u, v) = \frac{\sigma_{uv}(w)}{\sigma_{uv}}, \tag{11}$$

i.e., $f_w(u, v)$ is the fraction of shortest paths (SPs) from u to v that go through w (i.e., that w is internal to.) Let $\mathcal{F} = \{f_w, w \in V\}$ be the set of these functions. Given this definition, we have that

$$\mathsf{m}_{\mathcal{D}}(f_w) = \frac{1}{|\mathcal{D}|} \sum_{(u,v)\in\mathcal{D}} f_w(u, v) = \frac{1}{|V|(|V|-1)} \sum_{\substack{(u,v)\in V\times V \\ u\neq v}} \frac{\sigma_{uv}(w)}{\sigma_{uv}} = \mathsf{b}(w) \ .$$

The intuition behind ABRA-s is the following. Let $\mathcal{S} = \{(u_i, v_i), 1 \le i \le \ell\}$ be a collection of ℓ pairs (u, v) sampled independently and uniformly from $\mathcal{D}$. For the sake of clarity, we define

$$\widetilde{\mathsf{b}}(w) = \mathsf{m}_{\mathcal{S}}(f_w) = \frac{1}{\ell} \sum_{i=1}^{\ell} f_w(u_i, v_i) = \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{\sigma_{u_i v_i}(w)}{\sigma_{u_i v_i}} \ .$$

For each $w \in V$ consider the vector

$$\mathbf{v}_w = (f_w(u_1, v_1), \ldots, f_w(u_\ell, v_\ell)) \ .$$

It is easy to see that $\widetilde{\mathsf{b}}(w) = \|\mathbf{v}_w\|_1/\ell$. Let now $\mathcal{V}_{\mathcal{S}}$ be the *set* of these vectors:

$$\mathcal{V}_{\mathcal{S}} = \{\mathbf{v}_w, w \in V\} \ .$$

It is possible that $|\mathcal{V}_{\mathcal{S}}| \le |V|$ as there may be two different nodes u and v with $\mathbf{v}_u = \mathbf{v}_v$. This is indeed the usual case in practice. If we have complete knowledge of the set $\mathcal{V}_{\mathcal{S}}$ (i.e., of its elements), then we can compute the quantity

$$\omega^* = \min_{r\in\mathbb{R}^+} \frac{1}{r} \ln\left(\sum_{\mathbf{v}\in\mathcal{V}_{\mathcal{S}}} \exp\left[\frac{r^2 \|\mathbf{v}\|_2^2}{2\ell^2} \right] \right),$$

which, from Thm. 3.4, is an *upper bound* to $\mathsf{R}_{\mathcal{F}}(\mathcal{S})$. We can use ω^* to obtain an upper bound $\xi_{\mathcal{S}}$ to the supremum of the absolute deviation by plugging ω^* in (4). It follows from Thm. 3.3 that the collection $\widetilde{B} = \left\{\widetilde{\mathsf{b}}(w) = \|\mathbf{v}_w\|_1/\ell, w \in V\right\}$ is, with probability at least $1 - \eta$, a $\xi_{\mathcal{S}}$-approximation to the exact betweenness values.

ABRA-s builds on this intuition and works as follows. The algorithm builds a collection $\mathcal{S}$ by sampling pairs (u, v) independently and uniformly at random from $\mathcal{D}$ until $\mathcal{S}$ has size S_1. After sampling a pair (u, v), ABRA-s performs an $s - t$ SP computation from u to v and then backtracks from v to u along the SPs just computed, to keep track of the set $\mathcal{V}_{\mathcal{S}}$ of vectors (details given below). For clarity of presentation, let $\mathcal{S}_1$ denote $\mathcal{S}$ when it has size exactly S_1, and analogously for $\mathcal{S}_i$ and S_i, $i > 1$. Once $\mathcal{S}_i$ has been "built", ABRA-s computes $\xi_{\mathcal{S}_i}$ as described earlier, using $\eta = \delta/(3 \cdot 2^i)$. It then checks whether $\xi_{\mathcal{S}_i} \le \varepsilon$. This is ABRA-s *stopping condition*:[3] when it holds, ABRA-s returns

$$\widetilde{B} = \left\{\widetilde{\mathsf{b}}(w) = \|\mathbf{v}_w\|_1/S_i, w \in V\right\} \ .$$

Otherwise, ABRA-s iterates and continues adding samples from $\mathcal{D}$ to $\mathcal{S}$ until it has size S_2, and so on until $\eta_{\mathcal{S}_i} \le \varepsilon$ holds. The pseudocode for ABRA-s is presented in Alg. 1. including the steps to update $\mathcal{V}_{\mathcal{S}}$, described in the following,

[3] A different stopping condition for generic progressive sampling using Rademacher average was presented by Koltchinskii et al. [29] and used by Elomaa and Kääriäinen [16]. We choose to use ours because it is more efficient to compute.

ALGORITHM 1: ABRA-s: absolute error approximation of BC on static graphs

input : Graph $G = (V, E)$, sample schedule $(S_i)_{i \geq 1}$, accuracy parameter $\varepsilon \in (0, 1)$, confidence parameter $\delta \in (0, 1)$.

output : Pair $(\widetilde{B}, \xi)$ such that $\xi \leq \varepsilon$ and $\widetilde{B}$ is a set of BC approximations for all nodes in V, which is, with probability at least $1 - \delta$, an ξ-approximation to $B = \{\mathrm{b}(w), w \in v\}$.

$\mathcal{D} \leftarrow \{(u, v) \in V \times V, u \neq v\}$
$S_0 \leftarrow 0$,
$\mathbf{0} = (0)$
$\mathcal{V} = \{\mathbf{0}\}$
foreach $w \in V$ **do** $M[w] = \mathbf{0}$
$\mathrm{c}_{\mathbf{0}} \leftarrow |V|$
$i \leftarrow 1, j \leftarrow 1$
while *True* **do**
  for $\ell \leftarrow 1$ **to** $S_i - S_{i-1}$ **do**
    $(u, v) \leftarrow$ `uniform_random_sample(`$\mathcal{D}$`)`
    `compute_SPs(`u, v`)  //Truncated SP computation`
    if *reached* v **then**
      foreach $z \in \mathrm{P}_u[v]$ **do** $\sigma_{zv} \leftarrow 1$
      foreach *node* w *on a SP from* u *to* v*, in reverse order by* $\mathrm{d}(u, w)$ **do**
        $\sigma_{uv}(w) \leftarrow \sigma_{uw}\sigma_{wv}$
        $\mathbf{v} \leftarrow M[w]$
        $\mathbf{v}' \leftarrow \mathbf{v} \cup \{(j, \sigma_{uv}(w)/\sigma_{uv})\}$
        if $\mathbf{v}' \notin \mathcal{V}$ **then**
          $c_{\mathbf{v}'} \leftarrow 1$
          $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{v}'\}$
        else $\mathrm{c}_{\mathbf{v}'} \leftarrow \mathrm{c}_{\mathbf{v}'} + 1$
        $M[w] \leftarrow \mathbf{v}'$
        if $\mathrm{c}_{\mathbf{v}} > 1$ **then** $\mathrm{c}_{\mathbf{v}} \leftarrow \mathrm{c}_{\mathbf{v}} - 1$
        else $\mathcal{V} \leftarrow \mathcal{V} \setminus \{\mathbf{v}\}$
        foreach $z \in \mathrm{P}_u[w]$ **do** $\sigma_{zv} \leftarrow \sigma_{zv} + \sigma_{wv}$
    $j \leftarrow j + 1$
  $\omega_i^* \leftarrow \min_{r \in \mathbb{R}^+} \frac{1}{r} \ln\left(\sum_{\mathbf{v} \in \mathcal{V}} \exp\left[r^2 \|\mathbf{v}\|^2/(2S_i^2)\right]\right)$
  $\gamma_i \leftarrow \ln(3/\delta) + i \ln 2$
  $\xi_{S_i} \leftarrow 2\omega_i^* + \frac{\gamma_i + \sqrt{\gamma_i(\gamma_i + 4S_i\omega_i^*)}}{2S_i} + \sqrt{\frac{\gamma_i}{2S_i}}$
  if $\xi_{S_i} \leq \varepsilon$ **then**
    break
  else
    $i \leftarrow i + 1$
$\widetilde{B} \leftarrow \{\widetilde{\mathrm{b}}(w) \leftarrow \|M[w]\|_1/S_i, w \in V\}$
return $(\widetilde{B}, \xi_{S_i})$

Computing and maintaining the set $\mathcal{V}_\mathcal{S}$. We now discuss in details how ABRA-s efficiently maintain the set $\mathcal{V}_\mathcal{S}$ of vectors, which is used to compute the value $\xi_\mathcal{S}$ and the values $\widetilde{\mathrm{b}}(w) = \|\mathbf{v}_w\|_1/|\mathcal{S}|$ in $\widetilde{B}$. In addition to $\mathcal{V}_\mathcal{S}$, ABRA-s also maintains a map M from V to $\mathcal{V}_\mathcal{S}$ (i.e., $M[w]$ is a vector $\mathbf{v}_w \in \mathcal{V}_\mathcal{S}$), and a counter $\mathrm{c}_\mathbf{v}$ for each $\mathbf{v} \in \mathcal{V}_\mathcal{S}$, denoting how many nodes $w \in V$ have $M[w] = \mathbf{v}$.

At the beginning of the execution of the algorithm, $\mathcal{S} = \emptyset$ and $\mathcal{V}_\mathcal{S} = \emptyset$. Nevertheless, ABRA-s initializes $\mathcal{V}_\mathcal{S}$ to contain one special empty vector $\mathbf{0}$, with no components, and M so that $M[w] = \mathbf{0}$ for all $w \in V$, and $\mathrm{c}_\mathbf{0} = |V|$ (lines 3 and following in Alg. 1).

After having sampled a pair (u, v) from $\mathcal{D}$, ABRA-s updates $\mathcal{V}_\mathcal{S}$, M and the counters as follows. First, it performs (line 11) a $s - t$ SP computation from u to v using any SP algorithm (e.g., BFS, Dijkstra, or even any bidirectional search SP algorithm) modified, as discussed by Brandes [13, Lemma 3], to keep track, for each node w encountered during the computation, of the SP distance $\mathrm{d}(u, w)$ from u to w, of the number σ_{uw} of SPs from u to w, and of the set $\mathrm{P}_u(w)$ of (immediate) predecessors of w along the SPs from u.[4] Once v has been reached (and only if it has been reached), the algorithm starts backtracking from v towards u along the SPs it just computed (line 14). During this backtracking, the algorithm visits the nodes along the SPs in *inverse* order of SP distance from u, ties broken arbitrarily. For each visited node w different from u and v, ABRA-s computes the value $f_w(u, v) = \sigma_{uv}(w)/\sigma_{uv}$ of SPs from u to v that go through w, which is obtained as

$$\sigma_{uv}(w) = \sigma_{uw} \times \sum_{z \,:\, w \in \mathrm{P}_u(z)} \sigma_{zv}$$

where the value σ_{uw} is obtained during the $s - t$ SP computation, and the values σ_{zw} are computed recursively during the backtracking (line 25), as described by Brandes [13]. After computing $\sigma_{uv}(w)$, the algorithm takes the vector $\mathbf{v} \in \mathcal{V}_\mathcal{S}$ such that $M[w] = \mathbf{v}$ and creates a new vector $\mathbf{v}'$ by appending $\sigma_{uv}(w)/\sigma_{uv}$ to the end of $\mathbf{v}$.[5] Then it adds $\mathbf{v}'$ to the *set* $\mathcal{V}_\mathcal{S}$, updates $M[w]$ to $\mathbf{v}'$, and increments the counter $\mathrm{c}_{\mathbf{v}'}$ by one (lines 16 to 22). Finally, the algorithm decrements the counter $\mathrm{c}_\mathbf{v}$ by one, and if $\mathrm{c}_\mathbf{v}$ becomes equal to zero, ABRA-s removes $\mathbf{v}$ from $\mathcal{V}_\mathcal{S}$ (line 24). At this point, the algorithm moves to analyzing another node w' with distance from u less or equal to the distance of w from u. It is easy to see that when the backtracking reaches u, the set $\mathcal{V}_\mathcal{S}$, the map M, and the counters, have been correctly updated. An example of how the data structures evolves from one sample to the other is shown in Fig. 2.

We remark that to compute $\xi_{\mathcal{S}_i}$ and $\widetilde{B}$ and to keep the map M up to date, ABRA-s does not actually need to store the vectors in $\mathcal{V}_\mathcal{S}$ (even in sparse form), but it is sufficient to maintain their ℓ_1 and ℓ_2 (i.e., Euclidean) norms, which require much less space, at the expense of some additional bookkeeping.

4.1.1 *Quality guarantees.*

The following theorem shows the guarantees given by ABRA-s.

