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 and number of samples, 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 (March 2017), 39 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Centrality measures are fundamental concepts in graph analysis: they assign to each node or edge in the network a score that quantifies some notion of importance of the node/edge in the network [41]. 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, 19] (see Sect. 3 for formal definitions).

Brandes [14] 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, and 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 [50].
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.
 1556-4681/2017/3-ART1 $15.00
DOI: 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 [21, 28, 33, 34, 39, 40, 46] 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 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 [29, 52] and pseudodimension [45], fundamental concepts from the field of statistical learning theory [54]. 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, 48]. 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 [48], 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 and number of samples, the state-of-the-art methods offering the same guarantees [48].

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. Additional details can be found in the Appendices.

2 RELATED WORK

The definition of Betweenness Centrality comes from the sociology literature [2, 19], 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 [14] is currently the asymptotically fastest algorithm for computing the exact BC values for all nodes in

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
[15, 24, 25]	nodes	$O\left(\frac{1}{\varepsilon^2}\left(\ln\|V\| + \ln\frac{1}{\delta}\right)\right)$	Hoeffding's inequality, Union bound
[8, 48]	shortest paths	$O\left(\frac{1}{\varepsilon^2}\left(\log_2 \mathsf{VD}(G) + \ln\frac{1}{\delta}\right)\right)$ [†]	VC-Dimension
This work	pairs of nodes	Variable, at most $O\left(\frac{1}{\varepsilon^2}\left(\log_2 \mathsf{L}(G) + \ln\frac{1}{\delta}\right)\right)$ [‡]	Rademacher Averages, Pseudodimension

[*] See Def. 3.2 for the formal definition.
[†] $\mathsf{VD}(G)$ 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.

the network. A number of works also explored heuristics to improve BA [18, 51], 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. [25] and Brandes and Pich [15], and successive works explored the tradeoff space of sampling-based algorithms [7–9, 48]. Other works focused on estimating the betweenness centrality of a single target node, rather than on obtaining uniform guarantees for all the nodes [5, 26]. 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 [48, 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 [48] present algorithms that employ the Vapnik-Chervonenkis (VC) dimension [54] 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 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 [48].

A number of works [21, 28, 33, 34, 39, 40, 46] 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 [14] 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 [48] 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 [48] 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. [24] 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. [25] and Brandes and Pich [15]. 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 [17] and Riondato and Upfal [49] 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.

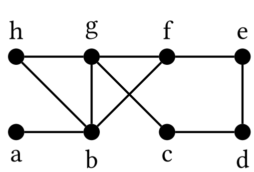
3.1 Graphs and Betweenness Centrality

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, 19]). 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 [48, 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 [41] and one limited to random walks of a fixed length [32]. Our results can be extended to many of these variants, following the same discussion as in [48, 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 $\tilde{B} = \{\tilde{\mathrm{b}}(w), w \in V\}$ such that, for all $w \in V$,

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

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 [29] 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 [54] but their usefulness extends way beyond the learning framework [49]. 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 [52] 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

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

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

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

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

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

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., [52, Thm. 26.5], and compare the bounds presented therein with the ones presented in the following.) Better (i.e., tighter) although more complex bounds are available [43, 44]. Specifically, we use Thm. 3.3, whose proof is presented in Appendix B. It extends [43, Thm. 3.11] to a probabilistic tail bound for the supremum of the *absolute value* of the deviation, and specializes it for functions with co-domain $[0,1]$.

[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. We show how to extend the results to the case of general *non-negative* functions (i.e., with co-domain $[0,b]$ for $b > 0$) in Appendix B. Extension to intervals of the reals are also possible.

[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.

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* $1-\eta$,

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

Even more refined bounds than the ones presented above are available [44] 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 ($|\mathcal{V}_{\mathcal{S}}| \leq |\mathcal{F}|$, 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*

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

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

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

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 [13]. More refined bounds can be derived but are more computationally expensive to compute [1].

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 $\widetilde{B}=\{\widetilde{\mathrm{b}}(w), w\in V\}$ 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 $f_w:\mathcal{D}\rightarrow[0,1]$ be the function

$$f_w(u,v)=\frac{\sigma_{uv}(w)}{\sigma_{uv}}, \tag{7}$$

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

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

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

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

For each $w\in V$ consider the vector

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

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

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

It is possible, if not likely, that $|\mathcal{V}_{\mathcal{S}}|\leq|V|$ as there may be two different nodes u and v with $\mathbf{v}_u=\mathbf{v}_v$. 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 $\widetilde{B} = \{\widetilde{\mathrm{b}}(w) = \|\mathbf{v}_w\|_1/\ell, w \in V\}$ is, with probability at least η, 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 each pair of nodes has been sampled, 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 $\xi_{\mathcal{S}_i} \leq \varepsilon$. This is ABRA-s *stopping condition*:[3] when it holds, ABRA-s returns

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

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

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 [14, 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 $\mathsf{P}_u(w)$ 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

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

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 [14]. 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

[3] A different stopping condition for generic progressive sampling using Rademacher average was presented by Koltchinskii et al. [30] and used by Elomaa and Kääriäinen [17]. We choose to use ours because it is more efficient to compute.
[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.

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 $B = \{\mathrm{b}(w), w \in v\}$.

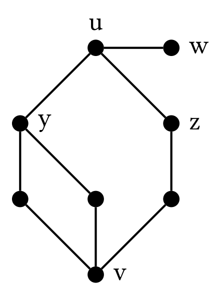
1 $\mathcal{D} \leftarrow \{(u, v) \in V \times V, u \neq v\}$
2 $S_0 \leftarrow 0$,
3 $\mathbf{0} = (0)$
4 $\mathcal{V} = \{\mathbf{0}\}$
5 **foreach** $w \in V$ **do** $M[w] = \mathbf{0}$
6 $\mathsf{c}_{\mathbf{0}} \leftarrow |V|$
7 $i \leftarrow 1, j \leftarrow 1$
8 **while** *True* **do**
9   **for** $\ell \leftarrow 1$ **to** $S_i - S_{i-1}$ **do**
10     $(u, v) \leftarrow$ `uniform_random_sample(`$\mathcal{D}$`)`
11   `compute_SPs(`u, v`)  //Truncated SP computation`
12   **if** *reached* v **then**
13   **foreach** $z \in \mathsf{P}_u[v]$ **do** $\sigma_{zv} \leftarrow 1$
14   **foreach** *node* w *on a SP from* u *to* v*, in reverse order by* $\mathrm{d}(u, w)$ **do**
15         $\sigma_{uv}(w) \leftarrow \sigma_{uw}\sigma_{wv}$
16         $\mathbf{v} \leftarrow M[w]$
17         $\mathbf{v}' \leftarrow \mathbf{v} \cup \{(j, \sigma_{uv}(w)/\sigma_{uv})\}$
18   **if** $\mathbf{v}' \notin \mathcal{V}$ **then**
19           $\mathsf{c}_{\mathbf{v}'} \leftarrow 1$
20           $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{v}'\}$
21   **else** $\mathsf{c}_{\mathbf{v}'} \leftarrow \mathsf{c}_{\mathbf{v}'} + 1$
22         $M[w] \leftarrow \mathbf{v}'$
23   **if** $\mathsf{c}_{\mathbf{v}} > 1$ **then** $\mathsf{c}_{\mathbf{v}} \leftarrow \mathsf{c}_{\mathbf{v}} - 1$
24   **else** $\mathcal{V} \leftarrow \mathcal{V} \setminus \{\mathbf{v}\}$
25   **foreach** $z \in \mathsf{P}_u[w]$ **do** $\sigma_{zv} \leftarrow \sigma_{zv} + \sigma_{wv}$
26     $j \leftarrow j + 1$
27   $\omega_i^* \leftarrow \min_{r \in \mathbb{R}^+} \frac{1}{r} \ln \left(\sum_{\mathbf{v} \in \mathcal{V}} \exp \left[r^2 \|\mathbf{v}\|^2 / (2S_i^2) \right] \right)$
28   $\gamma_i \leftarrow \ln(3/\delta) + i \ln 2$
29   $\xi_{\mathcal{S}_i} \leftarrow 2\omega_i^* + \frac{\gamma_i + \sqrt{\gamma_i(\gamma_i + 4S_i\omega_i^*)}}{2S_i} + \sqrt{\frac{\gamma_i}{2S_i}}$
30   **if** $\xi_{\mathcal{S}_i} \leq \varepsilon$ **then**
31   **break**
32   **else**
33     $i \leftarrow i + 1$
34 $\widetilde{B} \leftarrow \{\widetilde{\mathrm{b}}(w) \leftarrow \|M[w]\|_1 / S_i, w \in V\}$
35 **return** $(\widetilde{B}, \xi_{\mathcal{S}_i})$

