Bavarian: Betweenness Centrality Approximation with Variance-Aware Rademacher Averages*

CYRUS COUSINS†, University of Massachusetts Amherst, USA
CHLOE WOHLGEMUTH, Amherst College, USA
MATTEO RIONDATO, Amherst College, USA

"[A]llain Gersten, Hopfen, und Wasser" — 1516 Reinheitsgebot

We present Bavarian, a collection of sampling-based algorithms for approximating the Betweenness Centrality (BC) of all vertices in a graph. Our algorithms use Monte-Carlo Empirical Rademacher Averages (MCERAs), a concept from statistical learning theory, to efficiently compute tight bounds on the maximum deviation of the estimates from the exact values. The MCERAs provide a sample-dependent approximation guarantee much stronger than the state of the art, thanks to its use of variance-aware probabilistic tail bounds. The flexibility of the MCERAs allows us to introduce a unifying framework that can be instantiated with existing sampling-based estimators of BC, thus allowing a fair comparison between them, decoupled from the sample-complexity results with which they were originally introduced. Additionally, we prove novel sample-complexity results showing that, for all estimators, the sample size sufficient to achieve a desired approximation guarantee depends on the vertex-diameter of the graph, an easy-to-bound characteristic quantity. We also show progressive-sampling algorithms and extensions to other centrality measures, such as percolation centrality. Our extensive experimental evaluation of Bavarian shows the improvement over the state-of-the art made possible by the MCERAs (2–4x reduction in the error bound), and it allows us to assess the different trade-offs between sample size and accuracy guarantees offered by the different estimators.

CCS Concepts: • **Mathematics of computing → Probabilistic algorithms**; • **Information systems → Social networks**; • **Theory of computation → Shortest paths**; **Dynamic graph algorithms**; **Sketching and sampling**; **Sample complexity and generalization bounds**; **Approximation algorithms analysis**.

Additional Key Words and Phrases: Concentration bounds, Dynamic graphs, Percolation centrality, Random sampling, Sample complexity, Statistical learning theory

ACM Reference Format:
Cyrus Cousins, Chloe Wohlgemuth, and Matteo Riondato. 2022. Bavarian: Betweenness Centrality Approximation with Variance-Aware Rademacher Averages. *ACM Trans. Knowl. Discov. Data.* 1, 1 (December 2022), 44 pages. https://doi.org/10.1145/3577021

1 INTRODUCTION

A centrality measure [12] assigns to each vertex or edge in a graph a score that quantifies the importance of that vertex/edge. Many measures and algorithms for them have been introduced in

*A preliminary version of this work [26] appeared in the proceedings of ACM KDD'21.
†Part of the work done while affiliated to Brown University, Dept. of Computer Science.

Authors' addresses: Cyrus Cousins, Manning College of Information & Computer Sciences, University of Massachusetts Amherst, 140 Governors Dr., Amherst, MA, 01003, USA, cbcousins@umass.edu; Chloe Wohlgemuth, Dept. of Computer Science, Amherst College, AC #2232 Amherst College, Amherst, MA, 01002, USA, cwohlgemuth22@amherst.edu; Matteo Riondato, Dept. of Computer Science, Amherst College, AC #2232 Amherst College, Amherst, MA, 01002, USA, mriondato@amherst.edu.

the literature, from simple ones like degree, to sophisticated ones based on shortest paths, such as closeness centrality [6] and betweenness centrality [3, 32], which is the focus of this work.

Informally, Betweenness Centrality (BC) quantifies the importance of a vertex/edge z as the fraction of all Shortest Paths (SPs) in the graph that go through z (see (1)). It is a measure of *robustness*, in addition to being a centrality measure [11]: if a vertex/edge with high BC is removed from the graph, then the SPs between many pairs of vertices will change, or even cease to exist. BC has many practical applications, from community detection [53], to the study of the resilience of the electrical grid [48], to genomics [34].

Computing the exact BC of every vertex/edge in a graph $G = (V, E)$ is quite expensive. Brandes's algorithm (BA) [18], the state of the art, takes time $O(|V||E|)$ if G is unweighted, and $O(|V||E| + |V|^2 \log|V|)$ otherwise. These running times are *too slow to be practical* on even moderately-sized networks. For this reason, soon after the exact algorithm had been introduced, *sampling-based approximation algorithms* were proposed [20, 39]. Approximate BC scores are *sufficient and acceptable* in practice when they come with *strong guarantees* on their quality, i.e., with an upper bound on the *maximum deviation* of any BC estimate from its unknown exact value (i.e., a bound on the maximum estimation error). Good estimates are particularly valuable when the graph evolves in a *fully-dynamic* way, that is, when edges and vertices are arbitrarily inserted and removed over time. In this setting, computing the *exact* BC of each vertex/edge is not only expensive, but also has *limited value*, as these scores continuously change. High-quality approximations are sufficient, and are easy to maintain after updates to the graph.

Sampling-based algorithms exhibit an inherent trade-off between the sample size and the quality guarantee: the larger the sample, the smaller the upper bound on the maximum deviation is, i.e., the better the guaranteed quality of the estimates. Improved characterization of this trade-off allows for stronger quality guarantees. Previous works studying the trade-off [13, 20, 21, 33, 60, 61] (see Sect. 2 for an in-depth discussion of these and other relevant contributions) also often introduce novel sampling-based *estimators* for BC, and limit the characterization of the trade-off to a single estimator, usually the one they propose. Different estimators draw samples from different populations (e.g., single vertices, pairs of vertices, or single SPs, see Sect. 3.2), so it is not possible to fairly compare the quality guarantees they offer, i.e, the different characterizations of the sample-size-vs.-accuracy-guarantee trade-off. While interesting from theoretical and practical points of view, this proliferation of estimators does not help users in choosing which estimator to employ, and which characterization gives the best "bang for the buck," especially as there may be even better characterizations.

Contributions. We present BAVARIAN, a suite of sampling-based algorithms for approximating the BC of all vertices in a large graph. Our contributions are the following.

- BAVARIAN is an *algorithmic framework* that can be instantiated with BC estimators (Sect. 3.2), enabling the study of their different computational and statistical trade-offs (Sect. 6). This unifying approach can be used for both static and progressive sampling algorithms (Sect. 5.2), and extended to variants of BC, to percolation centrality, and to dynamic graphs (Sect. 5.3).
- Our analysis of BAVARIAN uses Monte-Carlo Empirical Rademacher Averages (MCERAs) (Sect. 3.3), a key concept from statistical learning theory [67]. The variance-aware tail bounds (Thm. 3.1) for the MCERAs exploit the low variance of BC functions (Sect. 3.3), resulting in a tight characterization of the sample-size-vs.-accuracy trade-off, and thus in better quality guarantees at smaller sample sizes than previously possible.
- Our unifying framework, and the use of the MCERAs, allows us to show that, for every estimator, the sample size sufficient to (probabilistically) obtain a desired accuracy depends logarithmically on the *vertex diameter* (Sect. 5.1), a characteristic and easy-to-bound quantity of the graph. Previous estimator-specific bounds have logarithmic dependencies on the size

of the largest connected component [61], or on the number of vertices [20], which are much larger than the vertex diameter: in many realistic network models, the diameter (and the vertex-diameter) is proportional to the logarithm of these quantities. This result generalizes one by Riondato and Kornaropoulos [60, Coroll. 1] for the specific estimator they introduce. It is not just of theoretical interest: we use it to develop sample schedules for practical progressive sampling algorithms (Sect. 5.2).
- Our experimental evaluation (Sect. 6) shows a 2–4x improvement in the quality guarantee offered by BAVARIAN over the state-of-the-art, which is made possible by the use of the MCERAs. We also analyze the behavior of different estimators for BC, in the first fair comparison enabled by our unifying framework.

Some proofs are deferred to App. A.1, and additional experimental results are in App. A.2.
The present article extends the conference version [26] in multiple ways, including:

- we deeply streamlined the proofs of the sufficient sampling size, getting rid of a complex concept (SP-enforcing families) that is no longer needed, and instead using Jensen's inequality and other basic probability concepts to show the relationship between the Rademacher averages of the different estimators;
- we discuss a different approach to compute the k-MCERA, based on *matrix multiplication*, which reveals additional interesting properties of the BC estimators;
- we introduce a novel *adaptive-sampling* variant of the algorithm, which takes a best-effort approach in tuning the sample size, to obtain the desired quality guarantees as fast as possible;
- we show results on the relationships between the wimpy variances of the function families for the different estimators, and also on the variances of the estimators themselves, extending and improving results by Riondato and Kornaropoulos [60, Lemma 9];
- we perform an experimental evaluation of the progressive sampling algorithm, and report the results, showing that it frees the user from having to choose a fixed sample size in advance, so they can instead express their desired quality guarantees; and
- we present all the proofs of our theoretical results, after refining their hypotheses, to strengthen them as much as possible. We also give examples to make the most important concepts concrete.

2 RELATED WORK

We now discuss the contributions on BC most related to ours. BC was originally introduced in sociology [3, 32], and many variants have been developed since then [14, 19, 29, 45, 52, 54, 56]. In this work, we focus on vertex BC, but the versatility of our approach allows us to tackle other variants (see Sect. 5.3).

The first efficient algorithm for BC [18], known as Brandes' algorithm (BA), uses an ingenious recursive formulation to obtain the BC of all vertices in time proportional to n Single-Source-Shortest-Paths (SSSP) computations in an n-vertex graph. The time complexity on unweighted graphs is $O(n^2 + nm)$ where m is the number of edges in the graph, and becomes $O(nm + n^2 \log n)$ on weighted graphs. Despite the remarkable efforts to make this algorithm more scalable in practice [30, 59, 62], its cost is still excessive even for non-humongous graphs. Approximation algorithms [9, 10, 13, 20, 21, 33, 39, 51, 60, 61] and heuristics [4, 31, 49, 50] propose to address this shortcoming. We focus here on works that offer *guarantees on their output* (see also Sect. 3.2). We refer the reader to [60, Sect. 2] and [12] for an in-depth discussion of the heuristics.

All approximation algorithms for BC are based on random sampling, but they draw samples from different populations according to different distributions, compute the approximations using different estimators, and their authors use different approaches to analyze the quality guarantees

they offer. Our work has two goals: *(1)* present BAVARIAN, a unifying framework that can be instantiated with all these estimators, thus allowing a fair comparison between them; and *(2)* use the MCERAs to obtain tighter quality guarantees and better dependency on graph properties for all estimators.

Brandes and Pich [20] and independently Jacob et al. [39] present the first sampling-based approximation algorithm for computing high-quality estimates of the BC of all nodes. Their estimator is closely related to the inner workings of BA (see Sect. 3.2), and the analysis of the sample size vs. quality trade-off uses Hoeffding's bound [38] and the union bound, thus it is quite loose. Geisberger et al. [33] present a slight refinement of the estimator, but the sample size is unchanged. Chehreghani et al. [21] present a sampling-based algorithm for estimating the BC of a *single* node. Extending the guarantee to all nodes via union bound results in the same sample size as the algorithm by Jacob et al. [39] and Brandes and Pich [20].

Riondato and Kornaropoulos [60] propose a new estimator (see Sect. 3.2) and an approximation algorithm whose sample size is obtained using an upper bound to the Vapnik-Chervonenkis dimension [68] of the problem. This quantity depends on the vertex-diameter of the graph (the largest number of vertices on a shortest path). The resulting sample size is therefore much smaller than the one derived by previous works, while still being the result of a worst-case analysis. Bergamini and Meyerhenke [9] show how to better approximate the vertex-diameter on directed networks, but they use the same sample size, as does, in the worst case, the progressive-sampling algorithm by Borassi and Natale [13].

Riondato and Upfal [61] introduce another novel estimator (see Sect. 3.2) and an approximation algorithm whose analysis uses Rademacher averages [43], another core concept of statistical learning theory. Rademacher averages more tightly characterize the trade-off between sample size and approximation quality using only *sample-dependent* quantities, rather than relying on *graph-dependent* (i.e., population-dependent) properties. As a result, the same approximation quality can be obtained with much smaller samples than before. Our work does not use *deterministic* upper bounds to the Rademacher averages. Rather, we leverage Monte-Carlo estimation (see (7)) and variance-aware tight deviation bounds (see Thm. 3.1), to achieve both aforementioned goals.

Recently, Fan et al. [31] used graph neural networks to approximate the top-k nodes with highest BC, but without guarantees on the output. As argued in Sect. 1, quality guarantees are important, if not necessary, to make approximate solutions acceptable.

Modern graphs often arise from *dynamic processes*, in which vertices and edges are continuously added and/or removed. Many works look at how to compute and update BC, either exactly or approximately, after an update or batch of updates [9, 10, 35, 37, 41, 46, 58, 61, 69]. We discuss how to adapt our results to the fully-dynamic case in Sect. 5.3.

The MCERA has received relatively little attention until now, despite being defined, in a less general form, a while ago [5]. De Stefani and Upfal [28] use it for adaptive data analysis, Cousins and Riondato [25] present refined convegence bounds for it, and Pellegrina et al. [55] use it for pattern mining. All these scenarios and results are very different from the one we consider, and the approaches taken by these works cannot be applied for betweenness centrality approximation.

3 PRELIMINARIES

We now state definitions and theorems used throughout the work. For the reader's convenience, we report the most used notation in Table 1.

3.1 Betweenness Centrality

Let $G = (V, E)$ be a graph. The edges may be directed or undirected and may have non-negative weights. We denote with $n = |V|$ the number of vertices in G. For ease of discussion, we assume

Table 1. Most important notation and abbreviations used in the paper.