Theorem 4.1. *Let r be the index of the last iteration of the algorithm, and let $(\widetilde{B}, \xi_{\mathcal{S}_r})$ be the output. With probability at least $1 - \delta$, $\widetilde{B}$ is an $\xi_{\mathcal{S}_r}$-approximation to the set $B = \{\mathrm{b}(w), w \in V\}$.*

Since $\xi_{\mathcal{S}_r} \leq \varepsilon$, we have that $\widetilde{B}$ is at least an ε-approximation, as required by the user.

We now prove Thm. 4.1.

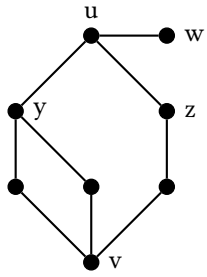
Consider a sequence $(X_j)_{j \geq 1}$ of random variables, where each X_j is a pair of distinct nodes (u, v) sampled independently and uniformly at random from $\mathcal{D} \subset V \times V$. We can reason about the sequence $(X_j)_{j \geq 1}$ *independently from the algorithm.*

Let $(S_i)_{i \geq 1}$ be the sample schedule fixed by the user. Consider the sequences $(X_j)_{j \geq 1}$ and $(S_i)_{i \geq 1}$ and let $(\mathcal{S}_i)_{i \geq 1}$ be the sequence s.t.

$$\mathcal{S}_i = \{X_1, \ldots, X_{S_i}\}, \text{ for all } i \geq 1 \ .$$

[4] Storing the set of immediate predecessors is not necessary. By not storing it, we can reduce the space complexity from $O(|E|)$ to $O(|V|)$, at the expense of some additional computation at runtime.

[5] The pseudocode of ABRA-s uses a sparse representation for the vectors $\mathbf{v} \in \mathcal{V}_\mathcal{S}$, storing only the non-zero components of each $\mathbf{v}$ as pairs (j, g), where j is the component index and g is the value of that component.



(a) A portion of the graph

Vertex x	$M[x]$
⋮	⋮
w	**a**
y	**a**
z	**a**
⋮	⋮

Vector **v**	$c_{\mathbf{v}}$
⋮	⋮
$\mathbf{a} = (a_1, \ldots, a_\ell)$	3
⋮	⋮

(b) Before the update

Vertex x	$M[x]$
⋮	⋮
w	**a**
y	**b**
z	**c**
⋮	⋮

Vector **v**	$c_{\mathbf{v}}$
⋮	⋮
$\mathbf{a} = (a_1, \ldots, a_\ell, 0)$	1
$\mathbf{b} = (a_1, \ldots, a_\ell, 2/3)$	1
$\mathbf{c} = (a_1, \ldots, a_\ell, 1/3)$	1
⋮	⋮

(c) After the update

Fig. 2. Example of the evolution of the data structures. Fig. 2a shows the relevant *portion* of the graph. The algorithm samples the pair (u, v). Figs. 2b and 2c show the data structures M and S, and the counter $c_{\mathbf{v}}$ for the relevant nodes w, x, and y before and after the update, respectively.

Fact 4.2. *For every $i \geq 1$, $\mathcal{S}_i = \{X_1, \ldots, X_{S_i}\}$ is a collection of S_i independent uniform samples from $\mathcal{D}$.*

Given $\eta \in (0, 1)$, let $(\gamma_{\eta,i})_{i \geq 1}$ be the sequence s.t.

$$\gamma_{\eta,i} = \frac{\eta}{2^i}, \text{ for all } i \geq 1 \ .$$

Given any collection $\mathcal{S} = \{(u_1, v_1), \ldots, (u_k, v_k)\}$ of elements from $\mathcal{D}$, let $\mathcal{V}_{\mathcal{S}}$ be the *set* of vectors

$$\mathcal{V}_{\mathcal{S}} = \left\{ \mathbf{v}_w = \left(\frac{\sigma_{u_1 v_i}(w)}{\sigma_{u_1, v_1}}, \ldots, \frac{\sigma_{u_k v_k}(w)}{\sigma_{u_k v_k}} \right), w \in V \right\},$$

and let

$$\omega(\mathcal{S}) = \min_{r \in \mathbb{R}^+} \frac{1}{r} \ln \left(\sum_{\mathbf{v} \in \mathcal{V}_{\mathcal{S}}} \exp\left[r^2 \|\mathbf{v}\|^2 / (2|\mathcal{S}|^2) \right] \right) \ .$$

Ggiven $\lambda \in (0, 1)$, define

$$\xi(\mathcal{S}, \lambda) = 2\omega(\mathcal{S}) + \frac{\lambda + \sqrt{\lambda(\lambda + 4|\mathcal{S}|\omega(\mathcal{S}))}}{2|\mathcal{S}|} + \sqrt{\frac{\lambda}{2|\mathcal{S}|}} \ .$$

Lemma 4.3. *Let $\eta \in (0, 1)$. Then*

$$\Pr\left(\exists i > 0 \text{ s.t. } \sup_{w \in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi(\mathcal{S}_i, \gamma_{\eta,i}) \right) \leq \delta \ . \tag{12}$$

The probability in (12) is taken over all realizations of the sequence $(\mathcal{S}_i)_{i \geq 1}$, i.e., over all realizations of the sequence $(X_j)_{j \geq 1}$.

Proof of Lemma 4.3. From the union bound we have:

$$\Pr\left(\exists i > 0 \text{ s.t. } \sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi(\mathcal{S}_i, \gamma_{\eta,i})\right) \leq \sum_{i=1}^{\infty} \Pr\left(\sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi(\mathcal{S}_i, \gamma_{\eta,i})\right) \ . \tag{13}$$

Since Fact 4.2 holds, we can apply Thm. 3.3 to each $\mathcal{S}_i$. Using the definition of $\gamma_{\eta,i}$ we have, for any fixed i:

$$\Pr\left(\sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi(\mathcal{S}_i, \gamma_{\eta,i})\right) \leq \frac{\eta}{2^i}$$

Continuing from (13) using the above inequality, we then have

$$\begin{aligned}\Pr\left(\exists i > 0 \text{ s.t. } \sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi(\mathcal{S}_i, \gamma_{\eta,i})\right) &\leq \sum_{i=1}^{\infty} \Pr\left(\sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi(\mathcal{S}_i, \gamma_{\eta,i})\right) \\ &\leq \sum_{i=1}^{\infty} \frac{\eta}{2^i} \leq \eta \ . \end{aligned}$$

□

Consider now a realization of the sequence $(X_j)_{j\geq 1}$, and therefore of the sequence $(\mathcal{S}_i)_{i\geq 1}$. Suppose that ABRA-s, instead of actually performing the sampling of pairs of nodes independently and uniformly at random from $\mathcal{D}$, is given the realizations of the sets $\mathcal{S}_i$ in order as follows: at the first iteration it is given $\mathcal{S}_1$, the second time it is given $\mathcal{S}_2 \setminus \mathcal{S}_1$ so that it has seen the whole $\mathcal{S}_2$, and so on.

Lemma 4.4. *Let r denote the last iteration after which this* variant *of* ABRA-s *stops. With probability at least $1-\delta$, the output of* this variant *of* ABRA-s *is an $\xi(\mathcal{S}_r, \gamma_{\delta,r})$-approximation to the set of exact betweenness centralities of all nodes.*

The probability in the lemma is taken over all possible realizations of the sequence $(\mathcal{S}_i)_{i\geq 1}$, i.e., of $(X_j)_{j\geq 1}$.

Proof of Lemma 4.4. Lemma 4.3 says that with probability at least $1-\delta$, the sequence $(\mathcal{S}_i)_{i\geq 1}$ is such that it holds

$$\sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \leq \xi(\mathcal{S}_i, \gamma_{\eta_i}), \text{ for all } i \geq 1 \ . \tag{14}$$

Whether this property holds or not for the realization of $(\mathcal{S}_i)_{i\geq 1}$, i.e., of $(X_j)_{j\geq 1}$, that is "fed" to the algorithm is completely independent from what the algorithm does. Indeed, the realization is fixed before the algorithm even starts.

Suppose (14) holds, which happens with probability at least $1-\delta$. The collection of pairs of nodes that the algorithm has seen is exactly the realization of $\mathcal{S}_r$. The property in (14) holds particularly for $i = r$ so we obtain

$$\sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_r}(w) - \mathrm{b}(w)| \leq \xi(\mathcal{S}_r, \gamma_{\eta_r}) \quad (\leq \varepsilon) \ .$$

□

Fact 4.5. *The distribution of the collection of pairs of nodes sampled by "vanilla" ABRA-s is the same as the distribution of the collection of transactions sampled by the variant of the algorithm described above because, given the same random bits, "vanilla" ABRA-s and the variant generate exactly the same sequence of pairs of nodes.*

By combining Lemma 4.4 and Fact 4.5 we obtain the correctness of the vanilla version of ABRA-s, thus proving Thm. 4.1.

4.1.2 Choosing a sample schedule. We now discuss how to choose a reasonable sample schedule $(S_i)_{i\geq 1}$, so that the algorithm terminates after having performed a small number of iterations at small sample sizes.

Initial sample size. The initial sample size S_1 should be such that

$$S_1 \geq \frac{(1 + 4\varepsilon + \sqrt{1 + 8\varepsilon})\ln(6/\delta)}{4\varepsilon^2} \quad . \tag{15}$$

To understand the intuition behind the lower bound, recall (4), and consider that, at the beginning of the algorithm, there is obviously no information available about $\mathrm{R}_{\mathcal{F}}(\mathcal{S}_1)$, except that it is *non-negative*, i.e., $\mathrm{R}_{\mathcal{F}}(\mathcal{S}_1) \geq 0$. It follows that, for the r.h.s. of (4) to be at most ε at the end of the first iteration (i.e., for the stopping condition to be satisfied at this time), it is necessary that

$$\frac{\ln(6/\delta)}{S_1} + \sqrt{\frac{\ln(6/\delta)}{2S_1}} \leq \varepsilon \quad . \tag{16}$$

Solving for S_1 under the constraints $S_1 \geq 1$, $\delta \in (0, 1)$, $\varepsilon \in (0, 1)$, gives the unique solution in (15).

One may not want to set S_1 equal to the r.h.s. of (15), because its derivation basically assumes that $\mathrm{R}_{\mathcal{F}}(\mathcal{S}_1) = 0$, which is unlikely if not impossible in practice and therefore it is almost guaranteed that, when S_1 equals the r.h.s., the stopping condition would not be satisfied at this sample size. A more reasonable approach would be to multiply the r.h.s. of (15) by some quantity greater than one.

Successive sample sizes. Any increasing sequence can be used as a sample schedule, but there is evidence that a geometrically increasing sample schedule, i.e., a sample schedule such that $S_i = c^i S_1$, for some $c > 1$, may be optimal [43].

4.1.3 Targeting a specific set of nodes. In some situations one may be interested in estimating the BC of only a subset $R \subset V$ of the nodes of the graph. ABRA-s can be easily adapted to this scenario. W.r.t. the pseudocode presented in Alg. 1, the changes are the following:

(1) the map M is initialized only with elements of R (line 5);
(2) $\mathbf{c_0}$ is initialized to $|R|$ (line 6);
(3) the updates to M, $\mathcal{V}$, $\mathbf{v}$, and $\mathbf{c_v}$ (lines 16 to 24 included) are performed only for nodes $w \in R$.

We denote this variant as ABRA-s-set, and we use it in Sect. 4.5.

Restricting to a specific set R of nodes can only have a positive impact on the running time, as the stopping condition may be satisfied earlier than in ABRA-s.

4.2 Upper bounds on the number of samples

It is natural to ask whether, given a graph $G = (V, E)$, there exists an integer s such that ABRA-s can stop and output $(\widetilde{B}, \xi)$ after having sampled s pair of nodes and $\widetilde{B}$ will be, with probability at least $1-\delta$, a ξ-approximation with $\xi \leq \varepsilon$, independently from whether the stopping condition is satisfied or not at that point in the execution. If such a sample size s exists, we can modify the stopping condition of ABRA-s to just stop after having examined a sample of that size, as we describe in Sect. 4.2.1. Such a sample size exists and it is a function of a characteristic quantity of the graph G and of ε and δ. Its derivation uses *pseudodimension* [41], an extension of the Vapnik-Chervonenkis dimension to real-valued functions (see Sect. 3.3).

4.2.1 Using the upper bounds. The upper bounds to the pseudodimension presented in the following subsections can be used in a variant of ABRA-s as follows.

Before starting the sampling process, we compute an upper bound d to the pseudodimension. This requires finding the weakly connected components of the graph G', which takes time $O(|E| + |V|)$. We can then use d to find the minimum $i > 0$ such that the sample size obtained by plugging d and $\eta = \delta/(3 \cdot 2^i)$ into (10) is smaller than the sample size S_i prescribed by the sample schedule. We then modify the sample schedule to use, at iteration i, the value S_{pseudo} obtained from (10) rather than S_i. We also *enrich* the stopping condition of ABRA-s to make the algorithm stop deterministically after having sampled S_{pseudo} pairs of nodes and output $(\widetilde{B}, \varepsilon)$ in this case (the algorithm may still stop earlier if $\xi_{S_j} < \varepsilon$ for any $j < i$.)