$\sigma_{uv}(w)/\sigma_{uv}$ 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 $\mathsf{c}_{\mathbf{v}'}$ by one (lines 16 to 22). Finally, the algorithm decrements the counter $\mathsf{c}_\mathbf{v}$ by one, and if $\mathsf{c}_\mathbf{v}$ becomes equal to zero, ABRA-s removes $\mathbf{v}$ from $\mathcal{V}_\mathcal{S}$ (line 24). At this point, the algorithm moves

[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.

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.



(a) A portion of the graph

Vertex x	$M[x]$
$\vdots$	$\vdots$
w	$\mathbf{a}$
y	$\mathbf{a}$
z	$\mathbf{a}$
$\vdots$	$\vdots$

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

(b) Before the update

Vertex x	$M[x]$
$\vdots$	$\vdots$
w	$\mathbf{a}$
y	$\mathbf{b}$
z	$\mathbf{c}$
$\vdots$	$\vdots$

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

(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.

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.

The following is a sketch of the proof of Thm. 4.1. The complete formal proof can be found in Appendix A.

PROOF (SKETCH). The collection $\mathcal{S}_i$ of pairs of nodes sampled by the algorithm up to any iteration $i \geq 0$ is a collection of independent, uniformly distributed random variables. Using the union bound, the definition of $\xi_{\mathcal{S}_i}$, and Thm. 3.3, we have:

$$\Pr\left(\exists i > 0 \text{ s.t. } \sup_{w \in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi_{\mathcal{S}_i}\right) \leq \sum_{i=1}^{\infty} \Pr\left(\sup_{w \in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi_{\mathcal{S}_i}\right) \leq \sum_{i=1}^{\infty} \frac{\delta}{2^i} \leq \delta \ . \tag{8}$$

Suppose that it is indeed the case that

$$\sup_{w \in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \leq \xi_{\mathcal{S}_i}, \text{for all } i \geq 0,$$

which, from (8), happens with probability at least $1 - \delta$. Then in particular it is true for the last iteration r of ABRA-s, and given the stopping condition, we have:

$$\sup_{w \in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_r}(w) - \mathrm{b}(w)| \leq \xi_{\mathcal{S}_r} \quad (\leq \varepsilon) \ . \qquad \square$$

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

$$S_1 \geq \frac{(1 + 4\varepsilon + \sqrt{1 + 8\varepsilon}) \ln(6/\delta)}{4\varepsilon^2} \ . \tag{9}$$

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 $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_1)$, except that it is *non-negative*, i.e., $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_1) \geq 0$. 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

$$\frac{\ln(6/\delta)}{S_1} + \sqrt{\frac{\ln(6/\delta)}{2S_1}} \leq \varepsilon \ . \tag{10}$$

Solving for S_1 under the domain constraints $S_1 \geq 1$, $\delta \in (0, 1)$, $\varepsilon \in (0, 1)$, gives the unique solution in (9).

One may not want to set S_1 equal to the r.h.s. of (9), because its derivation basically assumes that $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_1) = 0$, 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 the following.

Let r be a value in $(0, \varepsilon/2)$. We are going to use r as a (tentative) upper bound on $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_1)$. We use r as a (tentative) upper bound on $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_1)$. We ask what is the minimum size S_1 for which $\xi_{\mathcal{S}_1}$ would be at most ε, under the assumption that $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_1) \leq r$. Formally, let $\gamma_1 = \ln(6/\delta)$, to simplify the notation. We want to solve the inequality

$$\sqrt{\frac{\gamma_1}{2S_1}} + 2\frac{\gamma_1 + \sqrt{\gamma_1(\gamma_1 + 2S_1 r)}}{S_1} + 2r \leq \varepsilon \tag{11}$$

where S_1 acts as the unknown. The l.h.s. of this inequality is obtained from (4) using r in place of $\mathsf{R}_{\mathcal{F}}(\mathcal{S})$, S_1 in place of ℓ, $\delta/2$ in place of η, and slightly reorganize the terms for readability. As can be verified using a symbolic mathematical computation program, finding the solution to the above inequality requires computing the roots of the third grade polynomial (in S_1)

$$\begin{aligned} &4(\varepsilon - 2r)^4 S_1^3 + 4(2r - 2\varepsilon - 1)(\varepsilon - 2r)^2 \gamma S_1^2 \\ &+ (4(r)^2 + (1 + 2\varepsilon)^2 - 4r(3 + 2\varepsilon))\gamma^2 S_1 - 2\gamma^3 \end{aligned}$$

which can be done easily [42]. This polynomial is obtained from (11) by repeated squaring, therefore some spurious roots may have been introduced and must be filtered out when choosing S_1.

The assumption $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_{i+1}) \leq r$, which is not guaranteed to be true, is what makes this procedure a heuristic, but if the assumption is true, then the algorithm will stop after the first iteration. We experiment with different values for r in our experimental evaluation.

Note that the first sample size is fixed before the algorithm even starts (as is the whole sample schedule), so it does not depend in any way on the random choices made by the algorithm.

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 [47].

The approach we now describe is an *heuristic* that nevertheless aims at making use of all available information at the end of each iteration to the most possible extent, with the goal of increasing the chances that the stopping condition is satisfied at the next iteration, so that the algorithm saves time by avoiding to check the stopping condition at iterations at which it is unlikely to be satisfied.

Let $(S_i)_{i\geq 1}$ be a fixed sample schedule. The intuition is to assume that ω_i^*, which is an upper bound on $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_i)$, is also an upper bound on $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_j)$ for all $j > i$ whatever $\mathcal{S}_j$, which has size $|S_j|$, will be. At this point, provided $2\omega_i^* < \varepsilon$, we can ask what is the minimum j such that $\mathcal{S}_j$, which has size S_j, would be such that $\xi_{\mathcal{S}_{i+1}} \leq \varepsilon$, under the assumption that $\mathsf{R}_{\mathcal{F}}(\mathcal{S}_j) \leq \omega_i^*$. Clearly if $2\omega_i^* \geq \varepsilon$, we have no hope of finding such a j.