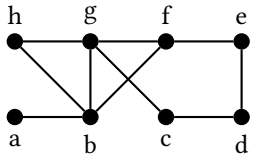
Symbol	Name / Meaning
$G = (V, E)$	Graph with $\lvert V\rvert = n$ vertices
$p = (u, z_1, \ldots, z_{\lvert p\rvert-2}, v)$	Path from vertex u to vertex v; the vertices in this ordered sequence are all distinct
$\mathsf{int}(p)$	Set of nodes *internal* to a path p, i.e., $\{z_1, \ldots, z_{\lvert p\rvert-2}\}$
Γ_{uv}	Set of Shortest Paths (SPs) from u to v
σ_{uv}	Number of SPs from u to v, i.e., $\lvert\Gamma_{uv}\rvert$
$\mathsf{b}(w)$	Betweenness Centrality (BC) of vertex $w \in V$
A	Generic approach for estimating the BC through sampling
$\mathcal{D}_{\mathsf{A},G}$	G-dependent domain from which A draws samples
π_{A}	Probability distribution over $\mathcal{D}_{\mathsf{A},G}$ from which A draws samples
m	Sample size
$\mathcal{S}$	Collection of m independent samples in $\mathcal{D}_{\mathsf{A},G}$ drawn from π_{A}
$\tilde{\mathsf{b}}_{\mathsf{A},\mathcal{S}}(w)$	Estimate of the BC of vertex/edge w using A on $\mathcal{S}$ (see (2))
$\mathcal{F}$	Family of functions from a domain $\mathcal{D}$ to [0,1]
$\mathsf{SD}(\mathcal{F}, \mathcal{S})$	Supremum Deviation (SD) of $\mathcal{F}$ on $\mathcal{S}$ (see (6))
k	Number of Monte-Carlo trials
$\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_m\}$	Bag of m i.i.d. k-dimensional Rademacher vectors
$\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda})$	k-trials Monte-Carlo Empirical Rademacher Average (k-MCERA) of $\mathcal{F}$ on $\mathcal{S}$ using $\mathbf{\Lambda}$ (see (7))
v, β	(Raw) Wimpy variance of $\mathcal{F}$ and empirical (raw) wimpy variance of $\mathcal{F}$ (see Sect. 3.3)

the graph to be readily available, so that operations such as sampling a vertex or a pair of vertices uniformly at random is easy. When this assumption does not hold, one can use appropriate methods to draw these samples [22, 23, 42]. We define a path p from a vertex u to a vertex v as an ordered sequence of *distinct* vertices $(u, z_1, \ldots, z_{\lvert p\rvert-2}, v)$. Given a path p between vertices u and v, a vertex w is *internal* to p if and only if $w \neq u$, $w \neq v$, and $w \in p$. We denote the set of vertices internal to a path p as $\mathsf{int}(p)$. The *length* of a path p is $\lvert p\rvert - 1$ if the edges do not have weights and is the sum of the weights of the edges $(u, z_1), (z_1, z_2), \ldots, (z_{\lvert p\rvert-2}, v)$ otherwise, i.e., an unweighted graph is essentially a weighted graph with all edge-weights equal to one. Any path p from u to v that has minimal length among all the paths from u to v is known as a *Shortest Path (SP)*. For any *ordered* pair of distinct vertices (u, v), let Γ_{uv} denote the set of SPs from u to v. To avoid having to repeatedly mention how we handle the special case of v being unreachable from u (i.e., when there is no path, and therefore no SP, between them), we say that in this case Γ_{uv} contains only a special *empty path* $p_{u,v} = (u, v)$, i.e., a "path" with no internal vertices. We define $\sigma_{uv} \doteq \lvert\Gamma_{uv}\rvert$, and we use $\sigma_{uv}(w)$ to denote the number of SPs from u to v that w is internal to.

The *Betweenness Centrality (BC)* $\mathsf{b}(w)$ *of a vertex* $w \in V$ is [3, 32]

$$\mathsf{b}(w) \doteq \frac{1}{n(n-1)} \sum_{\substack{(u,v)\in V\times V \\ u\neq v}} \frac{\sigma_{uv}(w)}{\sigma_{uv}} \quad (\in [0,1]) \ . \tag{1}$$

These and other notations are summarized in Table 1. An example graph, with associated BC values, from [60, Fig. 1], is shown in Fig. 1.



Vertex x	a	b	c	d	e	f	g	h
$\mathsf{b}(x)$	0	0.250	0.107	0.036	0.071	0.178	0.250	0

Fig. 1. Example of BC from [60, Fig. 1].

There exist many variants of BC [14, 19, 29, 45, 52, 54, 56]. For ease of presentation, we focus on the one for vertices, but our results can be extended to many of these variants (see Sect. 5.3).

Computing the exact BCs for all vertices is expensive [18] (see also Sect. 2). In this work, we are interested in *estimating the BCs* of all vertices, using different "approaches" (see Sect. 3.2) that offer *probabilistic guarantees* on the quality of the estimates that they compute. All approaches rely on *random sampling*: given the graph G, a user-defined *sample size* $m > 0$ and a user-defined *acceptable failure probability* $\delta \in (0,1)$, approach A creates a *sample* $\mathcal{S} = \{x_1, \ldots, x_m\}$ by drawing m *independent* samples from an approach-specific, G-dependent, population $\mathcal{D}_{\mathsf{A},G}$ according to an approach-specific distribution π_{A} over $\mathcal{D}_{\mathsf{A},G}$ (for ease of notation, we will often drop G when it is clear from the context, and just use $\mathcal{D}_{\mathsf{A}}$). A uses $\mathcal{S}$ to compute the estimate $\tilde{\mathsf{b}}_{\mathsf{A},\mathcal{S}}(w)$ for every vertex w (for ease of notation we will often drop $\mathcal{S}$ when it is clear from the context, and just use $\tilde{\mathsf{b}}_{\mathsf{A}}(w)$). For any approach A we consider, and any $w \in V$, the estimate $\tilde{\mathsf{b}}_{\mathsf{A},\mathcal{S}}(w)$, is the *sample mean* over $\mathcal{S}$ of a function f_w defined over $\mathcal{D}_{\mathsf{A},G}$, i.e.,

$$\tilde{\mathsf{b}}_{\mathsf{A},\mathcal{S}}(w) = \frac{1}{m} \sum_{x\in\mathcal{S}} f_w(x), \tag{2}$$

which is *unbiased*, i.e., its expectation w.r.t. the choice of $\mathcal{S}$ is the exact BC $\mathsf{b}(w)$. Also, for any approach A and any sample size $m > 0$, there is a function $\mathsf{eps}_{\mathsf{A},m}(\cdot)$ from $\mathcal{D}_{\mathsf{A}}^m \times (0,1)$ to the non-negative reals $\mathbb{R}_+$ such that, with probability at least $1-\delta$ over the runs of A with the same inputs G, m, and δ, all estimates are within $\mathsf{eps}_{\mathsf{A},m}(\mathcal{S},\delta)$ from their exact value. In other words, it holds

$$\Pr\left(\exists w \in V \text{ s.t. } \left|\mathsf{b}(w) - \tilde{\mathsf{b}}_{\mathsf{A},\mathcal{S}}(w)\right| > \mathsf{eps}_{\mathsf{A},m}(\mathcal{S},\delta)\right) < \delta \ . \tag{3}$$

One of our goals is to compare $\mathsf{eps}_{\mathsf{A}_1,m}(\cdot)$, $\mathsf{eps}_{\mathsf{A}_2,m}(\cdot)$, ..., for different approaches A_1, A_2, ..., as they give different characterizations of the trade-off between sample size and accuracy guarantees.

3.2 BC Estimation

We now review existing approaches for BC estimation. For each approach A, we define a domain $\mathcal{D}_{\mathsf{A}}$, a family $\mathcal{F}_{\mathsf{A}}$ of functions from $\mathcal{D}_{\mathsf{A}}$ to $[0,1]$, and a probability distribution π_{A} over $\mathcal{D}_{\mathsf{A}}$.

The RK estimator. Riondato and Kornaropoulos [60] introduce a BC estimator that is tailored to their use of VC-dimension [68] for the analysis of the sample size sufficient to obtain a high-quality estimate of the BC of all nodes. The domain $\mathcal{D}_{\mathsf{rk}}$ is the set of all shortest paths between all pairs of vertices in the graph G, i.e.,

$$\mathcal{D}_{\mathsf{rk}} \doteq \bigcup_{\substack{(u,v)\in V\times V \\ u\neq v}} \Gamma_{uv} \ .$$

Let $p_{uv} \in \mathcal{D}_{\mathsf{rk}}$ be any SP from u to $v \neq u$. The distribution π_{rk} over $\mathcal{D}_{\mathsf{rk}}$ assigns to p_{uv} the probability mass

$$\pi_{\mathsf{rk}}(p_{uv}) \doteq \frac{1}{n(n-1)\sigma_{uv}} \ .$$

Riondato and Kornaropoulos [60] show an efficient sampling scheme to draw independent samples from $\mathcal{D}_{\mathsf{rk}}$ according to π_{rk}. The cost of drawing a sample is essentially that of a truncated SSSP computation from u to v. The family $\mathcal{F}_{\mathsf{rk}}$ of functions contains one function $f_w : \mathcal{D}_{\mathsf{rk}} \to \{0,1\}$ for each vertex w, defined as

$$f_w(p) \doteq \mathbb{1}_{\mathsf{int}(p)}(w) \doteq \begin{cases} 1 & \text{if } w \in \mathsf{int}(p) \\ 0 & \text{otherwise} \end{cases} \ .$$

The same estimator is used by Borassi and Natale [13] for a progressive sampling algorithm for BC estimation.

The ABRA estimator. The ABRA algorithm [61] uses an estimator defined over the domain

$$\mathcal{D}_{\mathsf{ab}} \doteq \{(u,v) \in V \times V \ : \ u \neq v\},$$

i.e., over all pairs of distinct vertices. The distribution π_{ab} is uniform over $\mathcal{D}_{\mathsf{ab}}$; thus sampling from it is easy, per our assumptions. The family $\mathcal{F}_{\mathsf{ab}}$ contains a function f_w for each vertex w, defined as

$$f_w(u,v) \doteq \frac{\sigma_{uv}(w)}{\sigma_{uv}} \in [0,1] \ . \tag{4}$$

Given $(u,v) \in \mathcal{D}_{\mathsf{ab}}$, one can compute the value of $f_w(u,v)$ for each w in time proportional to performing a truncated SSSP from u to v.

The BP estimator. Jacob et al. [39], and independently Brandes and Pich [20] introduce a BC estimator that is closely related to how Brandes [18]'s algorithm computes the exact BC of all nodes. The domain $\mathcal{D}_{\mathsf{bp}}$ is the set V of vertices in G, and the distribution π_{bp} is uniform over this set. The family $\mathcal{F}_{\mathsf{bp}}$ contains one function f_w for each vertex w, defined as

$$f_w(v) \doteq \frac{1}{n-1} \sum_{z\neq v} \frac{\sigma_{vz}(w)}{\sigma_{vz}} \in [0,1] \ . \tag{5}$$

The value $f_w(v)$ can be computed by performing a (full) SSSP computation from v, and then backtracking along the resulting SP DAG as in the exact BC algorithm BA by Brandes [18]. Geisberger et al. [33] present a variant of the BP estimator to ameliorate some of its issues. The theoretical guarantees of this variant and its computation are essentially the same as the BP estimator, and the same observations we make for BP can be extended to this variant.

3.3 Bounding the Supremum Deviation

We now define the Supremum Deviation (SD), the Monte-Carlo Empirical Rademacher Average (MCERA), and the wimpy variance, and present important results using them. We tailor the definitions to our settings, and refer to the books by Shalev-Shwartz and Ben-David [63, Ch. 26] and Boucheron et al. [16, Ch. 11-13] for in-depth discussions.