4.2.2 General Cases. We now show a general upper bound on the pseudodimension of $\mathcal{F}$. The derivation of this upper bound follows the one for VC-Dimension in [44, Sect. 4], adapted to our settings.

Let $G = (V, E)$ be a graph, and consider the family

$$\mathcal{F} = \{f_w, w \in V\}$$

where f_w goes from $\mathcal{D} = \{(u, v) \in V \times V, u \neq v\}$ to $[0, 1]$ and is defined in (11). The rangeset $\mathcal{F}^+$ contains one range R_w for each node $w \in V$. The set $R_w \subseteq \mathcal{D} \times [0, 1]$ contains pairs in the form $((u, v), x)$, with $(u, v) \in \mathcal{D}$ and $x \in [0, 1]$. The pairs $((u, v), x) \in R_w$ with $x > 0$ are all and only the pairs in this form such that

(1) w is on a SP from u to v; and
(2) $x \leq \sigma_{uv}(w)/\sigma_{uv}$.

For any SP p let $\mathsf{Int}(p)$ be the *set* of nodes that are internal to p, i.e., not including the extremes of p. For any pair (u, v) of distinct nodes, let

$$N_{uv} = \bigcup_{p \in \mathcal{S}_{uv}} \mathsf{Int}(p)$$

be the set of nodes in the SP DAG from u to v, *excluding u and v*, and let $s_{uv} = |N_{uv}|$. Let $\mathsf{H}(G)$ be the maximum integer h such that there are at least $\lfloor \log_2 h \rfloor + 1$ pairs (u, v) such that is $s_{uv} \geq h$. Except in trivial cases, $\mathsf{H}(G) > 0$.

Theorem 4.6. *We have* $\mathsf{PD}(\mathcal{F}) \leq \lfloor \log_2 \mathsf{H}(G) \rfloor + 1$.

Proof. Let $k > \lfloor \log_2 \mathsf{H}(G) \rfloor + 1$ and assume for the sake of contradiction that $\mathsf{PD}(\mathcal{F}) = k$. From the definition of pseudodimension, we have that there is a set Q of k elements of the domain of $\mathcal{F}^+$ that is shattered.

From the definition of $\mathsf{H}(G)$ and from Lemma 3.7, we have that Q must contain an element $a = ((u, v), x)$, $x > 0$, of the domain of $\mathcal{F}^+$ such that $s_{uv} < \mathsf{H}(G)$.

There are 2^{k-1} non-empty subsets of Q containing a. Let us label these non-empty subsets of Q containing a as $S_1, \ldots, S_{2^{k-1}}$, where the labelling is arbitrary. Given that Q is shattered, for each set S_i there must be a range R_i in $\mathcal{F}^+$ such that $S_i = Q \cap R_i$. Since all the S_i's are different from each other, then all the R_i's must be different from each other. Given that a is a member of every S_i, a must also belong to each R_i, that is, there are 2^{k-1} distinct ranges in $\mathcal{F}^+$ containing a. But a belongs only to (not necessarily all) the ranges corresponding to nodes in N_{uv}. This means that a belongs to at most s_{uv} ranges in $\mathcal{F}^+$.

But $s_{uv} < \mathsf{H}(G)$ by definition of $\mathsf{H}(G)$, so p can belong to at most $\mathsf{H}(G)$ ranges from $\mathcal{R}_G$. Given that $2^{k-1} > \mathsf{H}(G)$, we reached a contradiction and there cannot be 2^{k-1} distinct ranges containing a, hence not all the sets S_i can be expressed as $Q \cap R_i$ for some $R_i \in \mathcal{F}^+$.

Then Q cannot be shattered and we have

$$\mathsf{PD}(\mathcal{F}) = \mathsf{VC}(\mathcal{F}^+) \leq \lfloor \log_2 \mathsf{H}(G) \rfloor + 1 \ .$$

□

Computing $\mathsf{H}(G)$ exactly is not practical as it would defeat the purpose of using sampling. Instead, we now present looser but efficient-to-compute upper bounds on the pseudodimension of $\mathcal{F}$ which can be used in practice.

Let $G = (V, E)$ be a graph and let $G' = (V', E')$ be the graph obtained by removing from V some nodes and from E the edges incident to any of the removed nodes. Specifically:

- If G is undirected, we obtain V' by removing all nodes of degree 1 from V.
- If G is directed, we obtain V' by removing all nodes u such that the elements of E involving u are either all in the form (u, v) or are all in the form (v, u).

Consider now the largest (in terms of number of nodes) *Weakly Connected Component* (WCC) of G', and let L be its size (number of nodes in it).

Lemma 4.7. *We have:*

$$\mathsf{PD}(\mathcal{F}) \leq \lfloor \log_2 L \rfloor + 1 \ .$$

Proof. Let's consider *undirected* graphs first. Each WCC of G' is a subset (potentially improper) of one and only one WCC of G. Let W be a WCC of G (W is a set of nodes, $W \subseteq V$) and let W' be the corresponding WCC of G' ($W' \subseteq V'$). Let (u, v) be a pair of nodes in W. It holds $N_{uw} \subseteq W$, i.e., $W \cap N_{uw} = N_{uw}$. We want to show that $N_{uw} \subseteq W'$.

Let v be any node in $W \setminus W'$ (if such a node exists, otherwise it must be $W' = W$ and therefore it must be $N_{uw} \subseteq W'$, since $N_{uw} \subseteq W$). It must be that $v \in V \setminus V'$, i.e., v is one of the removed nodes, which must have had degree 1 in G. The node v is not *internal* to any SP between any two nodes in G, i.e., $v \notin N_{zy}$ for any pair of nodes $(z, y) \in V \times V$, and particularly $v \notin N_{uw}$. This is true for any $v \in W \setminus W'$, hence $(W \setminus W') \cap N_{uw} = \emptyset$. We have:

$$\begin{aligned} W' \cap N_{uw} &= (W \cap N_{uw}) \setminus ((W \setminus W') \cap N_{uw}) \\ &= N_{uw} \setminus \emptyset = N_{uw}, \end{aligned}$$

i.e., $N_{uw} \subseteq W'$. Thus, $|N_{uw}| \leq |W'|$, and therefore $\mathsf{H}(G) \leq L$, from which we obtain the thesis, given Thm. 4.6.

Let's now consider *directed* graphs. It is no longer true that each WCC of G' is a subset (potentially improper) of one and only one WCC of G: there may be multiple WCCs of G' that are subsets of a WCC of G, hence we cannot proceed as in the case of undirected graphs.

Let $\{u, v, w, z\}$ be a set of nodes in V', such that at least three of them are distinct (if two of them are the same, we can assume w.l.o.g. that they are neither u and v nor w and z), and such that there is a path (and hence a SP) in G from u to v and from w to z, and that all these nodes belong to the same WCC of G but to two or more different WCCs of G'. We want to show that no set containing both $((u, v), x)$ and $((w, z), y)$ for some $x, y \in (0, 1)$ could have been shattered by $\mathcal{F}^+$.

Let $S = \{((u, v), x), ((w, z), y)\}$, for u, v, w, z as above. If $\mathcal{F}^+$ cannot shatter S than it cannot shatter any superset of S, so we can focus on S. We assumed that there is a SP from u to v and a SP from w to z in G. Any SP from u to v and from w to z still exists in G', as the removed nodes are not internal to any SPs in G. Hence u and v belong to the same WCC A in G' and w and z belong to the same WCC B in G'. We have, by construction of u, v, w, z that $A \neq B$.

Assume that S is shattered by $\mathcal{F}^+$. Then there must be a node h that is internal to both a SP from u to v and a SP from w to z. If there was not such a node h then S could not be shattered, as there would not be a node ℓ such that the intersection between S and the range R_ℓ associated to ℓ is S. Since h exists and it is internal to two SPs, then it must belong to V'. Since all SPs from u to v and from w to z still exist in G' then so do those that go through h. This means that there is a path from

each of u, v, w, z to the others (e.g., from u to each of v, w, and z), hence they should all belong to the same WCC of G', but this is a contradiction. Hence S cannot be shattered by $\mathcal{F}^+$.

This implies that sets that can be shattered by $\mathcal{F}^+$ are only sets in the form $\{((u_i, v_i), x_i), i = k\}$ such that all nodes u_i and v_i (for all i) belong to the same WCC of G'. Hence, we can proceed as in the undirected graphs case and obtain the thesis. □

The upper bound derived in Lemma 4.7 is somewhat disappointing, and sometimes non-informative: if G is undirected and has a single connected component, then the same bound to the sample size that can be obtained using the pseudodimension could be easily obtained using the union bound. We conjecture that it should be possible to obtain better bounds (see Conjecture 4.11).

4.2.3 Special Cases. In this section we consider some special restricted settings that make computing an high-quality approximation of the BC of all nodes easier. One example of such restricted settings is when the graph is *undirected* and every pair of distinct nodes is either connected with a *single* SP or there is no path between the two nodes (because they belong to different connected components). Examples of these settings are many road networks, where the unique SP condition is often enforced [19]. Riondato and Kornaropoulos [44, Lemma 2] showed that, in this case, the number of samples needed to compute a high-quality approximation of the BC of all nodes is *independent* of any property of the graph, and only depends on the quality controlling parameters ε and δ. The algorithm by Riondato and Kornaropoulos [44] works differently from ABRA-s, as it samples one SP at a time and only updates the BC estimation of nodes along this path, rather than sampling a pair of nodes and updating the estimation of all nodes on any SPs between the sampled nodes. Nevertheless we can actually even generalize the result by Riondato and Kornaropoulos [44], as shown in Thm. 4.8.

THEOREM 4.8. *Let $G = (V, E)$ be a graph such that it is possible to partition the set $\mathcal{D} = \{(u, v) \in V \times V, u \neq v\}$ in two classes: a class $A = \{(u^*, v^*)\}$ containing a single pair of different nodes (u^*, v^*) such that $\sigma_{u^*v^*} \leq 2$ (i.e., connected by either at most two SPs or not connected), and a class $B = \mathcal{D} \setminus A$ of pairs (u, v) of nodes with $\sigma_{uv} \leq 1$ (i.e., either connected by a single SP or not connected). Then the pseudodimension of the family of functions*

$$\{f_w \ : \ \mathcal{D} \to [0, 1], w \in V\},$$

where f_w is defined as in (11)*, is at most* 3.

To prove Thm. 4.8, we show in Lemma 4.9 that some subsets of $\mathcal{D} \times [0, 1]$ can not be shattered by $\mathcal{F}^+$, on any graph G. Thm. 4.8 follows immediately from this result.

LEMMA 4.9. *There exists no undirected graph $G = (V, E)$ such that it is possible to shatter a set*

$$B = \{((u_i, v_i), x_i), 1 \leq i \leq 4\} \subseteq \mathcal{D} \times [0, 1]$$

if there are at least three distinct values $j', j'', j''' \in [1, 4]$ for which

$$\sigma_{u_{j'}v_{j'}} = \sigma_{u_{j''}v_{j''}} = \sigma_{u_{j'''}v_{j'''}} = 1 \ .$$

PROOF. First of all, according to Lemmas 3.7 and 3.8, for B to be shattered it must be

$$(u_i, v_i) \neq (u_j, v_j) \text{ for } i \neq j$$

and $x_i \in (0, 1]$, $1 \leq i \leq 4$.

Riondato and Kornaropoulos [44, Lemma 2] showed that there exists no undirected graph $G = (V, E)$ such that it is possible to shatter B if

$$\sigma_{u_1v_1} = \sigma_{u_2v_2} = \sigma_{u_3v_3} = \sigma_{u_4v_4} = 1 \ .$$

Hence, what we need to show to prove the thesis is that it is impossible to build an undirected graph $G = (V, E)$ such that $\mathcal{F}^+$ can shatter B when the elements of B are such that

$$\sigma_{u_1 v_1} = \sigma_{u_2 v_2} = \sigma_{u_3 v_3} = 1 \quad \text{and} \quad \sigma_{u_4 v_4} = 2 \ .$$

Assume now that such a graph G exists and therefore B is shattered by $\mathcal{F}^+$.

For $1 \leq i \leq 3$, let p_i be the *unique* SP from u_i to v_i, and let p'_4 and p''_4 be the two SPs from u_4 to v_4.

First of all, notice that if any two of p_1, p_2, p_3 meet at a node a and separate at a node b, then they can not meet again at any node before a or after b, as otherwise there would be multiple SPs between their extreme nodes, contradicting the hypothesis. Let this fact be denoted as F_1.

Since B is shattered, its subset

$$A = \{((u_i, v_i), x_i), 1 \leq i \leq 3\} \subset B$$

is also shattered, and in particular it can be shattered by a collection of ranges that is a subset of a collection of ranges that shatters B. We now show some facts about the properties of this shattering which we will use later in the proof.