Formally, if we let

$$\gamma_j = \ln\frac{3}{\delta} + (j)\ln 2,$$

we want to find the minimum j for which the following inequality holds:

$$\sqrt{\frac{\gamma_j}{2S_j}} + 2\frac{\gamma_j + \sqrt{\gamma_j(\gamma_j + 2S_j\omega_i^*)}}{S_j} + 2\omega_i^* \leq \varepsilon \tag{12}$$

The l.h.s. of this inequality is obtained from (4) using ω_i^* in place of $\mathsf{R}_{\mathcal{F}}(\mathcal{S})$, S_j in place of ℓ, $\delta/2^j$ in place of η , and slightly reorganize the terms for readability. Finding the minimum j for which (12) is satisfied can be done efficiently by first testing for gemetrically increasing values of j until the equation holds, and then perform a binary search.

Once j is available, the algorithm essentially sets i to j on line 33 in Alg. 1, and at the next iteration on line 9 samples $S_j - S_{i_{\mathrm{prev}}}$ transactions, where i_{prev} is the index of the previous iteration. In other words, it skips checking the sample size for samples at size S_r, $i_{\mathrm{prev}} < r < j$.

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) c_0 is initialized to $|R|$ (line 6);
(3) the updates to M, $\mathcal{V}$, $\mathbf{v}$, and $\mathbf{c}_\mathbf{v}$ (lines 16 to 24 included) are performed only for nodes $w \in R$.

We denote this variant as ABRA-s-set, and we will 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 and correctness analysis use *pseudodimension* [45], an extension of the Vapnik-Chervonenkis dimension to real-valued functions. A short introduction on pseudodimension can be found in Appendix C. 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 4.2 ([37]). *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*

$$|\mathcal{S}| = \frac{c}{\varepsilon^2}\left(d + \log\frac{1}{\eta}\right) \quad . \tag{13}$$

Then

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

The constant c is universal.

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 (13) 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 (13) 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 [48, 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 (7). 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 4.3. *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 C.1, 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

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

□

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 4.4. *We have:*

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

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. 4.3.

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. □

We comment that the upper bound derived in Lemma 4.4 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 (see Thm. 4.2 below) could be easily obtained using the union bound. We conjecture that it should be possible to obtain better bounds (see Conjecture 4.8.)

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 [20]. Riondato and Kornaropoulos [48, 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 [48] 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 [48], as shown in Thm. 4.5.

THEOREM 4.5. *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*

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

where f_w *is defined as in* (7)*, is at most* 3.

To prove Thm. 4.5 we show, in Lemma 4.6, that some subsets of $\mathcal{D} \times [0,1]$ can not be shattered by $\mathcal{F}^+$, on any graph G. Thm. 4.5 follows immediately from this result.

Lemma 4.6. *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 C.1 and C.2, 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 [48, 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

$$\sigma_{u_1v_1} = \sigma_{u_2v_2} = \sigma_{u_3v_3} = 1 \quad \text{and} \quad \sigma_{u_4v_4} = 2 \ .$$

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 $L = \{k_1, k_2, \ldots\} \subseteq \{1,2,3,4\}$ of indices, let $w_L = w_{k_1,k_2,\ldots}$ 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

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

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 $w_{h,4}$, $1 \leq h \leq 3$, must be on the same SP from u_4 to v_4, and all nodes in the form $w_{i,j,4}$, $1 \leq i < j \leq 3$ 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. 4.5 is tight, i.e., there exists a graph for which the pseudodimension is exactly 3 [48, Lemma 4]. Moreover, as soon as we relax the requirement in Thm. 4.5 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 4.7. *There is an undirected graph $G = (V, E)$ such that there is a set $\{(u_i, v_i), u_i, v_i \in V, u_i \neq v_i, 1 \leq i \leq 4\}$ with $|\mathsf{S}_{u_1,v_1}| = |\mathsf{S}_{u_2,v_2}| = 2$ and $|\mathsf{S}_{u_3,v_3}| = |\mathsf{S}_{u_4,v_4}| = 1$ that is shattered.*

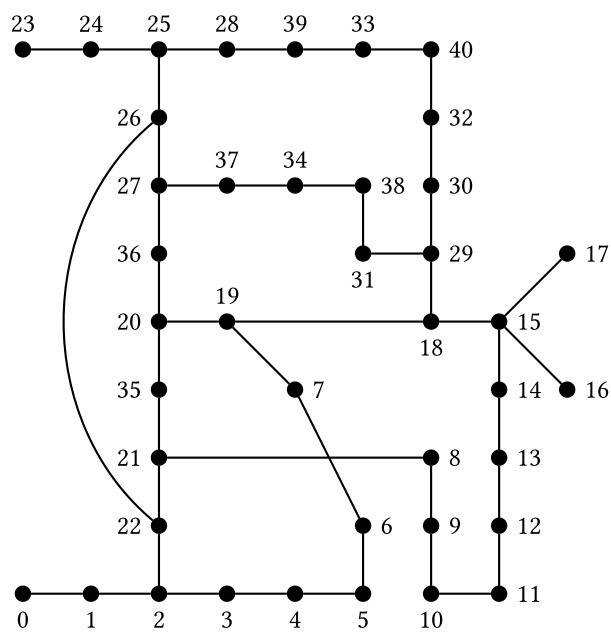


Fig. 3. Graph for Lemma 4.7

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

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

There is a single SP from 23 to 17:

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

There are exactly two SPs from 5 to 33:

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

There are exactly two SPs from 11 to 34:

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

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 Table 2. □

Table 2. How to shatter $Q = \{a, b, c, d\}$ from Lemma 4.7.

$P \subseteq Q$	Node v such that $P = Q \cap R_v$
$\emptyset$	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

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 [48, 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 4.6, and develop an additional stopping rule for ABRA-s based on the empirical pseudodimension.

Conjecture 4.8. *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*

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

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

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

4.3 Alternative Estimators

Geisberger et al. [20] 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 [15] 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 $1-\delta$ an 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. [20]. 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

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

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
$$\mathbf{v}' \leftarrow \mathbf{v} \cup \{(j, g_w(u,v,d))\} \enspace ;$$
- the definition of $\xi_{\mathcal{S}_i}$ on line 29 becomes
$$\xi_{\mathcal{S}_i} = 2\omega_i^* + \frac{2\gamma_i + \sqrt{2\gamma_i(2\gamma_i + 4\ell\omega_i^*)}}{2\ell} + 2\sqrt{\frac{\gamma_i}{2\ell}} \enspace .$$

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 $[0,b]$, as discussed in Appendix B ($b=2$ in this specific case.) 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 following 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. [20], 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 suboptimality 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. This is an interesting direction for future work.

4.4 Fixed Sample Size

In this section we introduce a variant 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 M, representing the number of samples (i.e., random pairs of nodes) that the algorithm will take. The algorithm will *always* perform M (and only M) 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 4.9. *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 [48], 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 $\mathsf{TOP}(k, G)$ 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. 4.11 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 $\mathsf{TOP}(k, G)$ 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:

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

The algorithm ABRA-s-k follows an approach similar to the one taken by the algorithm for the same task by Riondato and Kornaropoulos [48, 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

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

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 a *relative-error*

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

version of Thm. 3.3 (Thm. D.1 from Appendix D), 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:

$$\xi_{\mathcal{S}_i} = 2\omega_i^* + \frac{\theta^{-1}\ln(3/\eta) + \sqrt{(\theta^{-1}\ln(3/\eta) + 4\ell\omega_i^*)\theta^{-1}\ln(3/\eta)}}{2\ell} + \theta^{-1}\sqrt{\frac{\ln(3/\eta)}{2\ell}} \quad . \tag{14}$$

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 4.10. *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*

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

[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 $\xi_{\mathcal{S}_i}$ from (14) and applying Thm. D.1 from Appendix D instead of Thm. 3.3.