Let $\mathcal{F}$ be a family of functions from a domain $\mathcal{D}$ to $[0,1]$.[1] Let π be a distribution over $\mathcal{D}$, and m be a positive integer. Given a bag (sample) $\mathcal{S} = \{x_1, \ldots, x_m\}$ of m samples from $\mathcal{D}$ drawn independently according to π, the *Supremum Deviation (SD) of $\mathcal{F}$ on $\mathcal{S}$* is defined as

$$\mathsf{SD}(\mathcal{F}, \mathcal{S}) \doteq \sup_{f \in \mathcal{F}} \left| \mathbb{E}_\pi [f] - \frac{1}{m} \sum_{x \in \mathcal{S}} f(x) \right| . \tag{6}$$

The SD is the key concept in the study of empirical processes [57]. Sample-dependent quantities, such as the popular Empirical Rademacher average (ERA) [43], can be used to derive probabilistic upper bounds to the SD. In this work, we use a Monte-Carlo estimator of the ERA, first introduced by Bartlett and Mendelson [5]. For $k \geq 1$, let $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_m\}$ be a bag of m i.i.d. k-dimensional Rademacher vectors, i.e., vectors whose entries are drawn independently and uniformly from $\{-1, 1\}$. The *k-Trials Monte-Carlo Empirical Rademacher Average (k-MCERA) $\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda})$ of $\mathcal{F}$ on $\mathcal{S}$ using $\mathbf{\Lambda}$* is

$$\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda}) \doteq \frac{1}{k} \sum_{j=1}^{k} \left(\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{x_i \in \mathcal{S}} \boldsymbol{\lambda}_{i,j} f(x_i) \right) . \tag{7}$$

The expectation of the k-MCERA w.r.t. $\mathbf{\Lambda}$ is known as the *Empirical Rademacher Average* $\hat{\mathsf{R}}_m(\mathcal{F}, \mathcal{S})$, and the expectation w.r.t. both $\mathcal{S}$ and $\mathbf{\Lambda}$ is known as the *Rademacher Average* $\mathsf{R}_m(\mathcal{F}, \pi)$, i.e., [43]

$$\hat{\mathsf{R}}_m(\mathcal{F}, \mathcal{S}) \doteq \mathbb{E}_{\mathcal{S}} \left[\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda}) \right], \qquad \mathsf{R}_m(\mathcal{F}, \pi) \doteq \mathbb{E}_{\mathcal{S}, \mathbf{\Lambda}} \left[\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda}) \right] . \tag{8}$$

These quantities are a cornerstone of statistical learning theory. While Rademacher averages control the *expected SD*, the sharpest probabilistic bounds currently available use the k-MCERA and depend on the *variances* of the functions in $\mathcal{F}$. This fact is natural, since for each $f \in \mathcal{F}$, the *variance* $\mathbb{V}[f]$ controls $\mathsf{SD}(\{f\}, \mathcal{S})$, asymptotically through the Central Limit Theorem, or via Bennett's inequality [7] with finite sample guarantees. The *maximum* variance of a function $f \in \mathcal{F}$ would therefore control $\mathsf{SD}(\mathcal{F}, \mathcal{S})$. BC values are typically very small on large graphs, thus, for f in any of the families from Sect. 3.2, it is reasonable to assume that $\mathbb{V}[f] \doteq \mathbb{E}_\pi \left[f^2\right] - (\mathbb{E}_\pi [f])^2 \approx \mathbb{E}_\pi \left[f^2\right]$, which is the *raw variance* of f; henceforth all variances are raw unless otherwise noted. We define the (raw) *wimpy* variance v, and its estimator β, as

$$\mathsf{v} \doteq \sup_{f \in \mathcal{F}} \mathbb{E}_\pi \left[f^2\right] \text{ and } \beta \doteq \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{x_i \in \mathcal{S}} (f(x_i))^2 . \tag{9}$$

Clearly both v and β depend on $\mathcal{F}$, and the families introduced in Sect. 3.2 have different wimpy variances, as we show in Thm. 4.1.

The following result (proof in App. A.1) shows how to use the k-MCERA to compute an upper bound to the SD using only sample-dependent quantities. The reader is invited to compare this result to [25, Thm. 4], wherein the authors derive a similar bound involving *centralized* variances and Rademacher averages. We eschew such an approach here, as when all betweenness centrality values are very small, which is a common case, centralization greatly complicates the analysis and yields larger constant-factors, with negligible benefit.

[1] The results can be generalized to a generic co-domain $[a, b] \subset \mathbb{R}$, but all methods from Sect. 3.2 have families mapping to $[0, 1]$, so we restrict to $a = 0$ and $b = 1$.

Theorem 3.1. *Let $\eta \in (0,1)$, and define the quantities*

$$\gamma \doteq \beta + \frac{2\ln\frac{5}{\eta}}{3m} + \sqrt{\left(\frac{\ln\frac{5}{\eta}}{\sqrt{3}m}\right)^2 + \frac{2\beta\ln\frac{5}{\eta}}{m}}, \tag{10}$$

$$\rho \doteq \hat{\mathsf{R}}^k_m(\mathcal{F},\mathcal{S},\mathbf{\Lambda}) + \frac{2\ln\frac{5}{\eta}}{3km} + \sqrt{\frac{4\beta\ln\frac{5}{\eta}}{km}},$$

$$r \doteq \rho + \frac{\ln\frac{5}{\eta}}{3m} + \sqrt{\left(\frac{\ln\frac{5}{\eta}}{2\sqrt{3}m}\right)^2 + \frac{\rho\ln\frac{5}{\eta}}{m}}\ ,$$

$$\varepsilon \doteq 2r + \frac{\ln\frac{5}{\eta}}{3m} + \sqrt{\frac{2\,(\gamma+4r)\ln\frac{5}{\eta}}{m}}\ , \tag{11}$$

Then, with probability at least $1-\eta$ over the choice of $\mathcal{S}$ and $\mathbf{\Lambda}$, it holds that

$$\mathsf{SD}(\mathcal{F},\mathcal{S}) \le \varepsilon\ .$$

As can be seen from the proof, this result consolidates several probabilistic upper bounds:, with probability at least $1-\eta$, it holds simultaneously that $\mathsf{v} \le \gamma$,[2] $\hat{\mathsf{R}}_m(\mathcal{F},\mathcal{S}) \le \rho$, and $\mathsf{R}_m(\mathcal{F},\pi) \le r$. The apparent complexity of this result is due to the fact that it better characterizes the trade-off between the sample size (and other properties of the sample) and the quality of the estimates, by leveraging the *empirical wimpy variance*, i.e., the quantity β in (9). By looking at prior work under the lens of variance, one can see that the Hoeffding+union bound and the VC-dimension theorem, used respectively by Brandes and Pich [20] and Riondato and Kornaropoulos [60], assume the worst-case possible variance, while the bound used by Riondato and Upfal [61] improves to an exponential average over variances. These counting-based approaches are not sensitive to any correlations between the quantities being estimated, whereas the k-MCERA is. This advantage is one of the reasons why we choose to use this concept: this sensitivity is relevant to BC estimation, as the scores of nearby vertices are strongly correlated, thus one should leverage these correlations when computing the approximation quality guarantee. It holds

$$\varepsilon \in 2\hat{\mathsf{R}}^k_m(\mathcal{F},\mathcal{S},\mathbf{\Lambda}) + \mathrm{O}\left(\frac{\ln\frac{1}{\eta}}{m} + \sqrt{\frac{(\mathsf{v}+\mathsf{R}_m(\mathcal{F},\pi))\ln\frac{1}{\eta}}{m}}\right)\ .$$

Each f_w used for BC estimation has codomain $[0,1]$, so $\mathsf{v} = \max_{w\in V}\mathbb{E}_\pi\left[f_w^2\right] \le \max_{w\in V}\mathbb{E}_\pi\left[f_w\right] = \max_{w\in V}\mathsf{b}(w)$ (more refined bounds are discussed in Sect. 4.2). Thus, ε is, in some sense, "output-sensitive," as it depends on the maximum BC.

As $\mathcal{F}$ is fixed, the quantity on the r.h.s. of (11), is a function $\mathsf{g}_{\mathcal{F}}(\mathcal{S},\eta)$ depending *only* on $\mathcal{S}$ and on η. For every approach A from Sect. 3.2, we can define the function $\mathsf{eps}_{\mathsf{A},m}(\cdot)$ used in (3) as

$$\mathsf{eps}_{\mathsf{A},m}(\mathcal{S},\delta) = \mathsf{g}_{\mathcal{F}_\mathsf{A}}(\mathcal{S},\delta)\ .$$

4 STATISTICAL PROPERTIES OF BC ESTIMATORS

In this section, we show that the seemingly-disparate rk, ab, and bp estimators can be directly contrasted in terms of their statistical properties, rather precisely via the *wimpy variances* and *Rademacher averages* of the family of functions associated to them: we show that the raw variance

[2]Since we are considering functions with co-domain $[0,1]$, Popoviciu's inequality allows us to use $\gamma = \frac{1}{4}$ when the r.h.s. of (10) is larger than $\frac{1}{4}$. In general, any deterministic bound to the wimpy variance that is better than the r.h.s. of (10) can be used. For ease of presentation, we do not further mention this fact.

(and thus also centralized variance), the wimpy variance, as well as the Rademacher averages, of the rk, ab, and bp estimators obey a decreasing relationship. In Sect. 4.2 we discuss the variance of the three estimators.

Our starting point is the observation that the three estimators are equal (to the exact BC) in expectation, and each simply commutes progressively less randomness from the inner sample to the outer expectations. Each sampling algorithm may therefore be seen as a progressively more random stochastic approximation of the exact algorithm. Intuitively, consider that, when run with $m = 1$ samples, the relationship between these estimators can be expressed as

$$\mathsf{b}(w) = \mathbb{E}_{u \in V}\Big[\underbrace{\mathbb{E}_{v \neq u}\Big[\overbrace{\mathbb{E}_{p \in \Gamma_{uv}}\big[f_w(p)\big]}^{=f_w(u,v)\ (\mathsf{ab})}\Big]}_{=f_w(u)\ (\mathsf{bp})}\Big] \ , \tag{12}$$

where u is sampled uniformly at random from V, v is sampled uniformly at random from $V \setminus \{u\}$, and p is sampled uniformly at random from Γ_{uv}, or is $p = \emptyset$ if u and v are not connected. Thus, each estimator computes a conditional expectation as a proxy for the (unconditional) expectation of the betweenness centrality; bp conditions only on the first vertex u, ab conditions additionally on the second vertex v, and rk further conditions on a randomly sampled shortest path $p \in \Gamma_{uv}$.

We defer the proofs of the results in this section to App. A.1.

Theorem 4.1. *For the raw variances of the three estimators it holds*

$$\forall w \in V : \ \big(\mathsf{b}(w)\big)^2 \leq \mathbb{E}_{\pi_{\mathsf{bp}}}\left[(f_w(u))^2\right] \leq \mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(u,v))^2\right] \leq \mathbb{E}_{\pi_{\mathsf{rk}}}\left[(f_w(p))^2\right] = \mathsf{b}(w) \ .$$

The same relationship extends to the wimpy variances of the families of each estimator.

The raw variance of the rk family is completely specified as in the above theorem (rightmost equality), because the functions in this family are binary. For the other families, we show more informative bounds in Sect. 4.1.

Theorem 4.2. *For the Rademacher averages of the three estimators it holds*

$$\mathsf{R}_m(\mathcal{F}_{\mathsf{bp}}, \pi_{\mathsf{bp}}) \leq \mathsf{R}_m(\mathcal{F}_{\mathsf{ab}}, \pi_{\mathsf{ab}}) \leq \mathsf{R}_m(\mathcal{F}_{\mathsf{rk}}, \pi_{\mathsf{rk}}) \ .$$

We use Thm. 4.2 in Sect. 5.1, to show how to adapt upper bounds to the sample complexity of rk to that of ab and bp, improving existing bounds using, respectively, pseudodimension [61, Sect. 4.2] and the union bound [20].

Theorems 4.1 and 4.2 may seem to suggest that bp, having the lowest wimpy variance and Rademacher average, should be preferred over the other two estimators, but our experimental results will show a slightly different picture.

4.1 Refined Bounds on Wimpy Variances

We now show refined bounds for the wimpy variances of the families of functions corresponding to the ab and bp estimator (see Thm. 4.1 for a complete characterization of the wimpy variance of the rk family).

Lemma 4.3. *For any $w \in V$, it holds*

$$\mathsf{b}(w)\mathbb{E}_{\pi_{\mathsf{ab}}}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\middle|\sigma_{vu}(w) > 0\right] \leq \mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(u,v))^2\right] \leq \mathsf{b}(w) \max_{v,u \neq v} \frac{\sigma_{vu}(w)}{\sigma_{vu}} \ .$$

The lower bound improves over the naïve bound $(\mathsf{b}(w))^2$ showed in Thm. 4.1. The expectation in it is an interesting and "natural" statistic: it essentially is the centrality of w when considering

only the pairs of vertices v and u that have at least one SP between them that goes through w. The upper bound improves over the naïve $\mathsf{b}(w)$ upper bound, and again depends on properties of the graph, although the multiplier is likely often going to be 1.

We now show a similar result for the functions in the bp family.

Lemma 4.4. *For any $w \in V$, let v be drawn uniformly at random from V, and u be drawn uniformly at random from $V \setminus \{v\}$, it holds*

$$\mathsf{b}(w)\mathbb{E}_{\pi_{\mathsf{bp}}}\left[\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right]\middle|\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] > 0\right] \leq \mathbb{E}_{\pi_{\mathsf{bp}}}\left[(f_w(v))^2\right] \leq \mathsf{b}(w)\max_v \mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] .$$

The bounds again improve over the corresponding naïve bounds. The multiplier of $\mathsf{b}(w)$ on the l.h.s. can be thought as the betweenness of w when considering only the SPs from source vertices v that have at least one SP (to any destination) that goes through w. The multiplier to $\mathsf{b}(w)$ on the r.h.s. is strictly less than 1 as long as there is not a vertex v s.t. *all* SPs from v to any other vertex go through w, in which case v would be the maximizer. In unweighted graphs, such a vertex v can only be a vertex of (out-) degree 1 in the neighborhood of w.

4.2 Variance of the Estimators

An established way to compare unbiased estimators for the same quantity is to compare their variances, which also gives information about the (expected) Mean Squared Error of the estimators. Riondato and Kornaropoulos [60, Lemma 9] show that the variance of the bp estimator is not greater than the variance of the rk estimator, i.e., $\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{bp}}(w)] \geq \mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{rk}}(w)]$. In this section, we use Lemmas 4.3 and 4.4 to improve their analysis to show lower bounds to the *ratio* of the variances of any two of the three estimators for BC.

Lemma 4.5. *For any $w \in V$, it holds*

$$\frac{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{ab}}(w)]}{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{rk}}(w)]} \leq \max_{v,u\neq v}\frac{\sigma_{vu}(w)}{\sigma_{vu}} \qquad (\leq 1) .$$

The proof can be found in App. A.1. The following two results can be proved essentially in the same way.