Define

$$i^+ = \begin{cases} i+1 & \text{if } i = 1, 2 \\ 1 & \text{if } i = 3 \end{cases}$$

and

$$i^- = \begin{cases} 3 & \text{if } i = 1 \\ i-1 & \text{if } i = 2, 3 \end{cases} \ .$$

Let w_A be a node such that $R_{w_A} \cap A = A$. For any set $L = \{k_1, k_2, \dots\} \subseteq \{1, 2, 3, 4\}$ of indices, let $w_L = w_{k_1, k_2, \dots}$ be the node such that

$$R_L \cap A = \{((u_{k_\ell}, v_{k_\ell}), x_{k_\ell}), k_\ell \in L\} \ .$$

For example, for $i \in \{1, 2, 3\}$, w_{i,i^+} is the node such that

$$R_{w_{i,i^+}} \cap A = \{((u_i, v_i), x_i), ((u_{i^+}, v_{i^+}), x_{i^+})\} \ .$$

Analogously, w_{i,i^-} is the node such that

$$R_{w_{i,i^-}} \cap A = \{((u_i, v_i), x_i), ((u_{i^-}, v_{i^-}), x_{i^-})\} \ .$$

We want to show that w_A is on the SP connecting w_{i,i^+} to w_{i,i^-} (such a SP must exist because the graph is undirected and w_{i,i^+} and w_{i,i^-} must be on the same connected component, as otherwise they could not be use to shatter A.) Assume w_A was not on the SP connecting w_{i,i^+} to w_{i,i^-}. Then we would have that either w_{i,i^+} is "between" w_A and w_{i,i^-} ((i.e., along the SP connecting these nodes) or w_{i,i^-} is between w_A and w_{i,i^+}. Assume it was the former (the latter follows by symmetry). Then

(1) there must be a SP p' from u_{i^-} to v_{i^+} that goes through w_{i,i^-};
(2) there must be a SP p'' from u_{i^-} to v_{i^+} that goes through w_A;
(3) there is no SP from u_{i^-} to v_{i^+} that goes through w_{i,i^+}.

Since there is only one SP from u_{i^-} to v_{i^-}, it must be that $p' = p''$. But then p' is a SP that goes through $w_{i,i-}$ and through w_A but not through w_{i,i^+}, and p_i is a SP that goes through w_{i,i^-}, through w_{i,i^+}, and through w_A (either in this order or in the opposite). This means that there are at least two SPs between w_{i,i^-} and w_A, and therefore there would be two SPs between u_i and v_i, contradicting the hypothesis that there is only one SP between these nodes. Hence it must be that w_A is between w_{i,i^-} and w_{i,i^+}. This is true for all i, $1 \leq i \leq 3$. Denote this fact as F_2.

Consider now the nodes $w_{i,4}$ and $w_{j,4}$, for $i, j \in \{1, 2, 3\}$, $i \neq j$. We now show that they can not belong to the same SP from u_4 and v_4.

- Assume that $w_{i,4}$ and $w_{j,4}$ are on the same SP p from u_4 to v_4 and assume that $w_{i,j,4}$ is also on p. Consider the possible orderings of $w_{i,4}$, $w_{j,4}$ and $w_{i,j,4}$ along p.
  - If the ordering is $w_{i,4}$, then $w_{j,4}$, then $w_{i,j,4}$ or $w_{j,4}$, then $w_{j,4}$, then $w_{i,j,4}$, or the reverses of these orderings (for a total of four orderings), then it is easy to see that fact F_1 would be contradicted, as there are two different SPs from the first of these nodes to the last, one that goes through the middle one, and one that does not, but then there would be two SPs between the pair of nodes (u_k, v_k) where k is the index in $\{1, 2, 3\}$ different than 4 that is in common between the first and the last nodes in this ordering, and this would contradict the hypothesis, so these orderings are not possible.
  - Assume instead the ordering is such that $w_{i,j,4}$ is between $w_{i,4}$ and $w_{j,4}$ (two such ordering exist). Consider the paths p_i and p_j. They must meet at some node $w_{f_{i,j}}$ and separate at some node $w_{l_{i,j}}$. From the ordering, and fact F_1, $w_{i,j,4}$ must be between these two nodes. From fact F_2 we have that also w_A must be between these two nodes. Moreover, neither $w_{i,4}$ nor $w_{j,4}$ can be between these two nodes. But then consider the SP p. This path must go together with p_i (resp. p_j) from at least $p_{i,4}$ (resp. $p_{j,4}$) to the farthest between $w_{f_{i,j}}$ and $w_{l_{i,j}}$ from $p_{i,4}$ (resp. $p_{j,4}$). Then in particular p goes through all nodes between $w_{f_{i,j}}$ and $w_{l_{i,j}}$ that p_i and p_j go through. But since w_A is among these nodes,and w_A can not belong to p, this is impossible, so these orderings of the nodes $w_{i,4}$, $w_{j,4}$, and $w_{i,j,4}$ are not possible.

  Hence we showed that $w_{i,4}$, $w_{j,4}$, and $w_{i,j,4}$ can not be on the same SP from u_4 to v_4.
- Assume now that $w_{i,4}$ and $w_{j,4}$ are on the same SP from u_4 to v_4 but $w_{i,j,4}$ is on the other SP from u_4 to v_4 (by hypothesis there are only two SPs from u_4 to v_4). Since what we show in the previous point must be true for all choices of i and j, we have that all nodes $w_{h,4}$, $1 \le h \le 3$, must be on the same SP from u_4 to v_4, and all nodes in the form $w_{i,j,4}$, $1 \le i < j \le 3$ must be on the other SP from u_4 to v_4. Consider now these three nodes, $w_{1,2,4}$, $w_{1,3,4}$, and $w_{2,3,4}$ and consider their ordering along the SP from u_4 to v_4 that they lay on. No matter what the ordering is, there is an index $h \in \{1, 2, 3\}$ such that the SP p_h must go through the extreme two nodes in the ordering but not through the middle one. But this would contradict fact F_1, so it is impossible that we have $w_{i,4}$ and $w_{j,4}$ on the same SP from u_4 to v_4 but $w_{i,j,4}$ is on the other SP, for any choice of i and j.

We showed that the nodes $w_{i,4}$ and $w_{j,4}$ can not be on the same SP from u_4 to v_4. But this is true for any choice of the unordered pair (i, j) and there are three such choices, but only two SPs from u_4 to v_4, so it is impossible to accommodate all the constraints requiring $w_{i,4}$ and $w_{j,4}$ to be on different SPs from u_4 to v_4. Hence we reach a contradiction and B can not be shattered. □

The bound in Thm. 4.8 is tight, i.e., there exists a graph for which the pseudodimension is exactly 3 [44, Lemma 4]. Moreover, as soon as we relax the requirement in Thm. 4.8 and allow two pairs of nodes to be connected by two SPs, there are graphs with pseudodimension 4, as shown in the following Lemma.

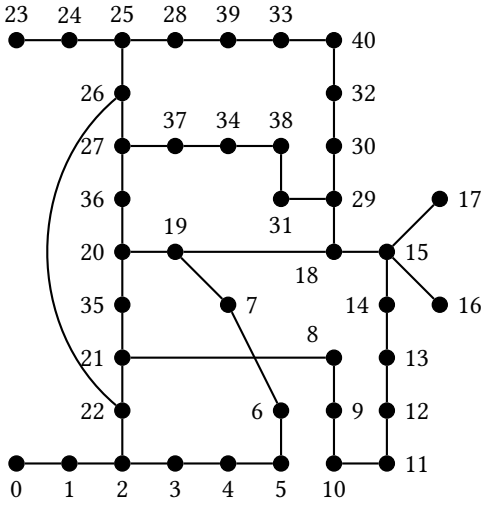
Lemma 4.10. *There is an undirected graph $G = (V, E)$ such that there is a set $\{(u_i, v_i), u_i, v_i \in V, u_i \neq v_i, 1 \le i \le 4\}$ with $|\mathrm{S}_{u_1,v_1}| = |\mathrm{S}_{u_2,v_2}| = 2$ and $|\mathrm{S}_{u_3,v_3}| = |\mathrm{S}_{u_4,v_4}| = 1$ that is shattered.*

Proof. Consider the undirected graph $G = (V, E)$ in Fig. 3a. There is a single SP from 0 to 16:

$$0, 1, 2, 22, 21, 35, 20, 19, 18, 15, 16 \ .$$

There is a single SP from 23 to 17:

$$23, 24, 25, 26, 27, 36, 20, 19, 18, 15, 17 \ .$$



(a) Graph for Lemma 4.10

(b) How to shatter $Q = \{a, b, c, d\}$ from Lemma 4.10.

$P \subseteq Q$	Node v s.t. $P = Q \cap R_v$
$\emptyset$	0
$\{a\}$	1
$\{b\}$	24
$\{c\}$	40
$\{d\}$	38
$\{a, b\}$	20
$\{a, c\}$	2
$\{a, d\}$	21
$\{b, c\}$	25
$\{b, d\}$	27
$\{c, d\}$	29
$\{a, b, c\}$	19
$\{a, b, d\}$	15
$\{a, c, d\}$	22
$\{b, c, d\}$	26
$\{a, b, c, d\}$	18

Fig. 3. Supporting figures and table for Lemma 4.10.

There are exactly two SPs from 5 to 33:

$$5, 4, 3, 2, 22, 26, 25, 28, 39, 33 \quad \text{and} \quad 5, 6, 7, 19, 18, 29, 30, 32, 40, 33 \ .$$

There are exactly two SPs from 11 to 34:

$$11, 10, 9, 8, 21, 22, 26, 27, 37, 34 \quad \text{and} \quad 11, 12, 13, 14, 15, 18, 29, 31, 38, 34 \ .$$

Let $a = ((0, 16), 1)$, $b = ((23, 17), 1)$, $c = ((5, 33), 1/2)$, and $d = ((11, 34), 1/2)$. We can shatter the set $Q = \{a, b, c, d\}$, as shown in Fig. 3b. □

For the case of *directed* networks, it is currently an open question whether a high-quality (i.e., within ε) approximation of the BC of all nodes can be computed from a sample whose size is independent of properties of the graph, but it is known that, even if possible, the constant would not be the same as for the undirected case [44, Sect. 4.1].

We conjecture that, given some information on how many pair of nodes are connected by x shortest paths, for $x \geq 0$, it should be possible to derive a strict bound on the pseudodimension associated to the graph. Formally, we pose the following conjecture, which would allow us to generalize Lemma 4.9, and develop an additional stopping rule for ABRA-s based on the empirical pseudodimension.

CONJECTURE 4.11. *Let $G = (V, E)$ be a graph and let ℓ be the maximum positive integer for which there exists a set $L = \{(u_1, v_i), \ldots, (u_\ell, v_\ell)\}$ of ℓ distinct pairs of distinct nodes such that*

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

then $\mathrm{PD}(\mathcal{F}) \leq \ell$.

The conjecture is tight in the sense that, e.g., for the graph in Fig. 3a, we have that $\ell = 4$ and the pseudodimension is exactly ℓ.

4.3 Alternative Estimators

Geisberger et al. [19] present an alternative estimator for BC using random sampling. Their experimental results show that the quality of the approximation is significantly improved, but they do not present any theoretical analysis. Their algorithm, which follows the work of Brandes and Pich [14], differs from ours as it samples nodes and performs a Single-Source-Shortest-Paths (SSSP) computation from each of the sampled nodes. We can use an adaptation of their estimator in a variant of ABRA-s, and we can prove that this variant is still computes, with probability at least $1 - \delta$ a ρ-approximation of the BC of all nodes with $\rho \leq \varepsilon$, therefore removing the main limitation of the original work, which offered no quality guarantees. We now present this variant considering, for ease of discussion, the special case of the linear scaling estimator by Geisberger et al. [19]. This technique can be extended to the generic parameterized estimators they present.