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

THEOREM 4.11. *With probability at least* $1 - \delta$*, the pair* $\left(\widetilde{\mathsf{TOP}}(k, G), \rho\right)$ *returned by* ABRA-s-k *is such that* $\rho \leq \varepsilon$ *and:*

(1) *for any pair* $(v, \mathrm{b}(v)) \in \mathsf{TOP}(k, G)$*, there is a pair* $\left(v, \widetilde{\mathrm{b}}(v)\right) \in \widetilde{\mathsf{TOP}}(k, G)$ *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* $\left(w, \widetilde{\mathrm{b}}(w)\right) \in \widetilde{\mathsf{TOP}}(k, G)$ *such that* $(w, \mathrm{b}(w)) \notin \mathsf{TOP}(k, G)$*, 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

$$\left|\widetilde{\mathrm{b}}(w) - \mathrm{b}(w)\right| \leq \xi, \text{ for all } w \in V \quad . \tag{15}$$

With probability at least $1 - \delta''$, we have, from the properties of ABRA-s-set-r in Thm. 4.10 that

$$\frac{\left|\widetilde{\mathrm{b}}(w) - \mathrm{b}(w)\right|}{\max\{y', \mathrm{b}(w)\}} \geq \rho, \text{ for all } w \in C \quad . \tag{16}$$

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

Consider the value y'. Since (15) holds, it is straightforward to see that $y' \leq b_k$. Thus, it also holds that all nodes appearing in $\mathsf{TOP}(k, G)$ belong to C, which may contain other nodes.

Since (16) 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 $\mathsf{TOP}(k, G)$ therefore has $\widetilde{\mathrm{b}}(w) \geq z(1 - \rho)$. It follows that all nodes appearing in $\mathsf{TOP}(k, G)$ will be in $\widetilde{\mathsf{TOP}}(k, G)$, satisfying the first part of (1) in the statement of the theorem.

The second part of (1) follows from (16), since for all nodes v appearing in $\mathsf{TOP}(k, G)$ it holds that $\mathrm{b}(v) \geq y'$.

Property (2) in the statement follows from (16), noticing that some of the nodes under consideration may have $\mathrm{b}(v) < y' \leq b_k$. □

5 DYNAMIC GRAPH BC APPROXIMATION

In this section we present an algorithm, named ABRA-d, that computes and keeps up to date an high-quality approximation of the BC of all nodes in a *fully dynamic graph*, i.e., in a graph where nodes and edges can be added or removed over time.

Our algorithm builds on the recent work by Hayashi et al. [24], who 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. [24] show how to populate and update 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. [24] is based on a static random sampling approach which is identical to the one described for ABRA-s, i.e., pairs of nodes are sampled and the BC estimation of the nodes along the SPs between the two nodes are updated as necessary. Their 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.

ABRA-d builds and improves over the algorithm presented by Hayashi et al. [24] as follows. Instead of using a static random sampling approach with a fixed sample size, it uses the progressive sampling approach and the stopping condition from ABRA-s to understand when we sampled enough to first populate the Hypergraph Sketch and the Two-Ball Index.

After each update to the graph, ABRA-d performs the same operations as in the algorithm by Hayashi et al. [24], with the crucial addition, after these operation have been performed, of keeping the set $\mathcal{V}_\mathcal{S}$ of vectors and the map M (already used in ABRA-s) up to date, and checking 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.

The overhead of additional checks of the stopping condition is minimal. On the other hand, the use of the progressive sampling scheme based on the Rademacher averages allows us to sample much fewer pairs of nodes than in the static sampling case based on the union bound: Riondato and Kornaropoulos [48] already showed that it is possible to sample much less than $O(\log|V|)$ nodes, and, as we show in our experiments, our sample sizes are even smaller than the ones by Riondato and Kornaropoulos [48]. The saving in the number of samples results in a huge speedup, as the running time of the algorithms are, in a first approximation, linear in the number of samples, and in a reduction in the amount of space required to store the data structures, 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 ABRA-d after each update has been processed is such that $\xi \leq \varepsilon$ and*

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

The proof follows from the correctness of the algorithm by Hayashi et al. [24] and of ABRA-s (Thm. 4.1).

6 EXPERIMENTAL EVALUATION

In this section we presents the results of our experimental evaluation.[7] 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 [14] and the approximation algorithm RK [48], which offers the same guarantees as ABRA-s (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-d in C++11, as an extension of the NetworKit library [53]. We use NLopt [27] 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$^{\text{TM}}$ II X4 955 processor and 16GB of RAM, running FreeBSD 12.

[7] We report the results for an older version of our algorithms. We are currently updating our implementation and re-running the experiments. We expect the results to not be significantly different.

Datasets and Parameters. We use graphs of various nature (communication, citations, P2P, and social networks) from the SNAP repository [36]. The characteristics of the graphs are reported in the leftmost column of Table 3.

In our experiments we varied ε in the range $[0.005, 0.3]$, and we also evaluate a number of different sampling schedules (see Sect. 6.2). 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. We performed five runs for each combination of parameters. The variance between the different runs was essentially insignificant, so we report, unless otherwise specified, the results for a random run.

6.1 Runtime and Speedup

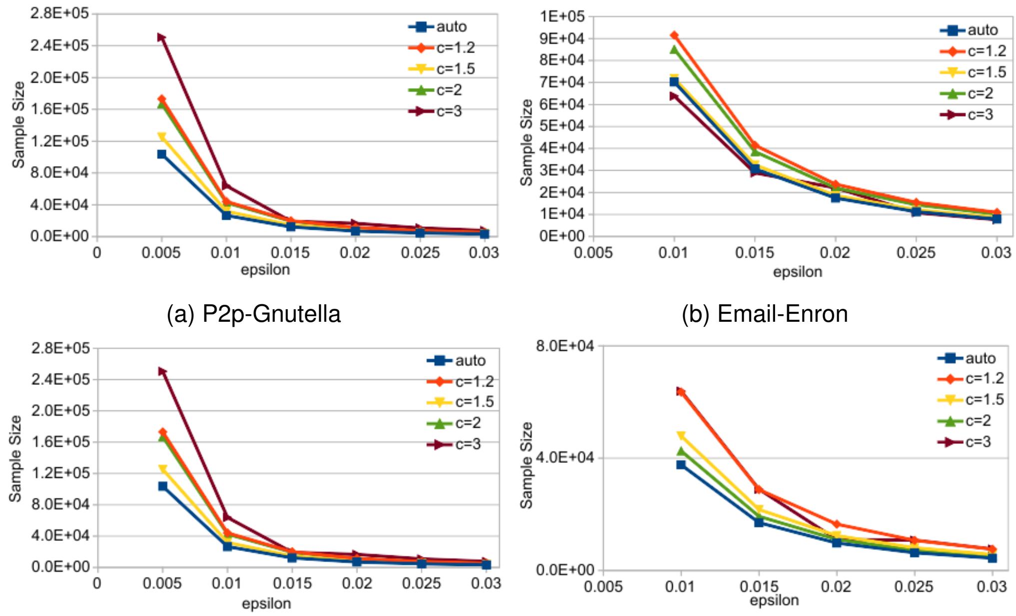
Our main goal was to develop an algorithm that can compute, with probability at least $1-\delta$, an ε-approximation of the BC of all nodes as fast as possible. Hence we evaluate the runtime and the speedup of ABRA-s w.r.t. BA and RK. The results are reported in columns 3 to 5 (from the left) of Table 3. The values for $\varepsilon = 0.005$ are missing for Email-Enron and Cit-HepPh because in these cases both RK and ABRA-s were slower than BA, a phenomena that, limited to RK, can be seen also in the first line of the results for soc-Epinions: in this case, RK is slower than BA, but ABRA-s is faster.