Lemma 4.6. *For any $w \in V$, it holds*

$$\frac{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{bp}}(w)]}{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{rk}}(w)]} \leq \max_{v\in V}\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] \qquad (\leq 1) .$$

Lemma 4.7. *For any $w \in V$, it holds*

$$\frac{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{bp}}(w)]}{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{ab}}(w)]} \leq 1 .$$

Thus, bp has lower variance than the other two, and ab has lower variance than rk. These facts are not entirely surprising, because, informally, bp "collects more information" *per sample* than the others, thus its estimations, for the same sample size, are more accurate. Nevertheless, it is important to remark that each estimator uses samples from a different population (see Sect. 3.2), and drawing the samples from the different population requires a different amount of work. The comparison, while meaningful, hides these aspects. In our experiments, we evaluate how these and other properties show up in practice.

5 THE BAVARIAN FRAMEWORK

We now present BAVARIAN, our unifying algorithmic framework for BC estimation. BAVARIAN uses the k-MCERA and the variance-aware tail bound from Thm. 3.1 to compute the sample-dependent approximation quality, and can be instantiated with any of the estimators discussed in Sect. 3.2. The pseudocode is shown in Alg. 1.

Algorithm 1: BAVARIAN

Input: method A, graph G, sample size m, failure probability δ, Monte-Carlo trial count k
Output: $(\tilde{B}, \varepsilon)$ with the properties presented in Thm. 5.1

1 $\mathsf{sums} \leftarrow$ map from V to vectors of size $k+2$
2 **foreach** $v \in V$ **do** $\mathsf{sums}[v] \leftarrow (\underbrace{-\infty, \ldots, -\infty}_{k}, 0, 0)$
3 **for** i **from** 1 **to** m **do**
4 $\quad s_i \leftarrow \mathtt{drawSample}(\mathsf{A}, G)$
5 $\quad Z \leftarrow \mathtt{getFunctionValues}(\mathsf{A}, s_i)$
6 $\quad \boldsymbol{\lambda}_i \leftarrow \mathtt{drawRademacher}(k)$
7 $\quad$ **foreach** $(v, f_v(s_i)) \in Z$ **do**
8 $\quad\quad \mathsf{sums}[v] \leftarrow \mathsf{sums}[v] + f_v(s_i) \cdot (\boldsymbol{\lambda}_{i,1}, \ldots, \boldsymbol{\lambda}_{i,k}, f_v(s_i), 1)$
9 $\tilde{B} \leftarrow \{(v, \mathsf{sums}[v][k+2]/m) : v \in V\}$
10 $\varepsilon \leftarrow \mathtt{getEpsilon}(\mathsf{sums}, m, \delta)$
11 **return** $(\tilde{B}, \varepsilon)$

The input parameters to BAVARIAN are: a BC estimation method $\mathsf{A} \in \{\mathsf{rk}, \mathsf{ab}, \mathsf{bp}\}$, a graph G, a sample size $m > 0$, a failure probability $\delta \in (0, 1)$, and the number k of Monte-Carlo trials for the k-MCERA (see Sect. 5.1 for a discussion of how to choose m and k). The output is a pair $(\tilde{B}, \varepsilon)$, where $\tilde{B}$ is a set of pairs $(v, \tilde{\mathsf{b}}_{\mathsf{A}}(v))$ for each $v \in V$, with $\tilde{\mathsf{b}}_{\mathsf{A}}(v)$ being the estimate of $\mathsf{b}(v)$ using A, and $\varepsilon \in (0, 1)$ is the probabilistically-guaranteed accuracy as specified in the following theorem (proof deferred to after the description of the algorithm).

THEOREM 5.1. *With probability at least $1 - \delta$ (over the runs of the algorithm), the pair $(\tilde{B}, \varepsilon)$ is such that*

$$\max_{v \in V} \left| \tilde{\mathsf{b}}_{\mathsf{A}}(v) - \mathsf{b}(v) \right| < \varepsilon \ .$$

BAVARIAN first creates a map sums from V to vectors[3] of size $k+2$ (line 1 of Alg. 1), initialized to contain one key v for each $v \in V$. The vector $\mathsf{sums}[v]$ has $k+2$ elements, with the first k elements initialized to $-\infty$ and the last two elements initialized to zero (line 2). We explain the role of this map below. The algorithm then enters a loop which is repeated for m iterations (lines 3–8). At the beginning of iteration i (line 4), BAVARIAN draws one element s_i from $\mathcal{D}_{\mathsf{A}}$ according to the distribution π_{A}. For example, if A is ab, the algorithm would draw a pair (u, v) of distinct vertices uniformly among all such pairs. It then uses the function `getFunctionValues` to compute, using the procedure specified by A, the set

$$Z = \{(w, f_w(s_i)) \text{ for each } w \in V \text{ s.t. } f_w(s_i) \neq 0\} \ .$$

For example, if A is ab, Z would contain pairs $(w, \sigma_{uv}(w)/\sigma_{uv})$ for all and only the vertices w internal to a SP from u to v, where $s_i = (u, v)$ is the population element drawn at this iteration. Together with a vector $\boldsymbol{\lambda}_i$ of k independent Rademacher variables (line 6), the set Z is used to

[3]We use 1-based indexing for vectors, i.e., the first element has index 1.

update the map sums to maintain the following invariant: at the end of every iteration i of the loop, it must hold for each $v \in V$ that

$$\mathsf{sums}[v][z] = \begin{cases} \sum_{j=1}^{i} \boldsymbol{\lambda}_{j,z} f_v(s_j) & \text{for } z = 1, \ldots, k \\ \sum_{j=1}^{i} (f_v(s_j))^2 & \text{for } z = k+1 \\ \sum_{j=1}^{i} f_v(s_j) & \text{for } z = k+2; \end{cases} \tag{13}$$

After m iterations of the for loop (lines 3–8), the algorithm obtains the BC estimate $\tilde{\mathsf{b}}_{\mathsf{A}}(v)$ for each vertex v as $\tilde{\mathsf{b}}_{\mathsf{A}}(v) = \mathsf{sums}[v][k+2]/m$, and stores them in the set $\tilde{B}$ (line 9). The probabilistic accuracy guarantee ε is computed by the function getEpsilon (line 10). This function receives as input all the necessary parameters to compute the value on the r.h.s. of (11): for each $v \in V$, it holds $\mathsf{sums}[v][k+1] = \sum_{i=1}^{k} (f_v(s_i))^2 = m\beta$, and $\mathsf{sums}[v][z] = \sum_{i=1}^{k} \lambda_{iz} f_v(s_i)$ for each $z = 1, \ldots, k$. The output of the algorithm is then $(\tilde{B}, \varepsilon)$. We can now prove Thm. 5.1.

Proof of Thm. 5.1. Thanks to the maintained invariant (13), each estimate $\tilde{\mathsf{b}}_{\mathsf{A}}(v)$, for some $v \in V$, is a sample mean (over the sample $\mathcal{S} = \{s_1, \ldots, s_m\}$) of a specific function f_v, and an unbiased estimate of the exact BC $\mathsf{b}(w) = \mathbb{E}[f_v]$. Additionally, the invariant (13) ensures that all the quantities needed to compute ε as in (11) are available, and thus the thesis follows from Thm. 3.1. □

Computing the k-MCERA with matrix multiplication. As described, the algorithm computes the k-MCERA by maintaining the vectors sums (see (13)) throughout its execution, and at the end, it finds the maximum, over the vectors, of each of the first k entries, sums these maxima, and divides the obtained sum by mk. This approach is the easiest to describe and to implement, but we now give an alternative way of computing the k-MCERA based on *matrix multiplication*, which shines additional light on the properties of the different estimators.

Let L be the $m \times k$ matrix whose i-th row is the Rademacher vector $\boldsymbol{\lambda}_i$, i.e., $L_{i,j} = \boldsymbol{\lambda}_{i,j}$, and let W be the $n \times m$ matrix such that $W_{i,j} \doteq f_{v_i}(s_j)$, where $v_1, \ldots, v_n$ is an arbitrary labeling of the vertices. The matrix product $S \doteq W \times L$ is a $n \times k$ matrix whose i-th row corresponds to the first k entries of the vector $\mathsf{sums}[v_i]$. Thus, to obtain the k-MCERA, it is sufficient to sum the columnwise maxima of S, and divide the result by mk. The approach we described when presenting the algorithm essentially performs naïve matrix multiplication, but by instead computing W during the loop, and sampling L just before computing ε, it is possible to use efficient matrix multiplication algorithms (e.g., Strassen's [66], without going into theoretically-superior but impractical "galactic" algorithms [2]) to obtain the entries of the sums vectors needed to compute the k-MCERA.

A natural question to ask given this matrix multiplication interpretation is, "How does the computational complexity behave as a function of k?" In particular, our algorithm with naïve matrix multiplication exhibits $\mathrm{O}(nmk)$ behavior, which suggests that there is a steep cost to increasing k. However, by splitting the computation of $S = W \times L$ into $\mathrm{O}(nm/k^2)$ multiplications of $k \times k$ matrices, computing each with Strassen's algorithm, and aggregating the results, we obtain a time complexity of $\mathrm{O}(nmk^{0.8074})$. This implies a much smaller cost to increasing k, suggesting that the intrinsic complexity of the k-MCERA computation is not heavily dependent on k, which is of interest beyond the betweeness centrality problem we study in this work.

It is also interesting to reflect on how to represent the matrix S for different estimators. Consider for example the bp estimator, and let G be an undirected connected graph and v_i be a node with degree greater than one. The value $f_{v_i}(s_j)$, where s_j is a sampled vertex in bp, always obeys $f_{v_i}(s_j) > 0$, except when $v_i = s_j$, thus S is usually dense for bp. In contrast, for the other estimators, $f_{v_i}(s_j)$ is often zero. For example, in the case of rk, $f_{v_i}(s_j) = 0$ for a sampled SP s_j whenever $v_j \notin \mathsf{int}(p_j)$. Thus, for the rk estimator, and to a lesser extent the ab estimator, a *sparse representation*

for S is more efficient. While matrix multiplication is typically studied in the dense setting, simple recursive methods like Strassen's algorithm apply equally to sparse matrix multiplication.

In conclusion, an efficient implementation of BAVARIAN may consider the use of matrix multiplication for computing the k-MCERA, choosing the appropriate representation for the matrices. As always in practice, only appropriate measurements on actual machines and representative inputs can guide this kind of optimization.

5.1 Choosing the parameters m and k

The sample size m and the number k of Monte-Carlo trials are important input parameters of BAVARIAN. We now discuss how to choose m as a function of a *user-specified* desired approximation quality $\bar{\varepsilon} \in (0, 1)$, and in Sect. 5.1.1 we discuss the choice of k. These results are not just of theoretical interest: we use them in a *progressive-sampling* variant of BAVARIAN in Sect. 5.2.

Minimum sample size. We start by deriving an upper bound to the minimum sample size m needed for the approximation quality guarantee ε returned by the algorithm to be *no-greater* than a user-desired upper bound $\bar{\varepsilon}$. From Thm. 3.1 it is evident that ε decreases as $\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\lambda})$ and β from (9) decrease, and as m and k increase. We here make the assumption that the zero-constant function belongs to $\mathcal{F}$, which is true whenever there is a vertex with BC zero, which is a quite common situation in practice, as it happens, among other cases, when the graph is undirected and there is a vertex of degree one, or when the graph is directed and there is a node with only outgoing edges or only incoming edges. Consider a procedure `zeroEpsilon(`m, k, η`)` that, given m, k, and η, computes ε as in Thm. 3.1 assuming that $\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\lambda}) = 0$ (which is the minimum possible because of the above assumption[4]) and $\beta = 0$, and let

$$\mathsf{m}_*(k, \eta, \bar{\varepsilon}) = \min\{m > 0 \ : \ \texttt{zeroEpsilon}(m, k, \eta) \leq \bar{\varepsilon}\} \ . \tag{14}$$

FACT 1. *BAVARIAN would never return a pair $(\tilde{B}, \varepsilon)$ with $\varepsilon \leq \bar{\varepsilon}$ if the passed sample size m is smaller than $\mathsf{m}_*(k, \eta, \bar{\varepsilon})$ (and the other input parameters are k and $\delta = \eta$).*

Computing $\mathsf{m}_*(k, \eta, \bar{\varepsilon})$ is easily done with an unbounded binary search on m using `zeroEpsilon`. In Fig. 2 we show a plot of $\mathsf{m}_*(k, 0.1, \bar{\varepsilon})$ for different values of k and $\bar{\varepsilon}$.

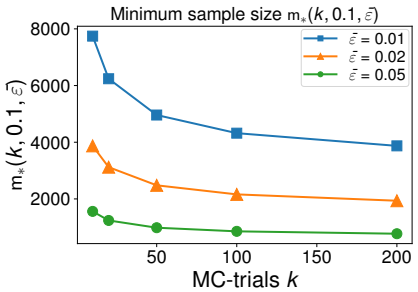


Fig. 2. Minimum sample size $\mathsf{m}_*(k, 0.1, \bar{\varepsilon})$.

For $m \geq \mathsf{m}_*(k, \eta, \bar{\varepsilon})$, the algorithm *may* return $\varepsilon < \bar{\varepsilon}$, but the assumptions $\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\lambda}) = 0$ and $\beta = 0$ require assumptions on the graph (mentioned above) and are so optimistic that it is almost impossible for the algorithm to return $\varepsilon < \bar{\varepsilon}$ for a sample size m *not much larger than* $\mathsf{m}_*(k, \eta, \bar{\varepsilon})$.

[4]Even without the assumption about the zero-constant function being in $\mathcal{F}$, it holds, for all graphs, that $\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\lambda}) \geq 0$ with probability at least 1/2, so 0 would be a lower bound to this quantity with the same probability.