The intuition behind the alternative estimator is to increase the estimation of the BC for a node w proportionally to the ratio between the SP distance $\mathrm{d}(u, w)$ from the first component u of the pair (u, v) to w and the SP distance $\mathrm{d}(u, v)$ from u to v. Rather than sampling pairs of nodes, the algorithm samples triples (u, v, d), where d is a *direction*, (either $\leftarrow$ or $\rightarrow$), and updates the betweenness estimation differently depending on d, as follows. Let $\mathcal{D}' = \mathcal{D} \times \{\leftarrow, \rightarrow\}$ and for each $w \in V$, define the function g_w from $\mathcal{D}'$ to $[0, 1]$ as:

$$g_w(u, v, d) = \begin{cases} \frac{\sigma_{uv}(w)}{\sigma_{uv}} \frac{\mathrm{d}(u,w)}{\mathrm{d}(u,v)} & \text{if } d = \rightarrow \\ \frac{\sigma_{uv}(w)}{\sigma_{uv}} \left(1 - \frac{\mathrm{d}(u,w)}{\mathrm{d}(u,v)}\right) & \text{if } d = \leftarrow \end{cases}$$

Let $\mathcal{S}$ be a collection of ℓ elements of $\mathcal{D}'$ sampled uniformly and independently at random with replacement. The alternative estimator $\widetilde{\mathrm{b}}(w)$ of the BC of a node w is

$$\widetilde{\mathrm{b}}(w) = \frac{2}{\ell} \sum_{(u,v,d) \in \mathcal{S}} g_w(u, v, d) = 2\mathrm{m}_{\mathcal{S}}(g_w) = \mathrm{m}_{\mathcal{S}}(2g_w) \ .$$

The presence of the factor 2 in the estimator calls for two minor adjustments in this variant of the algorithm w.r.t. the "vanilla" ABRA-s, since, in a nutshell, we now want to estimate the expectations of functions with co-domain in $[0, 2]$:

- the update to the vector $\mathbf{v}$ to obtain $\mathbf{v}'$ on line 17 of Alg. 1 becomes

$$\mathbf{v}' \leftarrow \mathbf{v} \cup \{(j, g_w(u, v, d))\} \ ;$$

- the definition of $\xi_{\mathcal{S}_i}$ on line 29 becomes

$$\xi_{\mathcal{S}_i} = 2\omega_i^* + \frac{2\gamma_i + \sqrt{2\gamma_i(2\gamma_i + 4\ell\omega_i^*)}}{2\ell} + 2\sqrt{\frac{\gamma_i}{2\ell}} \ .$$

These changes ensure that the output of this variant of ABRA-s is still a high-quality approximation of the BC of all nodes, i.e., that Thm. 4.1 still holds. This is due to the fact that the results on the Rademacher averages presented in Sect. 3.2 can be extended to families of functions whose co-domain is an interval $[0, b]$ (in this case, $b = 2$). Other details such the starting sample size and exact expression to compute the next sample size, described in a previous section, or the relative-error variant described in the he ollowing section, can also be adapted to this case.

It is important to mention that, despite having solved the main drawback of the work by Geisberger et al. [19], i.e., its lack of guarantees, the solution is not entirely satisfactory: the presence of the 2 in the estimator results in larger stopping sample sizes than the "vanilla" ABRA-s. This

drawback is due to the fact that the size of the co-domain of the functions, which in this case is 2, is used in the proof of Thm. 3.3 in place of the variance, which is suboptimal. This sub-optimality is evident in this case: the alternative estimator is supposed to have a lower variance than the vanilla one, but technical limitations in the proof do not allow us to exploit this fact. Removing this limitation is an interesting direction for future work.

4.4 Fixed Sample Size

In this section we introduce a variant ABRA-s-fix of ABRA-s that uses a *fixed* sample size, rather than using progressive sampling.

Instead of specifying ε and δ as part of the input, the user specifies δ and a positive integer value S, representing the number of samples (i.e., random pairs of nodes) that the algorithm will take. The algorithm will *always* perform S (and only S) iterations of the loop on lines 9–26 in Alg. 1, then computes ω^*, γ, and ξ as in lines 27–29 and it will output, *no matter the value of* ξ, the pair $(\widetilde{B}, \xi)$.

Theorem 4.12. *With probability at least* $1-\delta$*, the set* $\widetilde{B}$ *is a* ξ*-approximation to the set of exact betweenness centralities for all nodes.*

The proof is immediate from Thm. 3.3 and the definition of ξ.

An advantage of this algorithm with respect to other fixed sampling algorithms such as the one by Riondato and Kornaropoulos [44], is that it can compute the quality of the approximation directly from the sample, without having to compute characteristic quantities from the graph. Additionally, the parameter M is more interpretable, for an end user, than the parameter ε. On the other hand, the quality of the approximation that will be obtained is not known (or even set by the user) in advance before running the algorithm.

4.5 Relative-error Top-k Approximation

In practical applications it is usually sufficient to identify the nodes with highest BC, as they act, in some sense, as the "primary information gateways" of the network. In this section we present a variant ABRA-s-k of ABRA-s to compute a high-quality approximation of the set $\mathrm{TOP}(k, G)$ of the top-k nodes with highest BC in a graph G.

The approximation $\widetilde{\mathrm{b}}(w)$ returned by ABRA-s-k for a node w is within a *multiplicative* factor $\rho \leq \varepsilon$ from its exact value $\mathrm{b}(w)$, rather than an additive factor $\xi \leq \varepsilon$ as probabilistically guaranteed by ABRA-s (see Thm. 4.14 for ABRA-s-k's guarantees). Achieving such higher accuracy guarantees has a cost in terms of the number of samples needed to compute the approximations.

Formally, assume to order the nodes in the graph in decreasing order by BC, ties broken arbitrarily, and let b_k be the BC of the k^{th} node in this ordering. The set $\mathrm{TOP}(k, G)$ of the top-k nodes with highest betweenness in G is defined as the set of nodes with BC at least b_k, and can contain more than k nodes:

$$\mathrm{TOP}(k, G) = \{(w, \mathrm{b}(w)) \ : \ v \in V \text{ and } \mathrm{b}(w) \geq b_k\} \quad .$$

The algorithm ABRA-s-k follows an approach similar to the one taken by the algorithm for the same task by Riondato and Kornaropoulos [44, Sect. 5.2] and works in two phases. The pseudocode for ABRA-s-k is presented in Alg. 2 and we now describe how the algorithm works.

Let δ' and δ'' be such that $(1-\delta')(1-\delta'') = 1-\delta$. In the first phase, ABRA-s is run with input G, ε, δ', and $(S_i)_{i\geq 1}$, and it returns the pair $\left(\widetilde{B}', \xi\right)$.

Let now $\tilde{b}'_k$ be the k^{th} highest value $\widetilde{\mathrm{b}}'(w)$ in $\widetilde{B}'$, ties broken arbitrarily, and let $y' = \tilde{b}'_k - \xi$. Also let C be the set

$$C = \left\{v \in V \ : \ \widetilde{\mathrm{b}}'(v) \geq \tilde{b}'_k - 2\xi\right\} \quad .$$

ALGORITHM 2: ABRA-s-k: relative-error approximation of top-k BC nodes on static graph

input : Graph $G = (V, E)$, accuracy parameter $\varepsilon \in (0,1)$, confidence parameter $\delta \in (0,1)$, value $k \geq 1$, sample schedule $(S_i)_{i \geq 1}$

output: Pair $\left(\widetilde{B}, \rho\right)$, where $\rho \leq \varepsilon$ and $\widetilde{B}$ is a set of approximations of the BC of the top-k nodes in V with highest BC

$\delta', \delta'' \leftarrow$ reals such that $(1-\delta')(1-\delta'') = 1-\delta$

$\left(\widetilde{B}', \xi\right) \leftarrow$ output of ABRA-s with input $G, \varepsilon, \delta', (S_i)_{i \geq 1}$

$\tilde{b}'_k \leftarrow k^{\text{th}}$ highest value $\widetilde{\mathrm{b}}'(w)$ in $\widetilde{B}'$

$y' \leftarrow \tilde{b}'_k - \xi$

$C \leftarrow \left\{v \in V \ : \ \widetilde{\mathrm{b}}(v) \geq \tilde{b}'_k - 2\xi\right\}$

$\left(\widetilde{B}'', \rho\right) \leftarrow$ output of ABRA-s-set-r with input $G, \varepsilon, \delta'', C, y', (S_i)_{i \geq 1}$

$\tilde{b}''_k \leftarrow k^{\text{th}}$ highest value $\widetilde{\mathrm{b}}''(w)$ in $\widetilde{B}''$

$y'' \leftarrow \frac{\tilde{b}''_k}{1+\rho}$

$z \leftarrow \max\{y', y''\}$

$\widetilde{\mathrm{TOP}}(k, G) = \left\{\left(w, \widetilde{\mathrm{b}}''(w)\right) \ : \ w \in C \text{ and } \widetilde{\mathrm{b}}''(w) \geq z(1-\rho)\right\}$

return $\left(\widetilde{\mathrm{TOP}}(k, G), \rho\right)$

Before describing the second phase of ABRA-s-k, we introduce a variant ABRA-s-set-r of ABRA-s-set (see Sect. 4.1.3). ABRA-s-set-r has a modified stopping condition based on Thm. 3.5, which takes an additional parameter $\theta \in (0,1)$. The parameter θ plays a role in the stopping condition of ABRA-s-set-r. Indeed, ABRA-s-set-r is the same as ABRA-s-set, with the only crucial difference in the definition of the quantity $\xi_{\mathcal{S}_i}$, which becomes:

$$\xi_{\mathcal{S}_i} = \frac{1}{\theta}\left(2\omega_i^* + \frac{\ln(3/\eta) + \sqrt{(\ln(3/\eta) + 4\ell\omega_i^*)\ln(3/\eta)}}{2\ell} + \sqrt{\frac{\ln(3/\eta)}{2\ell}}\right) \ . \tag{17}$$

In the second phase, ABRA-s-k runs ABRA-s-set-r with input G, ε, δ'', C as the specific set of interest, $\theta = y'$, and $(S_i)_{i \geq 1}$.[6] It returns a pair $(\widetilde{B}'', \rho)$, with $\rho \leq \varepsilon$.

Let $\tilde{b}''_k$ be the k^{th} highest value $\widetilde{\mathrm{b}}''(w)$ in $\widetilde{B}''$, ties broken arbitrarily, and let $y'' = \tilde{b}''_k/(1+\rho)$.

ABRA-s-k first computes $z = \max\{y', y''\}$, and then computes the set

$$\widetilde{\mathrm{TOP}}(k, G) = \left\{\left(w, \widetilde{\mathrm{b}}(w)\right) \ : \ w \in C \text{ and } \widetilde{\mathrm{b}}(w) \geq z(1-\rho)\right\},$$

and returns $\left(\widetilde{\mathrm{TOP}}(k, G), \rho\right)$.

Guarantees. We now discuss the quality guarantees of the output of ABRA-s-k.

We first state a result on the output of ABRA-s-set-r.

THEOREM 4.13. *Let ε, δ, and θ in $(0,1)$ and let $Z \subset V$. Let $\left(\widetilde{B} = \left\{\widetilde{\mathrm{b}}(w), w \in Z\right\}, \rho\right)$ be the output of ABRA-s-set-r. With probability at least $1-\delta$ it holds that $\rho \leq \varepsilon$ and*

$$\frac{\left|\widetilde{\mathrm{b}}(w) - \mathrm{b}(w)\right|}{\max\{\theta, \mathrm{b}(w)\}} < \rho, \ \textit{for all } w \in Z \ .$$

[6] The sample schedules may be different between the first and second phases, but we do not discuss this case for ease of presentation.

The proof follows the same steps as the proof for Thm. 4.1, using the definition of ξ_{S_i} from (17) and applying Thm. 3.5 instead of Thm. 3.3.

We have the following result showing the properties of the collection $\widetilde{\mathrm{TOP}}(k, G)$.

Theorem 4.14. *With probability at least* $1 - \delta$*, the pair* $\left(\widetilde{\mathrm{TOP}}(k, G), \rho\right)$ *returned by ABRA-s-k is such that* $\rho \leq \varepsilon$ *and:*

(1) *for any pair* $(v, \mathrm{b}(v)) \in \mathrm{TOP}(k, G)$*, there is a pair* $\left(v, \widetilde{\mathrm{b}}(v)\right) \in \widetilde{\mathrm{TOP}}(k, G)$ *and* $\widetilde{\mathrm{b}}(v)$ *is such that* $\left|\widetilde{\mathrm{b}}(v) - \mathrm{b}(v)\right| \leq \rho \mathrm{b}(v)$*;*

(2) *for any pair* $\left(w, \widetilde{\mathrm{b}}(w)\right) \in \widetilde{\mathrm{TOP}}(k, G)$ *such that* $(w, \mathrm{b}(w)) \notin \mathrm{TOP}(k, G)$*, it holds that* $\widetilde{\mathrm{b}}(w) \leq (1 + \rho) b_k$.

Proof. With probability at least $1 - \delta'$, we have, from the properties of ABRA-s, that $\left(\widetilde{B}', \xi\right)$ are such that

$$\left|\widetilde{\mathrm{b}}(w) - \mathrm{b}(w)\right| \leq \xi, \text{ for all } w \in V \ . \tag{18}$$

With probability at least $1 - \delta''$, we have, from the properties of ABRA-s-set-r in Thm. 4.13 that

$$\frac{\left|\widetilde{\mathrm{b}}(w) - \mathrm{b}(w)\right|}{\max\{y', \mathrm{b}(w)\}} \leq \rho, \text{ for all } w \in C \ . \tag{19}$$

Suppose both these events occur, which happens with probability at least $1 - \delta$.

Consider the value y'. Since (18) holds, it is straightforward to see that $y' \leq b_k$. Thus, it also holds that all nodes appearing in $\mathrm{TOP}(k, G)$ belong to C, which may contain other nodes.

Since (19) holds, we have that for all $w \in C$ such that $\mathrm{b}(v) \geq y'$, it holds $\widetilde{\mathrm{b}}''(w)/(1 + \rho) \leq b(v)$. Similarly, for all $w \in C$ such that $\mathrm{b}(v) < y$, it holds $\widetilde{\mathrm{b}}''(w)/(1 + \rho) \leq y \leq b_k$. It follows from these properties of the nodes in C that $y'' \leq b_k$.