As expected, the runtime is a perfect linear function of the sample size (column 9), which in turns grows as ε^{-2}. The speedup w.r.t. the exact algorithm BA is significant and naturally decreases quadratically with ε. More interestingly ABRA-s is always faster than RK, sometimes by a significant factor. At first, one may think that this is due to the reduction in the sample size (column 10), but a deeper analysis shows that this is only one component of the speedup, which is almost always greater than the reduction in sample size. The other component can be explained by the fact that RK must perform an expensive computation (computing the vertex diameter [48] of the graph) to determine the sample size before it can start sampling, while ABRA-s can immediately start sampling and rely on the stopping condition, whose computation is inexpensive, as we will discuss. The different speedups for different graphs are due to different characteristics of the graphs: when the SP DAG between two nodes has many paths, ABRA-s does more work per sample than RK, which only backtracks along a single SP of the DAG, hence the speedup is smaller.

Runtime breakdown. The main challenge in designing a stopping condition for progressive sampling algorithm is striking the right balance between the strictness of the condition (i.e., it should stop early) and the efficiency in evaluating it. We now comment on the efficiency, and will report about the strictness in Sect. 6.2 and 6.3. In columns 6 to 8 of Table 3 we report the breakdown of the runtime into the main components. It is evident that evaluating the stopping condition amounts to an insignificant fraction of the runtime, and most of the time is spent in computing the samples (selection of nodes, execution of SP algorithm, update of the BC estimations). The amount in the "Other" column corresponds to time spent in logging and checking invariants. We can then say that our stopping condition is extremely efficient to evaluate, and ABRA-s is almost always doing "real" work to improve the estimation.

6.2 Sample Size and Sample Schedule

We evaluate the final sample size of ABRA-s and the performances of the "automatic" sample schedule (Sect. 4.1.2). The results are reported in columns 9 and 10 of Table 3. As expected, the sample size grows with ε^{-2}. We already commented on the fact that ABRA-s uses a sample size that is consistently (up to 4×) smaller than the one used by RK and how this is part of the reason why ABRA-s is much faster than RK. In Fig. 4 we show the behavior of the final sample size



(a) P2p-Gnutella

(b) Email-Enron

(c) Soc-Epinions1

(d) Cit-HepPh

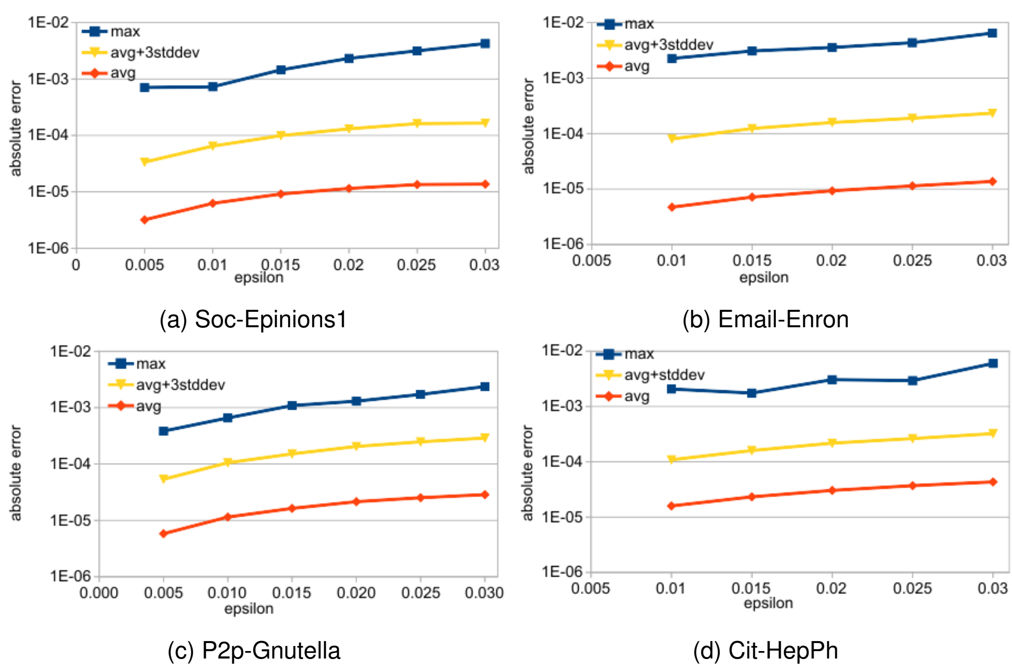
Fig. 4. Final sample size for different sample schedules.

chosen by the automatic sample schedule in comparison with *static geometric sample schedules*, i.e., schedules for which the sample size at iteration $i + 1$ is c times the size of the sample size at iteration i. We can see that the *automatic sample schedule is always better than the geometric ones*, sometimes significantly depending on the value of c (e.g., more than $2\times$ decrease w.r.t. using $c = 3$ for $\varepsilon = 0.05$). Effectively this means that the automatic sample schedule really frees the user from having to selecting a parameter whose impact on the performances of the algorithm may be devastating (larger final sample size implies higher runtime). Moreover, thanks to the automatic sample schedule, ABRA-s always terminates after just two iterations, while this was not the case for the geometric sample schedules (taking even 5 iterations in some cases): this means that effectively the automatic sample schedules "jumps" directly to a sample size for which the stopping condition will be verified. We can sum up the results and say that the stopping condition of ABRA-s stops at small sample sizes, smaller than those used in RK, and the automatic sample schedule we designed is extremely efficient at choosing the right successive sample size, to the point that ABRA-s only needs two iterations.

6.3 Accuracy

We evaluate the accuracy of ABRA-s by measuring the absolute error $|\widetilde{\mathrm{b}}(v) - \mathrm{b}(v)|$. The theoretical analysis guarantees that this quantity should be at most ε for all nodes, with probability at least $1 - \delta$. A first important result is that in *all* the thousands of runs of ABRA-s, the maximum error was *always* smaller than ε (not just with probability $> 1 - \delta$). We report statistics about the absolute error in the three rightmost columns of Table 3 and in Fig. 5. The minimum error (not reported) was always 0, so we do not report it in the table. The maximum error is *an order of magnitude smaller than* ε, and the average error is around *three orders of magnitude* smaller than ε, with a very small standard deviation. As expected, the error grows as ε^{-2}. In Fig. 5 we show the behavior of the maximum, average, and average plus three standard deviations (approximately corresponding to the 95% percentile), to appreciate how most of the errors are almost two orders of magnitude smaller than ε.

All 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 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.



(a) Soc-Epinions1

(b) Email-Enron

(c) P2p-Gnutella

(d) Cit-HepPh

Fig. 5. Absolute error evaluation. The vertical axis has a logarithmic scale.