Nevertheless, we leverage this property in Sect. 5.2, where we use (14) to derive a sample schedule for a progressive-sampling variant of Bavarian.

Sufficient sample size. We now present a result connecting structural properties of the graph G to a sample size *sufficient* for the algorithm to return an approximation quality bound ε not larger than the user-desired $\bar{\varepsilon}$. In the general case, the sample size we show depends on the *vertex-diameter* $\mathsf{vd}(G)$ of the graph G [60, Def. 1], i.e., the *maximum number of vertices on a SP in* G. On unweighted graphs, the vertex diameter equals the diameter plus one, as $\mathsf{vd}(G)$ counts *vertices*, while the diameter counts *edges*. On weighted graphs, the two quantities are not necessarily related. All that is really needed is an *upper bound* to the vertex-diameter, which is easy to compute in either case (see below). Riondato and Kornaropoulos [60, Coroll. 1] show that the *Vapnik-Chervonenkis (VC-) dimension* [68] (see [63, Ch. 6] for an introduction) of the task of approximating BC using the rk estimator is upper bounded by $\lfloor \log_2(\mathsf{vd}(G) - 2) \rfloor + 1$. They then use this result to derive a sufficient sample size for their BC approximation algorithm. They also show that, when G is undirected and there is no more than one SP between any pair of vertices in G, as is sometimes enforced on road networks, the VC-dimension collapses to 3, independently from the vertex-diameter of the graph [60, Lemma 2]. These results are limited to the rk estimator and do not make use of the Rademacher averages, which could lead to a much smaller sufficient sample size (see Thm. 5.4). We *extend and adapt* these results so that they can also be used to derive sufficient sample sizes for the bp and ab estimators and when using Rademacher averages, as shown in the following theorem.

Theorem 5.2. *Let $d \geq \lfloor \log_2(\mathsf{vd}(G) - 2) \rfloor + 1$ or $d = 3$ if G is undirected and there is at most one SP between any pair of vertices in G. Let*

$$\mathsf{m}^*(\bar{\varepsilon}, \eta) \doteq \frac{1}{\bar{\varepsilon}^2}\left(4Cd + 4\sqrt{\frac{Cd \ln \frac{2}{\eta}}{2}} + \frac{\ln \frac{2}{\eta}}{2}\right), \tag{15}$$

where C is a universal constant [36, 44, 64]. When run with inputs $m \geq \mathsf{m}^*(\bar{\varepsilon}, \eta)$, and $\delta = \eta$, Bavarian could return $(\tilde{B}, \bar{\varepsilon})$ rather than using the computed ε, and Thm. 5.1 would still hold.

Let us make some comments on Thm. 5.2 before delving in the proof. This result is *not only of theoretical interest*. Rather, it has a double *practical impact*: *(1)* a better characterization of the relationship between sample size and accuracy guarantee enables the user to better select what sample size to use; and *(2)* pushing down the sample size sufficient to obtain a desired approximation guarantee directly benefits progressive sampling algorithms (see Sect. 5.2), which can now deterministically terminate earlier.

It is evident from the statement of Thm. 5.2 that an *upper bound* to the vertex-diameter of G is sufficient to compute the sufficient sample size. In unweighted undirected graphs, it is possible to obtain a multiplicative 2-approximation of the diameter (thus of the vertex-diameter) with one SSSP computation from any source, by summing the lengths of the two longest SPs. For unweighted directed graphs, a direct and a reverse SSSP from the same source are needed. The resulting upper bound to the VC-dimension is an additive 1-approximation of the bound that would have been obtained if the exact vertex-diameter were used [60, Page 456].

Bergamini and Meyerhenke [8, Sect. 3] show how to get a multiplicative 2-approximation (assuming constant weights) to $\mathsf{vd}(G)$ on weighted graph, which leads to an additive 2-approximation of the upper bound to the VC-dimension.

The proof of Thm. 5.2 relies on a result (Thm. 5.4) to upper bound the Rademacher average $\mathsf{R}_m(\mathcal{F}, \pi)$ (see (8)) with the VC-dimension. Theorem 5.4 allows us to adapt the result on the VC-dimension of BC-estimation with the rk estimator [60, Coroll. 1, Lemma 2] to Rademacher averages.

The following result, a pinnacle of statistical learning theory, connects the Rademacher average to the supremum deviation.

Theorem 5.3 ([63, Thm. 26.5]). *With probability at least* $1-\eta$ *over the choice of* $\mathcal{S}$*, it holds*

$$\mathsf{SD}(\mathcal{F},\mathcal{S}) \le 2\mathsf{R}_m(\mathcal{F},\pi) + \sqrt{\frac{\ln\frac{2}{\eta}}{2m}} \ . \tag{16}$$

In all cases of interest for us, $\mathcal{F}$ and π are associated to a BC approximation method A (see Sect. 3.2), so we use the notation $\mathsf{R}_m(\mathsf{A})$ to refer to $\mathsf{R}_m(\mathcal{F}_\mathsf{A},\pi_\mathsf{A})$.

Theorem 5.4. *Let* $d \ge \lfloor \log_2(\mathsf{vd}(G)-2)\rfloor + 1$ *or* $d=3$ *if* G *is undirected and there is at most one SP between any pair of vertices in* G*.*

$$\mathsf{R}_m(\mathsf{rk}) \le \sqrt{\frac{288d\beta\ln\frac{e^3}{\sqrt{\beta}}}{m}} \tag{17}$$

Furthermore, for some universal constant $C < 262$*, it holds that*

$$\mathsf{R}_m(\mathsf{rk}) \le \sqrt{\frac{Cd}{m}} \ .$$

The proof of Thm. 5.2 for the rk estimator then follows from using the upper bound in Thm. 5.4 as an upper bound to the Rademacher average in Thm. 5.3, requiring the l.h.s. of (16) to be at most $\bar{\varepsilon}$, and solving for m to obtain the expression for $\mathsf{m}^*(\bar{\varepsilon},\eta)$. We can then use Thm. 4.2 to extend the result to the other two estimators ab and bp.

5.1.1 Choosing the number of Monte-Carlo trials. The beauty of the Monte-Carlo bound ρ in Thm. 3.1 is that the uncertainty error term added to $\hat{\mathsf{R}}^k_m(\mathcal{F},\mathcal{S},\mathbf{\Lambda})$ is *asymptotically negligible* for *any choice* of k (i.e., k is absent from the asymptotic form). A value $k \ge 8$ would match the $8r$ term inside the square root of ε, after which the Monte-Carlo estimation terms are negligible compared to other error terms. Constant-factor improvements are still possible for $k > 8$, though due to rapidly diminishing returns in k, additional computational resources are better spent on computing more shortest paths (i.e., taking more samples) than increasing k unboundedly.

From a computational perspective, k can be selected so the cost of computing SPs amortizes the cost of running k Monte-Carlo trials. Computing the k-MCERA[5] requires $\mathrm{O}(mkn)$ time for BC (often much less for the rk and ab estimators, see also Sect. 6). Assuming BFS with time complexity $\mathrm{O}(|E|)$, it makes sense to have $k \in \mathrm{O}(|E|/n)$ trials, and with Dijkstra's algorithm (time complexity $\mathrm{O}(|E| + n\log n)$), to have $k \in \mathrm{O}(|E|/n + \log n)$.

5.2 Progressive sampling algorithm

The algorithmic framework discussed so far uses *static* or *one-shot* sampling: it draws a *single* sample $\mathcal{S}$ of user-specified size m from the appropriate population, and uses information from the sample to compute the quality guarantee ε. In Sect. 5.1, we discussed how the user can make an informed decision on m based on their desired approximation quality $\bar{\varepsilon}$. The sample sizes m_* and m^* from (14) and (15) give a range $[m_*, m^*]$ such that Bavarian may return a value ε smaller than $\bar{\varepsilon}$ when the given sample size m is within this range. The likelihood that ε is smaller than $\bar{\varepsilon}$ clearly increases as m gets closer to m^*, and the algorithm is *guaranteed* to return $\varepsilon < \bar{\varepsilon}$ when $m \ge m^*$. We now introduce a more flexible variant of Bavarian based on *progressive sampling*: rather than

[5]We assume to not be using efficient matrix multiplication techniques, e.g., Strassen's algorithm (see the paragraph "Computing the k-MCERA with matrix multiplication" at the beginning of Sect. 5). Of course, using $\mathrm{o}(k)$ methods to compute the k-MCERA allows us to commensurately increase k while still amortizing this cost.

using a single fixed size sample, our algorithm BAVARIAN-P uses a *sample schedule* $\Delta = (m_i)_{i=0}^{T-1}$ to *iteratively grow* the sample $\mathcal{S}$ from an initial size m_0 to larger sizes $m_1, \ldots, m_{T-1}$ (where T is the maximum number of iterations, discussed below). The algorithm stops at the earliest iteration i such that the approximation quality ε obtained from the sample $\mathcal{S}$ at iteration i is no larger than the user-desired quality $\bar{\varepsilon}$. Thus, $\varepsilon \leq \bar{\varepsilon}$ is the *stopping condition* of our algorithm. The computation of ε must handle the fact that we are essentially analyzing multiple samples, as discussed in the proof of Thm. 5.5, which states the properties of BAVARIAN-P. We first present the algorithm (pseudocode in Alg. 2) with the sample schedule Δ and the desired quality $\bar{\varepsilon}$ passed as an input parameters, and then discuss appropriate choices for Δ. The only requirement on Δ is

$$m_{T-1} \geq \mathsf{m}^*\left(\bar{\varepsilon}, \frac{\delta}{T}\right) \tag{18}$$

for technical reasons explained in the proof of Thm. 5.5.

Algorithm 2: BAVARIAN-P

Input: method A, graph G, sample schedule $\Delta = (m_i)_{i=0}^{T-1}$, failure prob. δ, no. of MC-trials k, desired quality $\bar{\varepsilon}$
Output: $(\tilde{B}, \varepsilon)$ with the properties presented in Thm. 5.5

1 sums $\leftarrow$ map from V to vectors of size $k+2$
2 **foreach** $v \in V$ **do** $\mathsf{sums}[v] \leftarrow (\underbrace{-\infty, \ldots, -\infty}_{k}, 0, 0)$
3 $m_{-1} \leftarrow 0$, $i \leftarrow 0$
4 **do**
5 | `drawSamplesAndUpdateSums`(sums, $m_i - m_{i-1}$, A, G)
6 | **if** $i < T-1$ **then** $\varepsilon \leftarrow$ `getEpsilon`(sums, m_i, δ/T)
7 | **else** $\varepsilon \leftarrow \bar{\varepsilon}$
8 | $i \leftarrow i+1$
9 **while** $\varepsilon > \bar{\varepsilon}$
10 $\tilde{B} \leftarrow \{(v, \mathsf{sums}[v][k+2]/m_i) : v \in V\}$
11 **return** $(\tilde{B}, \varepsilon)$

BAVARIAN-P first initializes the data structure sums exactly as in the static-sampling case (see Alg. 1 and its description), and then, starting at $i = 0$, enters a loop (lines 4–9) that continues until the stopping condition is satisfied (more details in the following). At each iteration of the loop, the function `drawSamplesAndUpdateSums` is called (line 5). It performs the operations on lines 3 to 8 of Alg. 1: it draws $m_i - m_{i-1}$ samples from the population, computes the function values, draws $m_i - m_{i-1}$ k-dimensional Rademacher vectors, and appropriately updates sums while maintaining the invariant from (13). Then (line 6), the algorithm computes the approximation quality guarantee ε that can be obtained from the sample of size m_i seen so far, using δ/T as the failure probability (as discussed for Alg. 1).[6] BAVARIAN-P stops iterating when the computed ε is not larger than the desired $\bar{\varepsilon}$, which is guaranteed to be the case when i equals $T-1$, due to (18). The approximation $\tilde{B}$ is computed (line 10) and output together with ε.

[6]The reason for the division by T is that, as it will be clear from the proof of Thm. 5.5, a union bound is taken over all possible iterations. For ease of presentation, we assign to every iteration an equally failure probability $\delta_i \doteq \delta/T$, but more refined ways to "spread" δ over the iterations are possible, as long as the sum of the δ_i's equals δ.

Theorem 5.5. *With probability at least* $1-\delta$ *(over runs of* Bavarian*), the pair* $(\tilde{B}, \varepsilon)$ *returned by* Bavarian-P *is such that*

$$\max_{v \in V} \left| \tilde{\mathsf{b}}_{\mathsf{A}}(v) - \mathsf{b}(v) \right| < \varepsilon \leq \bar{\varepsilon} \ .$$

Proof. Consider a variant V of Bavarian-P that, rather than sampling progressively, receives as input, in addition to all the parameters listed in Alg. 2, an *ordered sequence*

$$\mathcal{Z} = \{(s_1, \boldsymbol{\lambda}_1), \ldots, (s_{m_{T-1}}, \boldsymbol{\lambda}_{m_{T-1}})\},$$

where $s_j \in \mathcal{D}_{\mathsf{A}}$ is sampled independently (from anything else) according to π_{A}, and $\boldsymbol{\lambda}_j$ is an independently-sampled k-dimensional vector of independent Rademacher variables. The variant V works essentially as Alg. 2 with the exception that, instead of sampling elements of π_{A} and sampling vectors of Rademacher variables, it just picks the "next" element from $\mathcal{Z}$ as needed.

The reason for introducing V is to make it evident that it is possible to reason about properties of $\mathcal{Z}$ *independently* from the workings of the algorithm, as $\mathcal{Z}$ is fixed before the algorithm even starts. Next, we show that V returns a pair $(\tilde{B}, \varepsilon)$ with the desired properties mentioned in the thesis, and then we argue how to extend these properties to Alg. 2.