Since $y' \leq b_k$ and $y'' \leq b_k$, it holds $z \leq b_k$. Any w appearing in $\mathrm{TOP}(k, G)$ therefore has $\widetilde{\mathrm{b}}(w) \geq z(1 - \rho)$. It follows that all nodes appearing in $\mathrm{TOP}(k, G)$ will be in $\widetilde{\mathrm{TOP}}(k, G)$, satisfying the first part of (1) in the statement of the theorem.

The second part of (1) follows from (19), since for all nodes v appearing in $\mathrm{TOP}(k, G)$ it holds that $\mathrm{b}(v) \geq y'$.

Property (2) in the statement follows from (19), noticing that some of the nodes under consideration may have $\mathrm{b}(v) < y' \leq b_k$. □

5 APPROXIMATING BETWEENNESS CENTRALITY IN DYNAMIC GRAPHS

Fully dynamic graphs are graphs that evolve over time, with nodes and edges added and removed at each time step. In this section we show how the analysis based on Rademacher averages and pseudodimension presented in the previous section can be used to improve an algorithm by Hayashi et al. [23] that computes and keeps up-to-date an high-quality approximation of the BC of all nodes in a fully dynamic graph. The improvement consists in a large reduction in the number of samples needed by Hayashi et al.'s algorithm.

Hayashi et al. [23] introduced two fast data structures called the Hypergraph Sketch and the Two-Ball Index: the Hypergraph Sketch stores the BC estimations for all nodes, while the Two-Ball Index is used to store the SP DAGs and to understand which parts of the Hypergraph Sketch needs to be modified after an update to the graph (i.e., an edge or node insertion or deletion). Hayashi et al. [23] show how to use these data structures to maintain a set of estimation that is, with probability at least $1 - \delta$, an ε-approximation of the BC of all nodes in a fully dynamic graph.

Using the novel data structures results in orders-of-magnitude speedups w.r.t. previous contributions [7, 8]. The algorithm by Hayashi et al. [23] is based on a random sampling where pairs of nodes are sampled and the BC estimation of the nodes along the SPs between the two nodes are updated as necessary. The same sampling scheme is used by ABRA-s, but the Hayashi et al.'s algorithm uses a *fixed* number of samples computed at the beginning of the algorithm and never changed during execution. Hayashi et al.'s analysis of the number of samples necessary to obtain, with probability at least $1-\delta$, an ε-approximation of the BC of all nodes uses the union bound, resulting in a number of samples that depends on the logarithm of the number of nodes in the graph, i.e., $O(\varepsilon^{-2}(\log(|V|/\delta)))$ pairs of nodes must be sampled.

The progressive sampling approach and the stopping condition used by ABRA-s can be used in Hayashi et al.'s algorithm in place of a fixed sample size. As a result, the algorithm can decide when it has sampled enough to first populate the Hypergraph Sketch and the Two-Ball Index at the beginning of the algorithm.

After each update to the graph, the algorithm, in addition to performing bookkeeping operations on the Hypergraph Sketch and the Two-Ball Index, must keep up-to-date the set $\mathcal{V}_S$ and the map M (already used in ABRA-s). Once the data structures are up-to-date, the algorithm checks whether the stopping condition is still satisfied. If it is not, additional pairs of nodes are sampled and the Hypergraph Sketch and the Two-Ball Index are updated with the estimations resulting from these additional samples. The sampling of additional pairs continues until the stopping condition is satisfied, according to a sample schedule. There is no need to discard samples when the stopping condition is satisfied: the value ξ returned by the algorithm would just be even smaller than ε.

The overhead of additional checks of the stopping condition is minimal, because checking the stopping condition is extremely efficient, as we show in our experimental evaluation. On the other hand, the use of the progressive sampling scheme based on the Rademacher averages allows the algorithm to sample much fewer pairs of nodes than in the static sampling case based on the union bound: Riondato and Kornaropoulos [44] already showed that it is possible to sample much less than $O(\log |V|)$ nodes and, as we show in our experiments, the sample sizes required by the progressive sampling approach are even smaller than the ones used in the algorithm by Riondato and Kornaropoulos [44], and thus in the one by Hayashi et al. [23]. Reducing the number of samples naturally leads to a speedup, as the running time of the algorithms are, in a first approximation, linear in the number of samples. Additionally, the amount of space required to store the data structures would also decrease, as they now store information about fewer SP DAGs.

THEOREM 5.1. *The pair* $(\widetilde{B} = \{\widetilde{\mathrm{b}}(w), w \in V\}, \xi)$ *returned by this variant of the algorithm by Hayashi et al. [23] after each update to the graph has been processed, is such that* $\xi \leq \varepsilon$ *and*

$$\Pr(\exists w \in V \text{ s.t. } |\widetilde{\mathrm{b}}(w) - \mathrm{b}(w)| > \xi) < \delta \ .$$

The proof follows from the correctness of the algorithm by Hayashi et al. [23] and of ABRA-s (Thm. 4.1).

6 EXPERIMENTAL EVALUATION

In this section we presents the results of our experimental evaluation. We measure and analyze the performances of ABRA-s in terms of its runtime and sample size and accuracy, and compared them with those of the exact algorithm BA [13] and the approximation algorithm RK [44], which offers the same guarantees as ABRA-s, i.e., it computes, with probability at least $1-\delta$, an ε-approximation of the BC of all nodes.

Implementation and Environment. We implement ABRA-s and ABRA-s-fix in C++11, as an extension of the NetworKit library [49]. We use NLopt [26] for the optimization steps. The code is

available from http://matteo.rionda.to/software/ABRA-radebetw.tbz2. We performed the experiments on a machine with a AMD Phenom^{TM} II X4 955 processor and 16GB of RAM, running FreeBSD 12.

Datasets and Parameters. We use graphs of various nature (communication, citations, P2P, and social networks) from the SNAP repository [34]. The characteristics of the graphs are reported in the Table 2.

Graph	Directed	Vertices	Edges
cit-HepPh	N	34,546	421,578
email-Enron	N	36,682	183,831
p2p-Gnutella31	Y	62,586	147,892
soc-Epinions1	Y	75,879	508,837

Table 2. Characteristics of the graphs.

In our experiments we set ε in the range $[0.01, 0.03]$. In all the results we report, δ is fixed to 0.1. We experimented with different values for this parameter, and, as expected, it has a very limited impact on the nature of the results, given the logarithmic dependence of the sample size on δ. For the sample schedule, we used a geometrically increasing sample schedule with $S_i = c^i S_0$ for all $i \geq 1$, where S_0 is the r.h.s. of (16) and $c \in \{1.2, 1.5, 2.0, 3.0\}$.

We performed five runs for each combination of parameters. The variance between the different runs was insignificant, so we report, unless otherwise specified, the results for a random run.

The "Change" column in some of the following tables reports the *percentage change* in the metric of interest w.r.t. a baseline, computed as

$$\text{change} = 100\frac{\text{ABRA-s} - \text{baseline}}{\text{baseline}} .$$

For example, if the metric of interest were the runtime, and the measured runtime of ABRA-s was 6 seconds and the one for the baseline BA was 9 seconds, then the percentage change would be -33.33.

6.1 Accuracy

We evaluate the accuracy of ABRA-s by measuring the absolute error $|\widetilde{\mathrm{b}}(v) - \mathrm{b}(v)|$ for al nodes v in the graph. The theoretical analysis guarantees that this quantity should be at most ξ and therefore at most ε for all nodes, with probability at least $1 - \delta$.

An important result is that in *all* the thousands of runs of ABRA-s, the maximum error was *always* smaller than ξ, not just with probability $> 1 - \delta$.

We report statistics about the absolute error in Table 3, computed for runs with a geometric schedule with $c = 1.5$ (results for other values of c were qualitatively equivalent). The minimum error was always zero, so we do not report it in the table. The maximum error is almost always *an order of magnitude smaller than* ξ, and the average error is around *three orders of magnitude* smaller. The standard deviation, which is very close to the average, suggests that most of the errors are almost two orders of magnitude smaller than ε, as can be verified by considering the average error plus three standard deviations, which include approximately 95% of the nodes. As expected, the maximum error grows as ε^{-2}.

These results show that *ABRA-s is very accurate, more than what is guaranteed by the theoretical analysis.* This can be explained by the fact that the bounds to the sampling size, the stopping

Table 3. Comparison between ε and ξ, and error distribution. Geometric schedule with $c = 1.5$.

	Error bound		Error distribution		
Graph	$\varepsilon \times 10^3$	$\xi \times 10^3$	Avg $\times 10^3$	Stdev $\times 10^3$	Max $\times 10^3$
Cit-HepPh	10	8.816	0.014	0.027	0.920
	15	13.175	0.021	0.040	1.374
	20	17.582	0.027	0.052	2.044
	25	21.892	0.033	0.067	2.153
	30	26.322	0.039	0.082	2.956
Email-Enron	10	8.689	0.005	0.023	1.008
	15	13.047	0.007	0.034	1.393
	20	17.349	0.009	0.048	2.112
	25	21.757	0.011	0.061	3.119
	30	25.992	0.013	0.070	2.515
P2p-Gnutella31	10	8.978	0.009	0.026	0.445
	15	13.438	0.014	0.038	0.797
	20	17.945	0.018	0.050	0.795
	25	22.645	0.022	0.063	1.514
	30	27.102	0.026	0.077	1.671
Soc-Epinions1	10	9.872	0.007	0.021	0.839
	15	14.815	0.009	0.030	0.843
	20	19.678	0.012	0.041	1.595
	25	24.637	0.014	0.052	2.135
	30	29.738	0.016	0.063	2.467

condition, and the sample schedule are *conservative*, in the sense that ABRA-s may be sampling more than necessary to obtain an ε-approximation with probability at least $1 - \delta$. Tightening any of these components would result in a less conservative algorithm that offers the same approximation quality guarantees, and is an interesting research direction.

6.2 Runtime and Sample Size

We evaluate the runtime of ABRA-s and compare it with the ones of BA and RK. Doing so also gives us a chance to comment on the sample sizes used by ABRA-s and compare them with those used by RK. The results are reported in Tables 4 to 7 (one table per graph).

General comments. We can observe that the runtime is essentially proportional to $1/\varepsilon^2$ and to the sample size, as expected from the theoretical results. There is no clear dependency between the sample schedule and the resulting runtime, a phenomenon that we analyze in detail in the following paragraphs.

We broke down the runtime in three components: "sampling" (inclusive of the time needed to run s-t SP computations and updating the data structures), "stopping condition" (inclusive of computing ω^* and ξ), and "other". In all the runs, "sampling" accounted for more than 99% of the total runtime.[7] In other words, ABRA-s spent almost all its execution doing useful work, i.e., collecting samples, running s-t SP computations, and updating the data structures. This is contrast with other algorithms, e.g., RK, that need to compute some characteristic quantity of the graph at the beginning of the execution. Evaluating the stopping condition is extremely efficient: it accounted for less than one percent of the runtime.

Comparison with BA. The change w.r.t. the exact algorithm BA is significant, reaching at times values smaller than -90 (essentially a 10x speedup) and almost always smaller than -50 (a 2x speedup). The change tends to be smaller in magnitude as ε becomes smaller, as expected. It happened only once, on the email-Enron graph for $\varepsilon = 0.010$ and $c = 2.0$ that BA was faster than ABRA-s. This event, like similar ones we discuss later in this section, happens due to the fact that the sampling

[7] These results are not reported in any table, as they were essentially the same for all graphs and all combinations of the parameters, so we limit ourselves to commenting them.

schedule must be fixed in advance, i.e., it cannot be adaptive or the correctness of the algorithm is compromised. The consequence is that the algorithm may sample much more than needed. Specifically, when the value ξ computed at the end of an iteration is very close but still larger than ε, the algorithm must move to the next iteration, which will have a much larger sample size (c times larger), and where almost surely ξ will be much smaller than ε and the algorithm will terminate. This is indeed the case for email-Enron, $\varepsilon = 0.010$ and $c = 2.0$, as can be seen by looking at the error bound ξ computed by the algorithm and reported in the second-to-last column from the right in Table 5: ξ is is significantly less than ε, and at the iteration before the last, we checked that ξ was very close but larger than ε. Indeed ABRA-s with other sample schedules is able to stop much earlier.

Table 4. Runtime and sample size comparison for the cit-HepPh graph.