6.4 Scalability

In this section we compare the scalability of ABRA-s to that of RK as the diameter of the graph grows. This choice is motivated by the fact that approximate betweenness estimation algorithms tends to all scale well as the number of nodes grows, because in many graph evolution models the diameter of the network tends to shrink as the number of nodes increases [35] (the one algorithm that does not scale well being the algorithm by Brandes and Pich [15] due to the fact that its sample size depends on the number of nodes in the graph.). On the other hand, RK is known to be susceptible to growth in the diameter, because its sample size depends on this quantity. We performed experiments to evaluate the resilience of ABRA-s to changes in the diameter. We created artificial graphs using the Email-Enron graph as the starting graph, then selecting a node v at random, and adding a *tail* (or chain) of k edges starting from v. If you think of the original graph as a balloon, you can think of the tail as the string to hold it. We used $k = 20, 40, 80, 160$. The results are presented in Table 4. 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.

Table 3. Runtime, speedup, breakdown of runtime, sample size, reduction, and absolute error.

			Speedup w.r.t.		Runtime Breakdown (%)					Absolute Error ($\times 10^5$)		
Graph	ε	Runtime (sec.)	BA	RK	Sampling	Stop Cond.	Other	Sample Size	Reduction w.r.t. RK	max	avg	stddev
Soc-Epinions1	0.005	569.44	1.15	2.46	99.978	0.021	0.002	134313	2.17	70.76	0.32	1.02
Directed	0.010	145.68	4.51	9.62	99.942	0.052	0.005	34456	8.48	72.80	0.63	1.95
$\lvert V\rvert = 75,879$	0.015	65.63	10.02	6.29	99.904	0.085	0.011	15463	4.72	144.90	0.92	3.00
$\lvert E\rvert = 508,837$	0.020	37.41	17.57	6.17	99.864	0.117	0.019	8857	3.66	231.93	1.15	3.98
	0.025	24.80	26.51	6.72	99.827	0.145	0.028	5842	3.13	315.32	1.35	4.95
	0.030	17.11	38.43	7.06	99.785	0.175	0.040	4036	2.90	423.53	1.37	5.06
P2p-Gnutella31	0.005	115.31	1.55	3.71	99.923	0.070	0.006	103637	3.20	29.87	0.52	1.43
Directed	0.010	29.19	6.11	14.65	99.781	0.196	0.023	26587	12.49	65.85	1.03	2.82
$\lvert V\rvert = 62,586$	0.015	13.47	13.25	7.99	99.663	0.291	0.046	12175	6.82	108.58	1.52	4.21
$\lvert E\rvert = 147,892$	0.020	7.66	23.29	6.27	99.563	0.362	0.075	6978	5.29	156.98	1.96	5.53
	0.025	5.10	34.98	5.40	99.489	0.403	0.108	4591	4.52	191.38	2.34	6.75
	0.030	3.63	49.21	3.44	99.409	0.442	0.149	3252	4.08	212.01	2.32	6.70
Email-Enron	0.010	204.43	1.17	1.09	99.975	0.022	0.002	70237	1.04	225.37	0.47	2.51
Undirected	0.015	88.82	2.70	2.51	99.953	0.043	0.005	30715	2.38	307.47	0.71	3.88
$\lvert V\rvert = 36,682$	0.020	50.57	4.74	1.96	99.928	0.065	0.008	17450	1.86	356.09	0.93	4.97
$\lvert E\rvert = 183,831$	0.025	32.41	7.40	1.73	99.899	0.090	0.011	11245	1.62	434.52	1.14	5.91
	0.030	22.78	10.53	1.56	99.870	0.114	0.016	7884	1.48	645.45	1.37	7.31
Cit-HepPh	0.010	246.90	2.07	1.93	99.948	0.049	0.003	37630	1.94	206.88	1.59	3.10
Undirected	0.015	111.37	4.58	4.29	99.904	0.090	0.006	17034	4.29	175.17	2.33	4.56
$\lvert V\rvert = 34,546$	0.020	63.72	8.01	3.33	99.859	0.131	0.010	9790	3.32	306.03	3.04	6.24
$\lvert E\rvert = 421,578$	0.025	41.37	12.33	2.89	99.815	0.172	0.014	6334	2.88	293.89	3.70	7.55
	0.030	28.76	17.74	2.66	99.773	0.209	0.018	4412	2.65	601.33	4.33	9.33

Table 4. Scalability comparison in the presence of long tails in the graph.

		Runtime (sec.)		Sample Size	
ε	Tail Length	ABRA-s	RK	ABRA-s	RK
0.005	20	780.89	914.67	275,084	332,104
	40	776.17	1027.48	272,698	372,104
	80	787.21	1140.89	276,540	412,104
	160	787.21	1254.28	275,044	452,104
0.010	20	198.26	228.98	70,149	83,026
	40	194.63	256.92	68,822	93,026
	80	198.38	284.96	69,970	103,026
	160	196.89	313.55	69,154	113,026
0.015	20	89.16	101.30	31,600	36,901
	40	88.30	114.27	31,367	41,345
	80	88.85	126.80	31,340	45,790
	160	89.28	139.45	31,556	50,234
0.020	20	48.20	57.10	17,178	20,757
	40	50.48	64.35	17,883	23,257
	80	49.56	71.40	17,528	25,757
	160	50.63	78.56	17,849	28,257
0.025	20	32.11	36.73	11,351	13,285
	40	32.13	41.11	11,412	14,885
	80	32.25	45.65	11,430	16,485
	160	31.17	50.05	11,038	18,085
0.030	20	22.77	25.45	8,056	9,226
	40	22.16	28.61	7,842	10,337
	80	22.83	31.70	8,105	11,448
	160	22.42	34.71	7,924	12,559

6.5 Dynamic BC Approximation

We did not evaluate ABRA-d experimentally, but, given its design, it is reasonable to expect that, when compared to previous contributions offering the same quality guarantees [8, 24], it would exhibit similar or even larger speedups and reductions in the sample size than what ABRA-s had w.r.t. RK. Indeed, the algorithm by Bergamini and Meyerhenke [7] uses RK as a building block and it needs to constantly keep track of (an upper bound on) the vertex diameter of the graph, a very expensive operation. On the other hand, the analysis of the sample size by Hayashi et al. [24] uses very loose simultaneous deviation bounds (the union bound). As already shown by Riondato and Kornaropoulos [48], the resulting sample size is extremely large and they already showed how RK can use a smaller sample size. Since we built over the work by Hayashi et al. [24] and ABRA-s improves over RK, we can reasonably expect it to have better performances than the algorithm by Hayashi et al. [24].

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 algorithms, including finding the top-k nodes with higher BC, using improved estimators, and special cases when there is a single SP. 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.

APPENDIX

A CORRECTNESS OF ABRA-S

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.

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

FACT A.1. *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.

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

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

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

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

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

LEMMA A.2. *Let $\eta \in (0, 1)$. Then*

$$\Pr\left(\exists i > 0 \ s.t. \sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi(\mathcal{S}_i, \gamma_{\eta,i})\right) \leq \delta \enspace . \tag{17}$$

The probability in (17) 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 A.2. From the union bound we have:

$$\Pr\left(\exists i > 0 \text{ s.t. } \sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi(\mathcal{S}_i, \gamma_{\eta,i})\right) \leq \sum_{i=1}^{\infty} \Pr\left(\sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \geq \xi(\mathcal{S}_i, \gamma_{\eta,i})\right) \ . \tag{18}$$

Since Fact A.1 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 (18) 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 A.3. *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 A.3. Lemma A.2 says that with probability at least $1-\delta$, the sequence $(\mathcal{S}_i)_{i\geq 1}$ is such that it holds

$$\sup_{w\in V} |\widetilde{\mathrm{b}}_{\mathcal{S}_i}(w) - \mathrm{b}(w)| \leq \xi(\mathcal{S}_i, \gamma_{\eta_i}), \text{ for all } i \geq 1 \ . \tag{19}$$

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 (19) 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 (19) 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 A.4. *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 Thm. A.3 and Fact A.4 we obtain the correctness of the vanilla version of ABRA-s, thus proving Thm. 4.1.