Fix $i \in [0, T-1]$, and consider the value ε_i computed by V using the sequence of samples $\mathcal{Z}_i = \{(s_1, \boldsymbol{\lambda}_1), \ldots, (s_{m_i}, \boldsymbol{\lambda}_{m_i})\}$. An application of Thm. 3.1 allows us to say that, with probability at least $1 - \delta/T$ (over the choice of $\mathcal{Z}_i$), it holds $\max_{v \in V} \left| \tilde{\mathsf{b}}_{\mathsf{A}}(v) - \mathsf{b}(v) \right| < \varepsilon_i$.

We can then apply the union bound over the T iterations, and obtain that, with probability at least $1 - \delta$ over its runs (i.e., over the choice of the sequence $\mathcal{Z}$), the output $(\tilde{B}, \varepsilon)$ of V satisfies

$$\max_{v \in V} \left| \tilde{\mathsf{b}}_{\mathsf{A}}(v) - \mathsf{b}(v) \right| < \varepsilon \ . \tag{19}$$

The last sample size m_{T-1} must satisfy the requirement from (18) and when using this sample size, V outputs $\varepsilon = \bar{\varepsilon}$. Combining this fact with (19), we get that the output $(\tilde{B}, \varepsilon)$ of V satisfies

$$\max_{v \in V} \left| \tilde{\mathsf{b}}_{\mathsf{A}}(v) - \mathsf{b}(v) \right| < \varepsilon \leq \bar{\varepsilon} \ .$$

We now argue that the properties enjoyed by V can be extended to Bavarian-P. Indeed to each execution of V with a set of parameters, we can associate an execution of Bavarian-P with the same set of parameters (minus $\mathcal{Z}$) such that the sequence of random bits used to generate $\mathcal{Z}$ is given to Bavarian-P and used to draw samples from $\mathcal{D}_{\mathsf{A}}$ and the Rademacher vectors $\boldsymbol{\lambda}$ when needed. Clearly, the ordered sequence of samples and Rademacher vectors generated by Bavarian-P in this execution is a sub-sequence (potentially improper) of the sequence $\mathcal{Z}$ given in input to V, and specifically exactly the sub-sequence of all and only the pairs that the execution of V "used" (the length of this sub-sequence is obviously m_i for some $i \in \{0, \ldots, T-1\}$). When using the sample size m_{T-1}, V returns $\bar{\varepsilon}$. The stopping condition on line 9 of Alg. 2 is guaranteed to be satisfied at some iteration, i.e., Bavarian-P will terminate. The pair $(\tilde{B}, \varepsilon)$ returned by this execution of Bavarian-P is the same as the one returned by the associated execution of V, and this correspondence is true for all executions of the two algorithms. Hence, the same properties on the output of V also hold for Bavarian-P, and the proof is complete. □

Choosing the sample schedule. We now discuss reasonable choices for the sample schedule $\Delta = (m_i)_{i=0}^{T-1}$. Assume, for ease of presentation, that the number of iterations T is a fixed parameter chosen by the user. We later remove this assumption. The correctness of the algorithm requires that the last sample size m_{T-1} satisfies (18), and, given what we discussed in Sect. 5.1, it makes no sense to use a larger sample size than the quantity in the r.h.s. of (18). Thus, we assume that m_{T-1} equals this quantity. It follows from Fact 1 that it also does not make sense to have $m_0 \leq \mathsf{m}_*(k, \delta/T, \bar{\varepsilon})$,

where m_* is defined in (14). It is not necessarily appropriate to use *exactly* $\mathsf{m}_*(k, \delta/T, \bar{\varepsilon})$ for m_0, for the reasons argued in Sect. 5.1, but this quantity can be used as a *lower bound* to m_0. Once m_0 and m_{T-1} have been fixed, we need to choose $T-2$ intermediate sample sizes between these two extremes. A reasonable approach is to follow a *geometrically-increasing* sample schedule, which has been shown in practice and in theory to be close to optimal [24, 40]. Thus, we have

$$m_i = m_0 \left(\frac{m_{T-1}}{m_0}\right)^{\frac{i}{T-1}}, \text{ for } i = 1, \ldots, T-2 \ .$$

The scaling factor m_{T-1}/m_0 is a result of fixing the number of iterations in advance. It is often more convenient to let the user specify a desired scaling factor $\theta > 1$. An unbounded binary search then finds the minimum number of iterations T^* such that there are exactly T^*-2 sample sizes $m_i = m_0\theta^i$ for $i = 1, \ldots, T^* - 2$ between the smallest sample size (computed using (14) with $\eta = \delta/T^*$) and the largest m_{T^*-1} (from (18), potentially smaller than $m_0\theta^{T^*-1}$).

An adaptive sampling approach. It is tempting to assume that one could easily make the sample schedule $\Delta = (m_i)_{i=0}^{T-1}$ *dynamic*, i.e., not fixed *a priori* as an input parameter to the algorithm. For example, one may want to use information obtained at the current iteration i from $\mathcal{S}_i$ to determine the sample size m_{i+1} to use at iteration $i+1$ so that $\varepsilon_{i+1} \leq \bar{\varepsilon}$, thus obtaining an *adaptive* progressive sampling algorithm. In the rest of this section, we use BAVARIAN-A to refer to such an algorithm.

The situation is more complicated than it may seem at first: using any information from the sample $\mathcal{S}_i$ to determine how much to grow $\mathcal{S}_i$ to obtain $\mathcal{S}_{i+1} \supset \mathcal{S}_i$, introduces *dependencies* between the samples that invalidate the proof of correctness from Thm. 5.5 and do not seem easy to handle. A workaround is to use *independent samples*, i.e., having $\mathcal{S}_{i+1}$ created from scratch rather than by growing $\mathcal{S}_i$. The correctness is then preserved, even if the *sample size* m_{i+1} depends on $\mathcal{S}_i$.

A second issue is that, depending on how m_{i+1} is chosen, it may happen that the algorithm uses an m_{i+1} only slightly larger than m_i, leading to negligible improvement (i.e., decrease) in ε_{i+1} w.r.t. ε_i, at the expense of many additional samples being drawn, thus effectively "wasting" an iteration. In addition to the direct cost of sampling (due to needing "completely fresh" samples at every iteration), this waste is expensive because it consumes an iteration for negligible benefit. Wasting an iteration is highly detrimental, because the number of iterations T is fixed in BAVARIAN-A, and the sample size m_{T-1} used at the T-th iteration (i.e., the last) must be such that the algorithm will return $\varepsilon_{T-1} \leq \varepsilon$. In order to guarantee this property, the sample size m_{T-1} may need to be much larger than the previous sample sizes, thus BAVARIAN-A must do all it can to terminate at an earlier iteration. A good way to address both of these issues is to ensure that successive sample sizes are not too close to each other, for example by imposing that $m_{i+1} \geq m_i\theta$, for some $\theta > 1$.

One way to determine m_{i+1} is to consider the values of the k-MCERA and of the empirical wimpy variance β obtained at iteration i, and use them in (11) to compute an m for which the resulting ε is not larger than $\bar{\varepsilon}$, which can be done with unbounded binary search, and take the larger of this value and θm_i. This approach may be too conservative, as the k-MCERA is generally decreasing in the sample size, and guessing that it will decay at a $\Theta(1/\sqrt{m})$ rate (see Thm. 5.4) yields a more aggressive estimate, though it increases the likelihood of not terminating before the T-th iteration.

It should be evident that this (and likely any) approach to compute the next sample size is a *heuristic*, in the sense that it does not guarantee that $\varepsilon_{i+1} \leq \bar{\varepsilon}$, and more refined heuristics can be developed. It is also possible to take into account information obtained at *all* iterations, rather than just the most recent one, to compute m_{i+1}, for example through curve fitting. We do not discuss these approaches here, as we focus on the main ideas of the adaptive sampling approach.

If BAVARIAN-A does not stop before the T-th iteration, there are several ways to select the last sample size m_{T-1} so that the algorithm can return $\varepsilon_{T-1} \leq \bar{\varepsilon}$. Naïvely, one could use the same a

priori sufficient sample size m_{T-1} used by BAVARIAN-P (see (15)). This quantity is usually very large, and we can use the adaptive nature of BAVARIAN-A to at least get some theoretical improvement, as follows. The union bound over all T iterations (which comes from the proof of correctness for BAVARIAN-P), ensures that all upper bounds to, e.g., the wimpy variance obtained in the previous iterations of BAVARIAN-A remain valid, so we may use the best bound γ thus-far obtained on the wimpy variance in place of β in (17), to obtain an expression for an upper bound to the Rademacher average, and then use this expression in place of r in (11) (with $\eta = \delta/T$), and solve for m using unbounded binary search to obtain m_{T-1}. When using this sample size,[7] BAVARIAN-A will return $\varepsilon_{T-1} = \bar{\varepsilon}$, like BAVARIAN-P does at the last iteration.

It is difficult to precisely analyze the additional cost, in terms of the number of drawn samples, of BAVARIAN-A w.r.t. the "one-shot" static sampling algorithm BAVARIAN, when the adaptive algorithm terminates before the T-th iteration, but we can at least make some *informal* claims about this additional cost in a restricted case.

Fix $\xi \in (0, 1)$, and let $m_{\mathrm{S},\bar{\varepsilon}} \doteq m_{\mathrm{S},\bar{\varepsilon}}(\xi)$ be the minimum sample size such that BAVARIAN returns an error bound $\varepsilon \leq \bar{\varepsilon}$ in a fraction at least ξ of all possible samples of size $m_{\mathrm{S},\bar{\varepsilon}}(\xi)$. The quantity $m_{\mathrm{S},\bar{\varepsilon}}$ is unknown, and it does not seem to be possible to derive an expression for it, but that is not a problem for our *informal* reasoning. Assume $\delta \leq 1/e$, then the size $m_{\mathrm{A},\bar{\varepsilon}} = m_{\mathrm{A}\bar{\varepsilon}}(\xi)$ with the same property for BAVARIAN-A should be

$$m_{\mathrm{A},\bar{\varepsilon}} \approx m_{\mathrm{S},\bar{\varepsilon}} \frac{\ln T + \ln \frac{1}{\delta}}{\ln \frac{1}{\delta}} \leq m_{\mathrm{S},\bar{\varepsilon}} \ln(eT), \tag{20}$$

due to the union bound over the T iterations (see discussion for BAVARIAN-P).

Assume now that θ and m_0 are such that there exists a $0 \leq i < T - 1$ such that $m_{\mathrm{A},\bar{\varepsilon}} \leq m_0\theta^i$, and let j be the smallest such i. From the definition of $m_{\mathrm{A},\bar{\varepsilon}}$, with probability at least $1 - \xi$, BAVARIAN-A will stop at the earliest iteration k with a sample size $m_k \geq m_0\theta^j$. It must be $k \leq j$. Assume this event occurs, i.e., indeed BAVARIAN-A stops at the sample size m_k.

For every $0 \leq i \leq k$, it holds $m_i \leq m_k/\theta^{k-i}$. Then, the total number of samples drawn by BAVARIAN-A is

$$\sum_{i=0}^{k} \frac{m_k}{\theta^i} = m_k \frac{(1 - \frac{1}{\theta^{k+1}})\theta}{\theta - 1} \leq m_k \frac{\theta}{\theta - 1} \quad . \tag{21}$$

Now, in the worst case, it may happen that $m_{k-1} = m_{\mathrm{A},\bar{\varepsilon}} - 1$, but if the heuristic used by BAVARIAN-A works sufficiently well, then it would suggest a sample size for the next iteration that is less than $m_0\theta^k$, thus it will actually hold $m_k = m_0\theta^k$, i.e., $m_k \approx m_{\mathrm{A},\bar{\varepsilon}}\theta$. We can then conclude, using this fact and equations (21) and (20), that BAVARIAN-A draws, in total, a number of samples that is at most

$$m_{\mathrm{S},\bar{\varepsilon}} \frac{\theta^2}{\theta - 1} \ln(eT) \quad . \tag{22}$$

Thus, the cost of "complete" adaptivity, in the sense of completely freeing the user from having to choose any sample size, is not excessive: the multiplicative factor in (22) is optimized for $\theta = 2$, when it takes the value $4 \ln(eT)$.

5.3 Extensions

The unifying approach of BAVARIAN and BAVARIAN-P allows them to be adapted to work with many variants of BC and other centrality measures. To avoid overweighting our presentation, we

[7] This procedure can actually be used at any the end of any iteration i to compute another candidate value z for m_{i+1}: if $m_i\theta$ or the value computed using the information from the current sample are larger than this candidate value z, then z should be used for m_{i+1}, and the algorithm should terminate and return $\varepsilon_{i+1} = \bar{\varepsilon}$ at the end of the next iteration.

only remark that the extensions to other BC variants would follow the approach discussed in [60, Sect. 6], and de Lima et al. [27] show how to estimate the percolation centrality via sampling by adapting the work of Riondato and Kornaropoulos [60] and Riondato and Upfal [61]. One can similarly adapt Bavarian to estimate percolation centrality. When dealing with fully-dynamic graphs with arbitrary insertions and/or deletions of edges and/or vertices, Bavarian can be used as a drop-in replacement for ABRA, as discussed in [61, Sect. 5].

An interesting direction for future work is understanding the requirements for other centrality measures to be amenable to approximation using the k-MCERA and/or similar tools. The extension is less straightforward than it may seem at first, due to the need for efficient sampling schemes from the appropriate populations.