		Runtime (sec.)						
			Change vs.		Sample size		Err. bound $\xi \times 10^3$	
$\varepsilon \times 10^3$	Schedule	ABRA-s	BA	RK	ABRA-s	Change vs. RK	ABRA-s	Change vs. RK
10	$c = 1.2$	346.25	-32.14	-27.45	52,961	-27.48	9.746	-2.54
	$c = 1.5$	314.39	-38.38	-34.12	47,888	-34.42	8.816	-11.84
	$c = 2.0$	276.15	-45.88	-42.14	42,566	-41.71	8.778	-12.22
	$c = 3.0$	417.32	-18.21	-12.56	63,849	-12.57	7.119	-28.81
15	$c = 1.2$	156.97	-69.23	-11.63	23,982	-26.11	14.645	-2.37
	$c = 1.5$	139.28	-72.70	-34.44	21,684	-33.19	13.175	-12.16
	$c = 2.0$	123.92	-75.71	-41.67	19,274	-40.61	13.186	-12.09
	$c = 3.0$	187.97	-63.16	-11.53	28,911	-10.92	10.634	-29.11
20	$c = 1.2$	88.73	-82.61	-25.78	13,737	-24.76	19.393	-3.04
	$c = 1.5$	80.42	-84.24	-32.73	12,420	-31.97	17.582	-34.12
	$c = 2.0$	71.88	-85.91	-39.88	11,040	-39.53	17.601	-11.99
	$c = 3.0$	107.18	-78.99	-10.35	16,560	-9.30	14.238	-28.81
25	$c = 1.2$	57.80	-88.67	-24.40	8,949	-23.41	24.346	-2.61
	$c = 1.5$	51.43	-89.92	-32.74	8,091	-30.76	21.892	-29.67
	$c = 2.0$	46.65	-90.86	-38.99	7,192	-38.45	21.835	-12.66
	$c = 3.0$	68.98	-86.48	-9.78	10,787	-7.69	17.702	-29.19
30	$c = 1.2$	41.22	-91.92	-22.27	6,324	-22.06	29.478	-1.74
	$c = 1.5$	37.15	-92.72	-29.95	5,717	-29.54	26.322	-12.26
	$c = 2.0$	32.80	-93.57	-38.16	5,081	-37.38	26.374	-12.09
	$c = 3.0$	48.90	-90.42	-7.80	7,621	-6.08	21.223	-29.26

Comparison with RK. Comparing the runtime of ABRA-s with that of RK, we can observe that the former is almost always much faster than the latter, reaching up to 6x speedups. The one exception is again the email-Enron graph (Table 5), where ABRA-s is at times slower than RK. We already mentioned how this phenomenon can be traced back to the use of a static sampling schedule. We will show in Sect. 6.4 that, when given the same amount of samples, the ABRA-s-fix variant performs much better than RK.

This relative slowdown is compensated by a reduction, often significantly greater than the slowdown, in the error bound ξ (which for RK is ε): the additional time taken by ABRA-s to collect more samples is not "wasted", as it results in a better upper bound on the maximum error, as can be seen in the two rightmost columns in the tables. Hence the "rigidness" due to the fixed sample schedule is significantly softened.

We observe the slowdown only on email-Enron. The reason is that email-Enron is the graph we analyzed with the smallest diameter, therefore RK can use a small sample size. We show in Sect. 6.3 that RK scale very badly as the diameter grows, while ABRA-s does not. On the other hand, for ABRA-s we do not have practical bounds to the pseudodimension of the problem (see Sect. 4.2), thus the algorithm cannot deterministically stop after having sample a certain number of pairs of

nodes, like RK does. Finding better bounds to the pseudodimension of the problem, for example by proving Conjecture 4.11, would allow ABRA-s to stop earlier. A possible workaround, until that happens, is to run RK in parallel with ABRA-s, running both with $\delta' = \delta/2$, and making RK use the same random samples and the same s-t SP computations, and stopping as either algorithm stops and returning the corresponding approximation.

Table 5. Runtime and sample size comparison for the email-Enron graph. The cases when ABRA-s was worse than BA and/or RK are in bold font.

		Runtime (sec.)						
			Change vs.		Sample size		Err. bound $\xi \times 10^3$	
$\varepsilon \times 10^3$	Schedule	ABRA-s	BA	RK	ABRA-s	Change vs. RK	ABRA-s	Change vs. RK
10	$c = 1.2$	218.03	-9.10	-2.17	76,265	**4.44**	9.863	-1.37
	$c = 1.5$	206.13	-14.06	-7.51	71,832	-1.64	8.689	-13.11
	$c = 2.0$	246.06	**2.59**	**10.40**	85,132	**16.58**	7.585	-24.15
	$c = 3.0$	183.76	-23.38	-17.55	63,849	-12.57	8.319	-16.81
15	$c = 1.2$	99.89	-58.35	**0.25**	34,535	**6.41**	14.853	-0.98
	$c = 1.5$	93.29	-61.10	-6.09	32,526	**0.22**	13.047	-13.02
	$c = 2.0$	110.57	-53.90	**11.30**	38,548	**18.77**	11.368	-24.21
	$c = 3.0$	82.95	-65.42	-16.50	28,911	-10.92	12.446	-17.03
20	$c = 1.2$	57.08	-76.20	**2.01**	19,782	**8.35**	19.698	-1.51
	$c = 1.5$	53.63	-77.64	-4.15	18,630	**2.04**	17.349	-34.76
	$c = 2.0$	63.17	-73.66	**12.89**	22,080	**20.94**	15.105	-24.47
	$c = 3.0$	47.05	-80.38	-16.02	16,560	-9.30	16.623	-16.89
25	$c = 1.2$	37.38	-84.42	**5.18**	12,887	**10.29**	24.647	-1.41
	$c = 1.5$	34.81	-85.49	-2.06	12,137	**3.87**	21.757	-30.61
	$c = 2.0$	40.98	-82.92	15.30	14,384	**23.10**	18.756	-24.98
	$c = 3.0$	30.82	-87.15	-13.28	10,787	-7.69	20.801	-16.80
30	$c = 1.2$	26.23	-89.06	**6.27**	9,107	**12.24**	29.619	-1.27
	$c = 1.5$	24.77	-89.67	**0.38**	8,576	**5.69**	25.992	-13.36
	$c = 2.0$	29.29	-87.79	**18.68**	10,162	**25.24**	22.461	-25.13
	$c = 3.0$	21.86	-90.88	-11.42	7,621	-6.08	24.781	-17.40

Table 6. Runtime and sample size comparison for the p2p-Gnutella31 graph.

		Runtime (sec.)						
			Change vs.		Sample size		Err. bound $\xi \times 10^3$	
$\varepsilon \times 10^3$	Schedule	ABRA-s	BA	RK	ABRA-s	Change vs. RK	ABRA-s	Change vs. RK
10	$c = 1.2$	33.06	-81.47	-69.27	30,648	-63.09	9.859	-1.41
	$c = 1.5$	34.11	-80.89	-68.29	31,925	-61.55	8.978	-10.22
	$c = 2.0$	45.80	-74.34	-57.43	42,566	-48.73	7.732	-22.68
	$c = 3.0$	68.51	-61.61	-36.32	63,849	-23.10	6.292	-37.08
15	$c = 1.2$	15.49	-91.32	-30.27	13,878	-62.39	14.901	-0.66
	$c = 1.5$	15.29	-91.43	-68.18	14,456	-60.82	13.438	-10.42
	$c = 2.0$	20.55	-88.49	-57.24	19,274	-47.77	11.636	-22.43
	$c = 3.0$	31.18	-82.53	-35.12	28,911	-21.65	9.433	-37.12
20	$c = 1.2$	8.60	-95.18	-68.78	7,949	-61.70	19.943	-0.29
	$c = 1.5$	8.87	-95.03	-67.79	8,280	-60.11	17.945	-32.81
	$c = 2.0$	11.81	-93.38	-57.12	11,040	-46.81	15.474	-22.63
	$c = 3.0$	17.67	-90.10	-35.82	16,560	-95.01	12.513	-37.44
25	$c = 1.2$	5.60	-96.86	-68.09	5,178	-61.02	24.719	-1.12
	$c = 1.5$	5.88	-96.70	-66.47	5,394	-59.40	22.645	-28.22
	$c = 2.0$	7.88	-95.58	-55.08	7,192	-45.86	19.200	-23.20
	$c = 3.0$	11.30	-93.67	-35.59	10,787	-18.80	15.669	-37.32
30	$c = 1.2$	4.01	-97.75	-67.85	3,659	-60.34	29.879	-0.40
	$c = 1.5$	4.19	-97.65	-66.44	3,811	-58.69	27.102	-9.66
	$c = 2.0$	5.56	-96.88	-55.43	5,081	-44.93	23.430	-21.90
	$c = 3.0$	8.28	-95.36	-33.65	7,621	-17.40	18.790	-37.37

Table 7. Runtime and sample size comparison for the soc-Epinions1 graph.

		Runtime (sec.)						
			Change vs.		Sample size		Err. bound $\xi \times 10^3$	
$\varepsilon \times 10^3$	Schedule	ABRA-s	BA	RK	ABRA-s	Change vs. RK	ABRA-s	Change vs. RK
10	$c = 1.2$	185.48	-71.79	-55.03	44,134	-39.56	9.976	-0.24
	$c = 1.5$	132.62	-79.83	-67.85	31,925	-56.28	9.872	-1.28
	$c = 2.0$	178.07	-72.91	-56.83	42,566	-41.71	8.495	-15.05
	$c = 3.0$	267.00	-59.39	-35.27	63,849	-12.57	6.908	-30.92
15	$c = 1.2$	83.80	-87.25	-35.60	19,985	-38.42	14.994	-0.04
	$c = 1.5$	60.02	-90.87	-73.98	14,456	-55.46	14.815	-1.24
	$c = 2.0$	80.13	-87.81	-65.26	19,274	-40.61	12.648	-15.68
	$c = 3.0$	120.28	-81.70	-47.85	28,911	-10.92	10.422	-30.52
20	$c = 1.2$	48.22	-92.67	-71.08	11,447	-37.30	19.982	-0.09
	$c = 1.5$	34.29	-94.78	-79.43	8,280	-54.65	19.678	-25.93
	$c = 2.0$	46.78	-92.88	-71.94	11,040	-39.53	16.930	-15.35
	$c = 3.0$	69.24	-89.47	-58.46	16,560	-9.30	13.831	-30.85
25	$c = 1.2$	31.34	-95.23	-77.14	7,457	-36.18	24.949	-0.20
	$c = 1.5$	22.93	-96.51	-83.27	5,394	-53.84	24.637	-21.29
	$c = 2.0$	29.89	-95.45	-78.20	7,192	-38.45	20.892	-16.43
	$c = 3.0$	45.32	-93.11	-66.94	10,787	-7.69	17.351	-30.59
30	$c = 1.2$	22.35	-96.60	-81.50	5,270	-35.05	29.691	-1.03
	$c = 1.5$	15.88	-97.58	-86.86	3,811	-53.03	29.738	-0.87
	$c = 2.0$	21.48	-96.73	-82.22	5,081	-37.38	25.196	-16.01
	$c = 3.0$	32.11	-95.12	-73.42	7,621	-6.08	20.495	-31.68

6.3 Scalability

In this section we compare the scalability of ABRA-s to that of RK as the vertex-diameter of the graph grows. This choice is motivated by the fact that approximate betweenness estimation algorithms tend[8] to scale well as the numbers of nodes and edges grows, because in many graph evolution models the vertex-diameter of the network tends to shrink as these quantities increase [33]. On the other hand, RK is known to be susceptible to growth in the vertex-diameter, because its sample size depends on this quantity. We here evaluate the resilience of ABRA-s to changes in the vertex-diameter.

We create artificial graphs using the email-Enron graph as the starting point, then selecting a node v at random, and adding a *tail* (or chain) of k edges starting from v. Thinking of the original graph as a balloon, the tail can be imagined as the string to hold the balloon. We use $k = 20, 40, 80, 160$. The email-Enron graph is undirected and has a (vertex)-diameter of 11, so by adding the tail the resulting graph had a much larger vertex-diameter, hence RK would have to collect more samples.

The results are presented in Table 8. ABRA-s used a geometric sample schedule with $c = 1.5$. It is evident that ABRA-s's runtime and sample size scale well and actually are independent from changes in the tail length, while this is clearly not the case for RK. This phenomena can be explained by the fact that ABRA-s's analysis is based on Rademacher averages, which take into account the distribution of the path lengths, while RK uses VC-dimension, which considers only the worst case.

6.4 Fixed sample size

In this section we compare the performances of ABRA-s-fix, presented in Sect. 4.4, and RK. We report the results on Email-Enron with $\delta = 0.1$ in Table 9. The sample sizes are chosen as a result of the way RK computes its sample size: we fix ε and compute the sample size such that, with

[8]The one algorithm that does not scale well being the algorithm by Brandes and Pich [14] due to the fact that its sample size depends on the number of nodes in the graph.

Table 8. Scalability as the vertex-diameter grows due to the addition of a tail of length k.