B IMPROVED BOUNDS

In this section we present the proof for Thm. 3.3, which simultaneously extends and fine-tunes [43, Thms. 3.11].

In the rest of this section, we let $\mathcal{F}$ be a family of functions from some domain $\mathcal{D}$ to $[0, b]$. The non-negativity of the members of $\mathcal{F}$ is crucial in many of the results we now present. Let $\mathcal{S} = \{s_1, \ldots, s_\ell\}$ be a collection of ℓ elements from $\mathcal{D}$, sampled independently.

We start with a few facts about the conditional Rademacher average $\mathsf{R}_\mathcal{F}(\mathcal{S})$.

Definition B.1 ([12]). A *non-negative* function f from a domain $\mathcal{X}^\ell$ to $\mathbb{R}$ is *c-self-bounding* if there exist functions g_i, $1 \leq fi \leq \ell$, from $\mathcal{X}^{\ell-1}$ to $\mathbb{R}$ such that, for all $(x_1, \ldots, x_n) \in \mathcal{X}^\ell$, *both* following conditions hold:

(1) $0 \leq f(x_1, \ldots, x_\ell) - g_i(x_1, x_{i-1}, x_{i+1}, \ldots, x_\ell) \leq c$ for all i, $1 \leq i \leq \ell$; and
(2) $\displaystyle\sum_{i=1}^{\ell}(f(x_1, \ldots, x_\ell) - g_i(x_1, x_{i-1}, x_{i+1}, \ldots, x_\ell) \leq f(x_1, \ldots, x_\ell)$.

The following specializes and finely tunes part of the proof of [43, Lemma 3.6] to functions with co-domain $[0, b]$.

Lemma B.2. *The conditional Rademacher average*

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

is a c-self-bounding function for $c = \frac{b}{2\ell}$.

Proof. The proof follows the same steps as the proof for [43, Lemma 3.6], with the additional observation that, for $\mathcal{F}$ satisfying the hypothesis, we have, for any i, $1 \leq i \leq \ell$,

$$\mathbb{E}_\lambda\left[\sup_{f\in\mathcal{F}} \frac{1}{\ell}\lambda_i f(s_i)\right] = \sup_{f\in\mathcal{F}}\frac{1}{\ell}\cdot\frac{1}{2}\cdot -1\cdot f(s_i) + \sup_{f\in\mathcal{F}}\frac{1}{\ell}\cdot\frac{1}{2}\cdot 1\cdot f(s_i) \leq 0 + \sup_{f\in\mathcal{F}}\frac{1}{\ell}\cdot\frac{1}{2}\cdot 1\cdot f(s_i) \leq \frac{b}{2\ell},$$

where the first inequality comes from the fact that $f(s_i) \geq 0$, hence $\sup_{f\in\mathcal{F}}\frac{1}{\ell}\cdot\frac{1}{2}\cdot -1\cdot f(s_i) \leq 0$. □

Theorem B.3 (Remark 3 after Thm. 2.1, 12). *Let Z be a c-self-bounding function. Then for any $t > 0$,*

$$\Pr\left(\mathbb{E}[Z] - Z \geq t\right) \leq \exp\left[-\frac{t^2}{2c\mathbb{E}[Z]}\right] \quad .$$

The following fact is immediate from the definition of conditional Rademacher average.

Fact B.4. *Define $\mathcal{F}^-$ as the family of functions containing a function $f^- = -f$ for every function $f \in \mathcal{F}$ (i.e., f^- is such that $f^-(x) = -f(x)$, for every $x \in \mathcal{D}$.) Then*

$$\mathsf{R}_{\mathcal{F}^-}(\mathcal{S}) = \mathsf{R}_\mathcal{F}(\mathcal{S}) \quad .$$

We now present some results about the supremum of the deviations $\sup_{f\in\mathcal{F}}(\mathsf{m}_\mathcal{S}(f) - \mathsf{m}_\mathcal{D}(f))$, and then prove Thm. 3.3.

Lemma B.5 ([31]). *We have*

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}}(\mathsf{m}_\mathcal{S}(f) - \mathsf{m}_\mathcal{D}(f))\right] \leq 2\mathbb{E}\left[\mathsf{R}_\mathcal{F}(\mathcal{S})\right] \quad .$$

Hence, using Fact B.4, we also also have that

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}}(\mathsf{m}_{\mathcal{D}}(f)-\mathsf{m}_{\mathcal{S}}(f))\right]=\mathbb{E}\left[\sup_{f^-\in\mathcal{F}^-}(\mathsf{m}_{\mathcal{S}}(f^-)-\mathsf{m}_{\mathcal{D}}(f^-))\right]\leq 2\mathbb{E}\left[\mathsf{R}_{\mathcal{F}}(\mathcal{S})\right] \quad . \tag{20}$$

Definition B.6 (Bounded differences inequality). Let $g:\mathcal{X}^\ell\to\mathbb{R}$ be a function of ℓ variables. The function g is said to satisfy the bounded difference inequality if and only if, for each i, $1\leq i\leq\ell$, there is a nonnegative constant c_i such that:

$$\sup_{\substack{x_1,\ldots,x_\ell\\ x_i'\in\mathcal{X}}}|g(x_1,\ldots,x_\ell)-g(x_1,\ldots,x_{i-1},x_i',x_{i+1},\ldots,x_\ell)|\leq c_i \quad . \tag{21}$$

Fact B.7. *The quantity*

$$g(\mathcal{S})=g(s_1,\ldots,s_\ell)=\sup_{f\in\mathcal{F}}(\mathsf{m}_{\mathcal{S}}(f)-\mathsf{m}_{\mathcal{D}}(f))$$

satisfies the bounded difference inequality with $c_i=\frac{b}{\ell}$, $1\leq i\leq\ell$.

The following concentration inequality is a classic result.

Theorem B.8 (McDiarmid [38]'s inequality). *Let* $g:\mathcal{X}^\ell\to\mathbb{R}$ *be a function satisfying the bounded difference inequality with constants* c_i, $1\leq i\leq\ell$. *Let* $x_1,\ldots,x_\ell$ *be* ℓ independent *random variables taking value in* $\mathcal{X}$. *Then we have*

$$\Pr\left(g(x_1,\ldots,x_\ell)-\mathbb{E}[g]>t\right)\leq e^{-2t^2/C},$$

where $C=\sum_{i=1}^\ell c_i^2$.