6 EXPERIMENTAL EVALUATION

We now present the results of our experimental evaluation of Bavarian and Bavarian-P and of the comparison between the different estimators presented in Sect. 3.2.

Goals. A first goal is to evaluate the behavior of the quality guarantee ε output by Bavarian as the input parameters m and k change, and to compare it with that of the state of the art for each estimator. This comparison allows us to assess the power of the k-MCERA and of the variance-aware tail bound on the SD (Thm. 3.1). We also evaluate the trade-off between ε and runtime for the different BC estimators, to understand which one is more "efficient" in terms of the accuracy guarantee improvement per unit of time. Bavarian *does not in any way change the empirical properties of the estimators* (e.g., the similarity between the ranking of the vertices by estimated BC and the ranking by exact BC), so we do not evaluate them. For in-depth evaluations of such very important aspects, which are not impacted by Bavarian, see the original works [20, 60, 61] and the experimental comparisons by Matta et al. [51] and AlGhamdi et al. [1].

Implementation, environment, and datasets. We implemented Bavarian and Bavarian-P in C++20 as an extension of NetworKit [65], and compared it with the implementation of Brandes's algorithm [18] included in the same suite. There are many opportunities for parallelization, some of which we implement, but when evaluating the algorithms we disable parallelism. All the code and datasets can be found at https://github.com/acdmammoths/Bavarian-code, including a `README.md` file with instructions, and a script to run all our experiments and to generate all the figures and tables. We run the experiments on a FreeBSD 11 Amazon AWS instance with 8 AMD EPYC CPUs and 32GB of RAM. We used different combinations of values for m, k, δ, and on different graphs (see below), doing five runs per combination. The difference across runs is minimal, which is not surprising as most of our results are not *absolutely* tight, while being *tighter* than previous works. We report the results for the median, the maximum and the minimum over the runs (represented as shaded areas around the curve for the median). We only show results for $\delta = 0.1$. This parameter has minimal impact on the performances of our algorithm, and results for other values of δ are qualitatively similar.

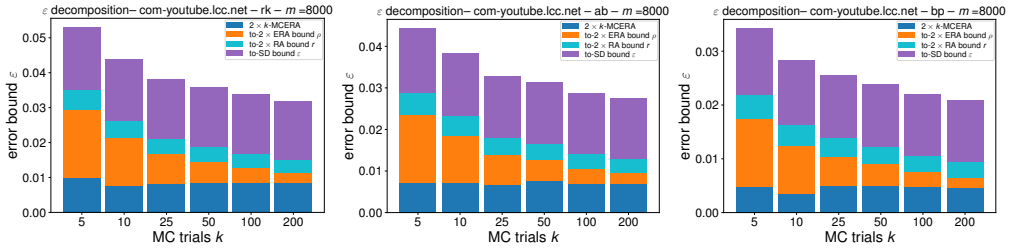
We used datasets from the SNAP repository [47]. Their salient characteristics are reported in Table 2.

6.1 Results

We start by remarking that in all the hundreds of runs we performed, the supremum deviation (i.e., the maximum error in our estimation) was always less than the error ε returned by Bavarian, and often significantly so. This fact is not surprising, as Thm. 3.1 is not necessarily tight. Hence, Bavarian is even more accurate than the theory guarantees, which leaves space for future improvements.

Table 2. Dataset characteristics

Graph	Vertices	Edges	Directed
ca-AstroPh	17,903	197,031	N
cit-HepPh	34,546	421,578	N
email-Enron	36,682	183,831	N
p2p-Gnutella31	62,586	147,892	Y
soc-Epinions1	75,879	508,837	Y
loc-gowalla	196,591	950,327	N
com-dblp	317,080	1,049,866	N
com-youtube	1,134,890	2,987,624	N
as-skitter	1,694,616	11,094,209	N



(a) rk (b) ab (c) bp

Fig. 3. Error bound ε decomposition.

Impact of k. We evaluate the impact of k on ε, as the impact of k on the runtime is negligible, because BAVARIAN's runtime is heavily dominated by the SSSP computations. We run BAVARIAN for different values of k and split ε in its components from Thm. 3.1. In Fig. 3, we show the decomposition, and observe the behavior as function of k, on the `com-youtube` graph, with $m = 8000$ (results for other values of the parameters and other datasets are qualitatively similar, and are shown in Fig. 15 in App. A.2). The x-axis of the figure has essentially a logarithmic scale. We can see that ε rapidly decreases as a function of k, but the returns are diminishing. The decrease of ε is readily explained by looking at its components: the dark blue (bottom) part of the bars is *twice* the k-MCERA (i.e., $2\hat{\mathrm{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda})$), the orange ($2^{\text{nd}}$ from bottom) portion is the difference between 2ρ and $2\hat{\mathrm{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda})$, the cyan ($3^{\text{rd}}$ from bottom) is the difference between $2r$ and 2ρ, and the purple (top portion) is the difference between ε and $2r$. The only part that changes significantly with k is the orange one, which corresponds to the probabilistic tail bound from the k-MCERA to the ERA, its expectation w.r.t. $\mathbf{\Lambda}$. As the k-MCERA is tightly concentrated around the ERA, it is not surprising that with even a relatively small number of MC trials, this tail bound becomes negligible w.r.t. the other terms.

Impact of m. Figure 4 show the impact of the sample size m on the error bound ε and on the runtime, respectively, for $k = 100$. As expected, the error bound decreases approximately as $1/sqrtm$ and the runtime increases linearly with m. The use of the k-MCERA and of Thm. 3.1 allows BAVARIAN to obtain much smaller (2–4x) error bounds than previous works for all estimators, and much closer to the empirical maximum error measured on the sample. Such an improvement is significant: the goal in developing sampling-based approximation algorithms, is to obtain error

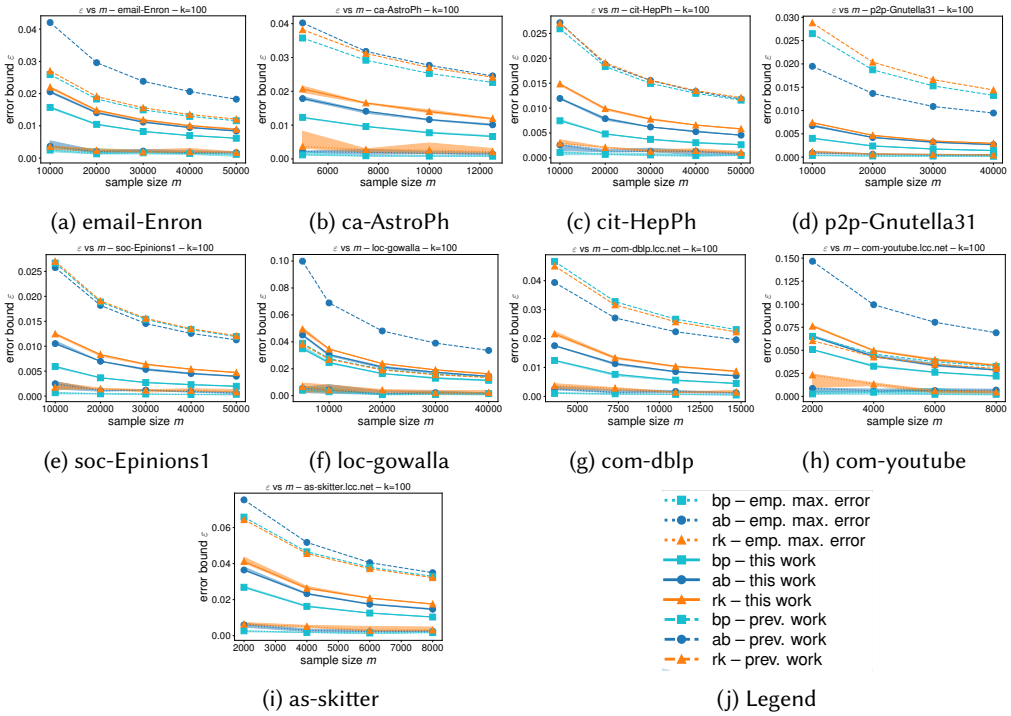


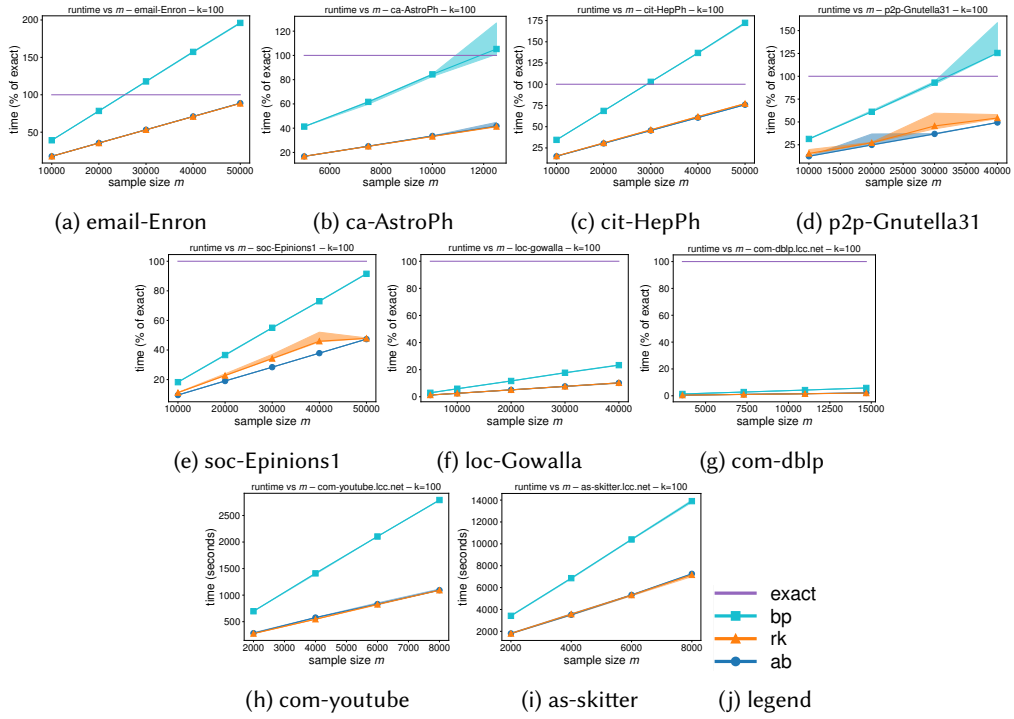
Fig. 4. Error bound ε vs. sample size m.

bounds (i.e., quality guarantees) that are as close as possible to the empirical maximum error, in order to be informative, as the empirical maximum error is not available in practice (if one had the exact results available, there would be no need to use the approximation algorithm). In this sense, the improvement in terms of the distance from the empirical maximum error can be considered much more important than the relative or absolute improvement over the previously available error bounds. The error bounds can still be tightened, but the leap forward towards the empirical maximum error is significant.

A 2–4x decrease in the error bound also directly translates in a 4–16x reduction in the sample size needed to obtain that upper bound (the error bound ε decreases as $O(1/\sqrt{m})$, where m is the sample size), thus in a similar reduction in the running time: such a reduction can mean the difference between being able to use the approximation algorithm (because it would take less time than running the exact algorithm), and not being able to use it.

From just Fig. 4, one may be tempted to say that the bp estimator should be preferred, because it results in a smaller ε. This conclusion would not be appropriate: as discussed in Sect. 3.2, the estimators sample from different populations, and do different work per sample, so their performance at equal sample sizes cannot really be compared.

Units of guarantee per unit of time. Figure 5 clearly shows that BAVARIAN with bp takes much longer at each sample size than with other estimators (which take approximately the same time

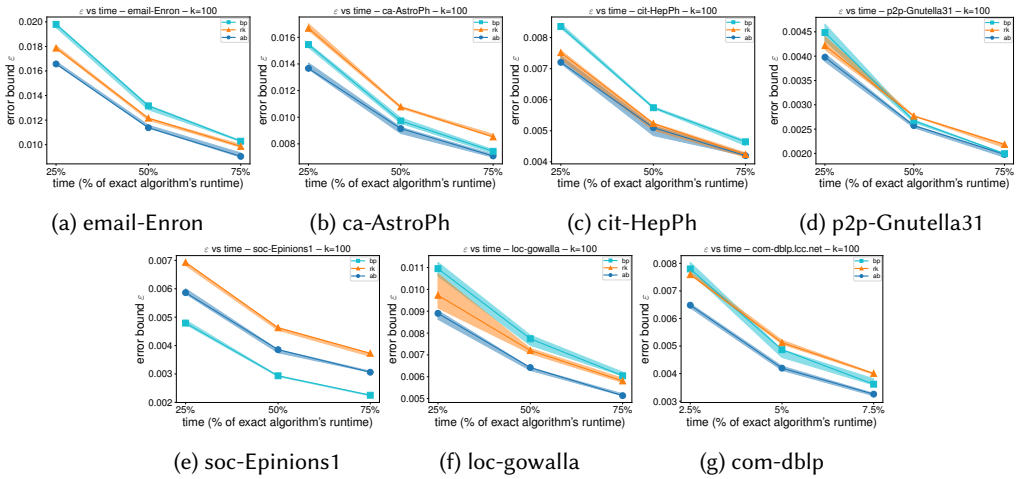


(a) email-Enron (b) ca-AstroPh (c) cit-HepPh (d) p2p-Gnutella31

(e) soc-Epinions1 (f) loc-Gowalla (g) com-dblp

(h) com-youtube (i) as-skitter (j) legend

Fig. 5. Runtime vs. sample size m.