		Runtime (sec.)			Sample size		
ε	k	ABRA-s	RK	Change	ABRA-s	RK	Change
0.005	20	800.44	914.67	-12.49	281,870	332,104	-15.13
	40	802.54	1027.48	-21.89	281,870	372,104	-24.25
	80	803.72	1140.89	-29.55	281,870	412,104	-31.60
	160	800.98	1254.28	-36.14	281,870	452,104	-37.65
0.010	20	202.89	228.98	-11.39	71,832	83,026	-13.48
	40	203.79	256.92	-20.68	71,832	93,026	-22.78
	80	201.63	284.96	-29.24	71,832	103,026	-30.28
	160	203.34	313.55	-35.15	71,832	113,026	-36.45
0.015	20	92.13	101.30	-9.06	32,526	36,901	-11.86
	40	91.76	114.27	-19.69	32,526	41,345	-21.33
	80	91.13	126.80	-28.13	32,526	45,790	-28.97
	160	91.37	139.45	-34.47	32,526	50,234	-35.25
0.020	20	52.92	57.10	-7.32	18,630	20,757	-10.25
	40	51.84	64.35	-19.43	18,630	23,257	-19.90
	80	52.08	71.40	-27.06	18,630	25,757	-27.67
	160	52.81	78.56	-32.77	18,630	28,257	-34.07
0.025	20	34.13	36.73	-7.06	12,137	13,285	-8.64
	40	34.32	41.11	-16.51	12,137	14,885	-18.46
	80	33.98	45.65	-25.57	12,137	16,485	-26.38
	160	34.39	50.05	-31.29	12,137	18,085	-32.89
0.030	20	24.25	25.45	-4.73	8,576	9,226	-7.05
	40	16.12	28.61	-43.64	5,717	10,337	-44.69
	80	24.02	31.70	-24.24	8,576	11,448	-25.09
	160	24.44	34.71	-29.57	8,576	12,559	-31.71

probability at least $1 - \delta$, RK outputs an ε-approximation. We then run ABRA-s-fix with the same sample size and value δ, and observe the value for ξ, the second component of the output of ABRA-s-fix, to compare it with ε.

Table 9. Comparison between ε and ξ, and change for ABRA-s-fix and RK on email-Enron.

	Error bound		Change	
Sample size	$\varepsilon \times 10^3$	$\xi \times 10^3$	Error Bound	Sample size
73026	10	7.74	-22.52	-66.56
32456	15	11.71	-21.87	-63.83
18257	20	15.71	-21.45	-62.09
11685	25	19.76	-20.94	-59.98
8114	30	23.79	-20.67	-58.90

The fourth column from the left reports the percentage change between ξ and ε. The fifth column from the left shows the percentage change between the sample size used by ABRA-s-fix and the sample size that RK would need to achieve an ε equal to the ξ reported in the third column. The reduction in the bound to the maximum error exceeds 20%, and the reduction in sample size is around 60%.

These results are even more remarkable when one considers the fact that RK is given the big advantage of computing the diameter of the graph, which is needed by RK to compute the sample size. Without this knowledge, RK would not even be able to start. On the other hand, ABRA-s-fix has no need for any information about the complete graph, and can just start sampling and compute its quality guarantees only on the sample, thanks to the use of Rademacher averages.

7 CONCLUSIONS

We presented ABRA, a family of sampling-based algorithms for computing and maintaining high-quality approximations of (variants of) the BC of all nodes in static and dynamic graphs with updates

(both deletions and insertions). We discussed a number of variants of our basic algorithm, including finding the top-k nodes with higher BC, using improved estimators, using a fixed sample size, and special cases when there is a single SP betwen nodes. ABRA greatly improves, theoretically and experimentally, the current state of the art. The analysis relies on Rademacher averages and on pseudodimension, fundamental concepts from statistical learning theory. To our knowledge this is the first application of these results and ideas to graph mining, and we believe that they should be part of the toolkit of any algorithm designer interested in efficient algorithms for data analysis.

ACKNOWLEDGMENTS

The authors are thankful to Elisabetta Bergamini and Christian Staudt for their help with the NetworKit code, to Emanuele Natale and Michele Borassi for finding a mistake, now corrected, in one of our proofs, and to Nikolaj Tatti and Fabio Vandin for the stimulating discussions.

This work was supported in part by the National Science Foundation grant IIS-1247581 (https://www.nsf.gov/awardsearch/showAward?AWD_ID=1247581) and the National Institutes of Health grant R01-CA180776 (https://projectreporter.nih.gov/project_info_details.cfm?icde=0&aid=8685211).

This work is supported, in part, by funding from Two Sigma Investments, LP. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflects the views of Two Sigma Investments, LP or the National Science Foundation.

REFERENCES

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, and Sandro Ridella. 2014. A Deep Connection Between the Vapnik-Chervonenkis Entropy and the Rademacher Complexity. *IEEE Transactions on Neural Networks and Learning Systems* 25, 12 (2014), 2202–2211.

[2] Jac. M. Anthonisse. 1971. *The rush in a directed graph.* Technical Report BN 9/71. Stichting Mathematisch Centrum, Amsterdam, Netherlands.

[3] Martin Anthony and Peter L. Bartlett. 1999. *Neural Network Learning – Theoretical Foundations.* Cambridge University Press.

[4] Martin Anthony and John Shawe-Taylor. 1993. A result of Vapnik with applications. *Discrete Applied Mathematics* 47, 3 (1993), 207–217.

[5] David A. Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. 2007. Approximating Betweenness Centrality. In *Algorithms and Models for the Web-Graph*, Anthony Bonato and FanR.K. Chung (Eds.). Lecture Notes in Computer Science, Vol. 4863. Springer Berlin Heidelberg, 124–137.

[6] Peter L. Bartlett and Gábor Lugosi. 1999. An inequality for uniform deviations of sample averages from their means. *Statistics & Probability Letters* 44, 1 (1999), 55–62.

[7] Elisabetta Bergamini and Henning Meyerhenke. 2015. Fully-dynamic Approximation of Betweenness Centrality. In *Proceedings of the 23rd European Symposium on Algorithms (ESA '15)*. 155–166.

[8] Elisabetta Bergamini and Henning Meyerhenke. 2016. Approximating Betweenness Centrality in Fully-dynamic Networks. *Internet Mathematics* 12, 5 (2016), 281–314.

[9] Elisabetta Bergamini, Henning Meyerhenke, and Christian L. Staudt. 2015. Approximating Betweenness Centrality in Large Evolving Networks. In *17th Workshop on Algorithm Engineering and Experiments (ALENEX '15)*. SIAM, 133–146.

[10] Michele Borassi and Emanuele Natale. 2016. KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation. *CoRR* abs/1604.08553 (2016).

[11] Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi. 2005. Theory of classification: A survey of some recent advances. *ESAIM: Probability and Statistics* 9 (2005), 323–375.

[12] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization.* Cambridge University Press.

[13] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology* 25, 2 (2001), 163–177.

[14] Ulrik Brandes and Christian Pich. 2007. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos* 17, 7 (2007), 2303–2318.

[15] Corinna Cortes, Spencer Greenberg, and Mehryar Mohri. 2013. Relative Deviation Learning Bounds and Generalization with Unbounded Loss Functions. *CoRR* abs/1310.5796 (Oct 2013).

[16] Tapio Elomaa and Matti Kääriäinen. 2002. Progressive Rademacher Sampling. In *AAAI/IAAI*, Rina Dechter and Richard S. Sutton (Eds.). AAAI Press / The MIT Press, 140–145.

[17] Dóra Erdős, Vatche Ishakian, Azer Bestavros, and Evimaria Terzi. 2015. A Divide-and-Conquer Algorithm for Betweenness Centrality. In *SIAM International Conference on Data Mining (SDM '15)*. SIAM, 433–441.

[18] Linton C. Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* 40 (1977), 35–41.

[19] Robert Geisberger, Peter Sanders, and Dominik Schultes. 2008. Better Approximation of Betweenness Centrality. In *10th Workshop on Algorithm Engineering and Experiments (ALENEX '08)*. SIAM, 90–100.

[20] O. Green, R. McColl, and David A. Bader. 2012. A Fast Algorithm for Streaming Betweenness Centrality. In *2012 International Conference on Privacy, Security, Risk and Trust (PASSAT '12)*. IEEE, 11–20.

[21] Sariel Har-Peled and Micha Sharir. 2011. Relative (p, ε)-Approximations in Geometry. *Discrete & Computational Geometry* 45, 3 (2011), 462–496.

[22] David Haussler. 1992. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation* 100, 1 (1992), 78–150.

[23] Takanori Hayashi, Takuya Akiba, and Yuichi Yoshida. 2015. Fully Dynamic Betweenness Centrality Maintenance on Massive Networks. *Proceedings of the VLDB Endowment* 9, 2 (2015), 48–59.

[24] Riko Jacob, Dirk Koschützki, KatharinaAnna Lehmann, Leon Peeters, and Dagmar Tenfelde-Podehl. 2005. Algorithms for Centrality Indices. In *Network Analysis*, Ulrik Brandes and Thomas Erlebach (Eds.). Lecture Notes in Computer Science, Vol. 3418. Springer Berlin Heidelberg, 62–82.

[25] Shiyu Ji and Zenghui Yan. 2016. Refining Approximating Betweenness Centrality Based on Samplings. *CoRR* abs/1608.04472 (2016).

[26] Steven G. Johnson. 2014. The NLopt nonlinear-optimization package. (2014). http://ab-initio.mit.edu/nlopt.

[27] Miray Kas, Matthew Wachs, Kathleen M. Carley, and L. Richard Carley. 2013. Incremental Algorithm for Updating Betweenness Centrality in Dynamically Growing Networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '13)*. IEEE/ACM, 33–40.

[28] Vladimir Koltchinskii. 2001. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory* 47, 5 (July 2001), 1902–1914.

[29] Vladimir Koltchinskii, C.T. Abdallah, Marco Ariola, Peter Dorato, and Dmitry Panchenko. 2000. Improved sample complexity estimates for statistical learning control of uncertain systems. *IEEE Trans. Autom. Control* 45, 12 (Dec. 2000), 2383–2388. https://doi.org/10.1109/9.895579

[30] Nicolas Kourtellis, Tharaka Alahakoon, Ramanuja Simha, Adriana Iamnitchi, and Rahul Tripathi. 2012. Identifying high betweenness centrality nodes in large social networks. *Social Network Analysis and Mining* 3, 4 (2012), 899–914.

[31] Nicolas Kourtellis, Gianmarco De Francisci Morales, and Francesco Bonchi. 2015. Scalable Online Betweenness Centrality in Evolving Graphs. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2494–2506.

[32] Min-Joong Lee, Jungmin Lee, Jaimie Yejean Park, Ryan Hyun Choi, and Chin-Wan Chung. 2012. QUBE: A Quick Algorithm for Updating Betweenness Centrality. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*. IW3C2, 351–360.

[33] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data* 1, 1 (2007).

[34] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).

[35] Yi Li, Philip M. Long, and Aravind Srinivasan. 2001. Improved Bounds on the Sample Complexity of Learning. *J. Comput. System Sci.* 62, 3 (2001), 516–527.

[36] Meghana Nasre, Matteo Pontecorvi, and Vijaya Ramachandran. 2014. Betweenness centrality – incremental and faster. In *International Symposium on Mathematical Foundations of Computer Science (MFCS '14)*. 577–588.

[37] Meghana Nasre, Matteo Pontecorvi, and Vijaya Ramachandran. 2014. Decremental All-Pairs ALL Shortest Paths and Betweenness Centrality. In *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC '14)*. 766–778.

[38] Mark E. J. Newman. 2010. *Networks – An Introduction*. Oxford University Press.

[39] Luca Oneto, Alessandro Ghio, Davide Anguita, and Sandro Ridella. 2013. An improved analysis of the Rademacher data-dependent bound using its self bounding property. *Neural Networks* 44 (2013), 107–111.

[40] Luca Oneto, Alessandro Ghio, Sandro Ridella, and Davide Anguita. 2016. Global Rademacher Complexity Bounds: From Slow to Fast Convergence Rates. *Neural Processing Letters* 43, 2 (2016), 567–602.

[41] David Pollard. 1984. *Convergence of stochastic processes*. Springer-Verlag.

[42] Matteo Pontecorvi and Vijaya Ramachandran. 2015. Fully Dynamic Betweenness Centrality. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC '15)*. 331–342.

[43] Foster Provost, David Jensen, and Tim Oates. 1999. Efficient Progressive Sampling. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*. 23–32.

[44] Matteo Riondato and Evgenios M. Kornaropoulos. 2016. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery* 30, 2 (2016), 438–475.

[45] Matteo Riondato and Eli Upfal. 2015. Mining Frequent Itemsets through Progressive Sampling with Rademacher Averages. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, 1005–1014. Extended version available from http://matteo.rionda.to/papers/RiondatoUpfal-FrequentItemsetsSamplingRademacher-KDD.pdf.

[46] Matteo Riondato and Eli Upfal. 2016. Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 1145–1154.

[47] Ahmet Erdem Sarıyüce, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek. 2013. Shattering and Compressing Networks for Betweenness Centrality. In *SIAM International Conference on Data Mining (SDM '13)*. SIAM, 686–694.

[48] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

[49] Christian L. Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. 2016. NetworKit: An Interactive Tool Suite for High-Performance Network Analysis. *Network Science* to appear (2016).

[50] Vladimir N. Vapnik. 1999. *The Nature of Statistical Learning Theory*. Springer-Verlag.