We now prove Thm. 3.3 for the more general case of functions in $[0,b]$. We can recover the results in the statement of Thm. 3.3 by setting $b=1$. The proof follows, for the most part, the same steps as in the proof for [43, Thm. 3.11].

of Thm. 3.3. Assume that the following three events all hold simultaneously:

- $E_1=$ "$\sup_{f\in\mathcal{F}}(\mathsf{m}_{\mathcal{S}}(f)-\mathsf{m}_{\mathcal{D}}(f))\leq\mathbb{E}\left[\sup_{f\in\mathcal{F}}(\mathsf{m}_{\mathcal{S}}(f)-\mathsf{m}_{\mathcal{D}}(f))\right]+b\sqrt{\frac{\ln(3/\eta)}{2\ell}}$";
- $E_2=$ "$\sup_{f\in\mathcal{F}}(\mathsf{m}_{\mathcal{D}}(f)-\mathsf{m}_{\mathcal{S}}(f))\leq\mathbb{E}\left[\sup_{f\in\mathcal{F}}(\mathsf{m}_{\mathcal{D}}(f)-\mathsf{m}_{\mathcal{S}}(f))\right]+b\sqrt{\frac{\ln(3/\eta)}{2\ell}}$";
- $E_3=$ "$\mathbb{E}[\mathsf{R}_{\mathcal{F}}(\mathcal{S})]\leq\mathsf{R}_{\mathcal{F}}(\mathcal{S})+\sqrt{\mathbb{E}[\mathsf{R}_{\mathcal{F}}(\mathcal{S})]\frac{b\ln(3/\eta)}{\ell}}$";

From Thms. B.3 and B.8, using also Lemma B.2 and Fact B.7, we have that each of the complementary events to E_1, E_2, and E_3 holds with probability at most $\eta/3$, hence the event $E_1\cap E_2\cap E_3$ holds with probability at least $1-\eta$. Assuming that that is the case, then it holds

$$\sup_{f\in\mathcal{F}}|\mathsf{m}_{\mathcal{S}}(f)-\mathsf{m}_{\mathcal{D}}(f)|\leq 2\mathbb{E}\left[\mathsf{R}_{\mathcal{F}}(\mathcal{S})\right]+b\sqrt{\frac{\ln(3/\eta)}{2\ell}} \quad . \tag{22}$$

In the rest of the proof, we show how to bound $\mathbb{E}\left[\mathsf{R}_{\mathcal{F}}(\mathcal{S})\right]$ using the sample-dependent quantity $\mathsf{R}_{\mathcal{F}}(\mathcal{S})$. Given that E_3 is verified, and, for any $\alpha>0$, it holds that

$$\sqrt{xy}\leq\frac{\alpha}{2}x+\frac{1}{2\alpha}y,$$

then we have

$$\mathbb{E}[\mathsf{R}_{\mathcal{F}}(\mathcal{S})]\leq\min_{\alpha\in(0,2)}\left[\frac{2\mathsf{R}_{\mathcal{F}}(\mathcal{S})}{2-\alpha}+\frac{b\ln(3/\eta)}{\ell\alpha(2-\alpha)}\right] \quad . \tag{23}$$

The minimum on the r.h.s. is attained for

$$\alpha^* = \frac{2b\ln(3/\eta)}{b\ln(3/\eta) + \sqrt{b\ln(3/\eta)(4\ell \mathsf{R}_{\mathcal{F}}(\mathcal{S}) + \ln(3/\eta))}}$$

We plug α^* into the argument of the min on the r.h.s. of (23), and then use the resulting bound to $\mathbb{E}[\mathsf{R}_{\mathcal{F}}(\mathcal{S})]$ in the r.h.s. of (22) to obtain the thesis:

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

□

C 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 [52] 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

$$R_f = \{(x,t) \;:\; t \le f(x)\} \enspace .$$

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

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

The *empirical pseudodimension* [45] of $\mathcal{F}$ on a subset $B \subseteq D$, denoted as $\mathsf{EPD}_{\mathcal{F}}(B)$, is the empirical VC-dimension of $\mathcal{F}^+$:

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

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

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

The following two technical lemmas are, to the best of our knowledge, new. We use them later to bound the pseudodimension of a family of functions related to betweenness centrality.

Lemma C.1. *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 C.2. *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. □

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

$$\sup_{f \in \mathcal{F}} \frac{|\mathsf{m}_\mathcal{D}(f) - \mathsf{m}_\mathcal{S}(f)|}{\max\{p, \mathsf{m}_\mathcal{D}(f)\}} \quad . \tag{24}$$

This quantity is inspired by the definition of *relative* (p, ε)*-approximations* [22].

Li et al. [37] used *pseudodimension* to study the quantity

$$\sup_{f \in \mathcal{F}} \frac{|\mathsf{m}_\mathcal{D}(f) - \mathsf{m}_\mathcal{S}(f)|}{\mathsf{m}_\mathcal{D}(f) + \mathsf{m}_\mathcal{S}(f) + p} \quad . \tag{25}$$

Har-Peled and Sharir [22] derived their concept of relative (p, ε)-approximation from this quantity and were only concerned with binary functions. The quantity in (25) has been studied often in the literature of statistical learning theory, see for example [3, Sect. 5.5], [11, Sect. 5.1], and [23], while other works (e.g., [11, Sect. 5.1], [16], [4], and [6]) focused on the quantity

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

We focus on the quantity in (24) because it is more useful in our specific case.

It is easy to see that

$$\sup_{f \in \mathcal{F}} \frac{|\mathsf{m}_\mathcal{D}(f) - \mathsf{m}_\mathcal{S}(f)|}{\max\{p, \mathsf{m}_\mathcal{D}(f)\}} \le \sup_{f \in \mathcal{F}} \frac{|\mathsf{m}_\mathcal{D}(f) - \mathsf{m}_\mathcal{S}(f)|}{p} \quad . \tag{26}$$

For any $f \in \mathcal{F}$, define $f_{/p}$ to be the function from $\mathcal{D}$ to $[0, 1/p]$ such that $f_{/p}(x) = f(x)/p$, for any $x \in \mathcal{D}$. Define the family

$$\mathcal{F}_{/p} = \{f_{/p}, f \in \mathcal{F}\} \quad .$$

We have from (26) that

$$\sup_{f \in \mathcal{F}} \frac{|\mathsf{m}_\mathcal{D}(f) - \mathsf{m}_\mathcal{S}(f)|}{\max\{p, \mathsf{m}_\mathcal{D}(f)\}} \le \sup_{f_{/p} \in \mathcal{F}_{/p}} |\mathsf{m}_\mathcal{D}(f_{/p}) - \mathsf{m}_\mathcal{S}(f_{/p})| \quad .$$

Therefore, a bound to the r.h.s. of this equation implies a bound to the quantity from (24) that we are interested in. We can use the results from Appendix B to obtain such a bound to the supremum of the absolute deviations of the sample means from their expectations for the functions in $\mathcal{F}_{/p}$. In particular, we can use the version of Thm. 3.3 for functions in $[0, b]$ that we proved in Appendix B to derive the following result for the quantity from (24), using $b = 1/p$.

Let $\mathcal{S}$ be a collection of ℓ elements of $\mathcal{D}$ sampled independently.

Theorem D.1. *Let $\eta \in (0,1)$. Then, with probability at least $1-\eta$,*

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

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] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. 2000. A sharp concentration inequality with application. *Random Structures & Algorithms* 16, 3 (2000), 277–292.

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

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

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

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

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

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

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

[28] 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.

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

[30] 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. DOI: https://doi.org/10.1109/9.895579

[31] Vladimir Koltchinskii and Dmitry Panchenko. 2002. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics* 30, 1 (2002), 1–50.

[32] 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.

[33] 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.

[34] 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.

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

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

[37] 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.

[38] Colin McDiarmid. 1989. On the method of bounded differences. *Surveys in combinatorics* 141, 1 (1989), 148–188.

[39] 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.

[40] 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.

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

[42] Terence R.F. Nonweiler. 1968. Algorithms: Algorithm 326: Roots of low-order polynomial equations. *Commun. ACM* 11, 4 (1968), 269–270.

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

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

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

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