(a) email-Enron (b) ca-AstroPh (c) cit-HepPh (d) p2p-Gnutella31

(e) soc-Epinions1 (f) loc-gowalla (g) com-dblp

Fig. 6. Error bound ε vs. time

because they do similar work per sample), and becomes slower than the exact algorithm earlier.[8]

[8]The com-youtube and as-skitter graphs are too large to run the exact algorithm on them, so Figs. 5h and 5i do not show a horizontal curve for the exact algorithm runtime, and we report the running times in seconds, rather than as percentages of the running time of the exact algorithm.

Thus, bp is more "efficient", in terms of "unit of guarantee" per sample unit, than the other estimators, but it is *less efficient per unit of time*. We therefore evaluate the error bound ε returned by BAVARIAN with each estimator when run for a fixed amount of time. Specifically, we use, as these timeouts, three percentages of the runtime of the exact algorithm on the graphs (25%, 50%, and 75% for each graph, except com-dblp, for which we used 2.5%, 5%, and 7.5% to keep the running time reasonable. We did not use com-youtube and as-skitter for this experiment because the exact algorithm would take an unreasonable amount of time on our hardware). In Fig. 6, we show the results.[9]

It is clear that rk should never be preferred, as it is always dominated by at least one of the other two estimators. The comparison between ab and bp is a bit more complicated: while in most graphs ab is more efficient, in soc-Epinions1 (Fig. 6e) bp gives a much better error bound. We analyzed the results in this dataset more in depth and found that the k-MCERA for bp approximately half that for ab. This aspect drives down the error bound for bp w.r.t. the one for ab, and is in part explained by the fact that the empirical wimpy variance β_{bp} for bp was close to 1/5 of that for ab, since the RA depends on the wimpy variance (see Thm. 5.4). We believe that this large difference may in part be caused by the fact that soc-Epinions1 is a directed graph, but we do not see the same phenomenon on other directed graphs (ca-AstroPh and p2p-Gnutella31). A theoretical study of the cases when bp may be more efficient than ab would be an interesting direction for future work, but seems extremely challenging and may lead to conclusions that are not actionable: the user may need to select an estimator before knowing anything about the graph. The results of this experiment make us conclude, that in most cases, ab should likely be preferred, as it gives the best "bang for the buck."

Evaluation of BAVARIAN-P. In our runs of BAVARIAN-P, we fixed $\delta = 0.1$, $k = 100$, and used a desired quality guarantee $\bar{\varepsilon} \in \{0.01, 0.015, 0.02\}$, and run the algorithm with a geometric schedule $m_i = \theta^i m_0$ for a scaling factor $\theta \in \{1.25, 1.5, 2\}$, and $m_0 = m_*(k, \delta, \bar{\varepsilon})$ (see (14)). The property of the estimations are the same as those of BAVARIAN, so we focus here on discussing the salient properties that are characteristic of a *progressive* sampling algorithm. The results are reported in Table 3 for email-Enron and Table 4 for com-dblp, while results for other datasets, which are qualitatively similar, are in App. A.2. In the tables we show the value ε actually returned by the algorithm, the ratio $\varepsilon/\bar{\varepsilon}$ (which must not be greater than 1), the ε that would have been returned by a *static* algorithm using the bounds presented in previous work (more details below), the *final* sample size, the iteration at which the algorithm returned $\varepsilon \leq \bar{\varepsilon}$, and the maximum number of iterations that the algorithm may have taken. The first observation we can make is that the algorithm stops in all cases relatively early, i.e., at an iteration much earlier than the last. This fact confirms the looseness of the terminal sample size $m^*(\varepsilon, \delta)$ (see (18)), computed using an upper bound to the VC-dimension, which is itself used in a worst-case upper bound to the n-MCERA. Developing better bounds is, as always, an interesting direction for future work, but not an easy one: Riondato and Kornaropoulos [60, Sect. 4.2] show that the upper bound to the VC-dimension is actually tight for some class of graphs. A potentially more promising direction could be trying to incorporate in the computation of $m^*(\varepsilon, \delta)$ an upper bound to the wimpy variance that depends on easy-to-compute characteristic

[9]We do not report the error bounds for the original "non-BAVARIAN" algorithms run for a percentage of the exact time, because to do so would require a reimplementation to support stopping after a specified amount of time and returning their error bounds. These modifications are not straightforward to make in the original code. Additionally, the original algorithms have other steps in addition to sampling that make implementing this feature a bit tricky: for example, the original rk needs to compute an approximation of the vertex-diameter to compute its error bound, and the original ab needs to perform a convex minimization. The code cannot know in advance how long these operations are going to take, which would impact the reported results. In BAVARIAN, we were able to eliminate such additional operations, so essentially all its running time is spent on sampling, as the computation of the error bound is straightforward and is done *while* sampling.

Table 3. Progressive sampling results — email-Enron

method	$\bar{\varepsilon}$	θ	ε	$\varepsilon/\bar{\varepsilon}$	prev. work ε	m	iter	max. iters
ab	0.010	1.25	0.009067	0.906651	0.019064	47047	12	34
		1.50	0.008498	0.849769	0.018373	50576	7	19
		2.00	0.009289	0.928888	0.019756	43264	4	11
	0.015	1.25	0.014332	0.955460	0.029723	19832	10	32
		1.50	0.013222	0.881440	0.027718	22418	6	18
		2.00	0.011484	0.765627	0.024365	28928	4	11
	0.020	1.25	0.019429	0.971460	0.039128	11867	9	31
		1.50	0.015737	0.786850	0.032777	16677	6	17
		2.00	0.013373	0.668675	0.028584	21376	4	10
bp	0.010	1.25	0.009537	0.953671	0.016691	24087	9	34
		1.50	0.009428	0.942846	0.017278	22478	5	19
		2.00	0.009853	0.985291	0.017613	21632	3	11
	0.015	1.25	0.013797	0.919800	0.022994	12692	8	32
		1.50	0.012333	0.822173	0.021190	14945	5	18
		2.00	0.012712	0.847440	0.021539	14464	3	11
	0.020	1.25	0.018339	0.916935	0.029726	7594	7	31
		1.50	0.018869	0.943460	0.030089	7412	4	17
		2.00	0.014877	0.743865	0.025057	10688	3	10
rk	0.010	1.25	0.009289	0.928880	0.012459	47047	12	34
		1.50	0.008800	0.880028	0.012016	50576	7	19
		2.00	0.009475	0.947457	0.012992	43264	4	11
	0.015	1.25	0.013385	0.892367	0.017163	24790	11	32
		1.50	0.013919	0.927900	0.018048	22418	6	18
		2.00	0.012122	0.808140	0.015888	28928	4	11
	0.020	1.25	0.017970	0.898505	0.022187	14834	10	31
		1.50	0.016520	0.825980	0.020926	16677	6	17
		2.00	0.014479	0.723970	0.018483	21376	4	10

quantities of the graph. In practice though, the fact that $m^*(\varepsilon, \delta)$ appear to be a bit loose has no impact, as the algorithm is still able to stop early.

A second observation is that ε is, from time to time, much smaller than $\bar{\varepsilon}$ (i.e., the ratio $\varepsilon/\bar{\varepsilon}$ is much smaller than 1). These results can be explained by the fact that the sample schedule is *fixed a priori*, and is guided by $\bar{\varepsilon}$, δ, and θ (see Sect. 5.2), thus the first iteration for which $\varepsilon \leq \bar{\varepsilon}$ may have a sample size sufficiently large that ε is much smaller than the user-desired error bound: a smaller sample size would have been sufficient to get $\varepsilon \leq \bar{\varepsilon}$, but the inflexibility of the sample schedule does not allow us to select the next sample size "on the fly". The choice of the parameter θ is important in this sense: a lower multiplier θ implies that the sample sizes of different iterations are closer to each other, thus the algorithm is *unlikely* to take many more samples than needed to get $\varepsilon < \bar{\varepsilon}$, and therefore will obtain a ε close to $\bar{\varepsilon}$. Nevertheless, it is difficult to *optimally* choose θ *a priori*: there are cases where, for lower θ, the algorithm terminates at a larger sample size than at higher values for this parameter (e.g., on email-Enron, using the bp method, with $\bar{\varepsilon} = 0.01$, or on the same graph with rk and $\bar{\varepsilon} = 0.015$.) This effect is again due to the fact that the sample schedule is fixed *a priori* and depends on θ, so the algorithm may not terminate at an iteration, but would have terminated

Table 4. Progressive sampling results — com-dblp

method	$\bar{\varepsilon}$	θ	ε	$\varepsilon/\bar{\varepsilon}$	prev. work ε	m	iter	max. iters
ab	0.010	1.25	0.009282	0.928247	0.024600	9865	5	34
		1.50	0.009456	0.945582	0.025405	9990	3	19
		2.00	0.008760	0.876025	0.023547	10816	2	11
	0.015	1.25	0.014432	0.962147	0.034764	5197	4	32
		1.50	0.012297	0.819820	0.030401	6642	3	18
		2.00	0.011612	0.774127	0.029827	7232	2	11
	0.020	1.25	0.017727	0.886330	0.040692	3888	4	31
		1.50	0.019693	0.984660	0.045224	3294	2	17
		2.00	0.014423	0.721135	0.035420	5344	2	10
bp	0.010	1.25	0.009652	0.965203	0.039380	5050	2	34
		1.50	0.007750	0.774992	0.034291	6660	2	19
		2.00	0.009362	0.936239	0.038054	5408	1	11
	0.015	1.25	0.013081	0.872033	0.048531	3325	2	32
		1.50	0.014771	0.984707	0.051506	2952	1	18
		2.00	0.012980	0.865360	0.046538	3616	1	11
	0.020	1.25	0.016806	0.840300	0.056104	2488	2	31
		1.50	0.018187	0.909340	0.059717	2196	1	17
		2.00	0.015939	0.796975	0.054138	2672	1	10
rk	0.010	1.25	0.009549	0.954909	0.024334	12332	6	34
		1.50	0.008724	0.872364	0.022076	14985	4	19
		2.00	0.006892	0.689154	0.018373	21632	3	11
	0.015	1.25	0.012863	0.857567	0.029985	8122	6	32
		1.50	0.014213	0.947560	0.033158	6642	3	18
		2.00	0.013469	0.897953	0.031777	7232	2	11
	0.020	1.25	0.018585	0.929270	0.038763	4860	5	31
		1.50	0.017594	0.879725	0.038444	4941	3	17
		2.00	0.016709	0.835465	0.036966	5344	2	10

with just a few more (relatively speaking) samples, much fewer than the amount taken at the next iteration, for the chosen value of θ. Clearly there is always an θ that is optimal for a specific graph and a specific desired value ε: if we assume that the algorithm needs m samples to get a quality guarantee $\varepsilon \leq \bar{\varepsilon}$, then the optimal θ is the that results in a sample size m_i closest to but larger than m. In general though, a lower θ is usually to be preferred: in some cases the difference is of thousands fewer samples taken when considering a lower θ than a larger one (e.g., on email-Enron with ab, for $\bar{\varepsilon} = 0.01$).

Another observation is that the ε computed by Bavarian-P is much smaller than the one computed by *static* sampling algorithm presented in previous works, as we also already observed for Bavarian in Fig. 4 and comment on it. Let us now explain the consequence of this fact for progressive sampling. Bavarian-P "pays" for its being a progressive sampling algorithm through an union bound over the iterations (Sect. 5.2), which makes the computed ε larger than if we directly used a sample of the size equal to the sample size used at the last iteration. The static sampling algorithm presented in previous work do not have to pay for this price. Nevertheless, as can be observed from Tables 3 and 4 (and all the others in App. A.2), in *all* cases, the ε returned by the

progressive algorithm BAVARIAN-P is much smaller than the ε returned by previous *static* algorithms. By comparing this "prev. work ε" with $\bar{\varepsilon}$, one can also see that the former is larger than the latter, which means that progressive sampling versions of the previous works would not be able to stop as early as BAVARIAN-P, thus proving its superiority. This advantage of BAVARIAN-P is due to the use of the k-MCERA and the refined bounds on it that use the empirical wimpy variance.

In conclusion, we can say that the progressive sampling algorithm achieves its goals of freeing the user from having to select a sample size "blindly" and allows them to express their desired quality guarantees $\bar{\varepsilon}$ instead, while being much more efficient (by stopping earlier) than existing algorithms, both progressive and static.

7 CONCLUSIONS

We present BAVARIAN, a suite of algorithms for approximating the BC of all vertices in a graph, with stringent probabilistic quality guarantees. BAVARIAN achieves a better trade-off between sample size and approximation quality, thanks to the use of k-MCERAs and their variance-aware bounds, which are of independent interest. The unifying framework offered by BAVARIAN allows for a fair comparison of different sampling-based estimators for BC, and for us to generalize sample-complexity results that held only for one estimator. The results of our experimental evaluation of BAVARIAN show it guarantees much better approximations at smaller sample sizes, i.e., faster, than the previous state of the art. A few MC-trials (i.e., low k) are sufficient to get a low error bound. The ab estimator stands out, in most cases, as more efficient than the others in terms of "unit of guarantee" per unit of time.

ACKNOWLEDGMENTS

Part of this work is supported by the National Science Foundation awards RI-1813444 (https://www.nsf.gov/awardsearch/showAward?AWD_ID=1813444) and IIS-2006765 (https://www.nsf.gov/awardsearch/showAward?AWD_ID=2006765) and by the DARPA/ARFL grant FA8750.

REFERENCES

[1] Ziyad AlGhamdi, Fuad Jamour, Spiros Skiadopoulos, and Panos Kalnis. 2017. A Benchmark for Betweenness Centrality Approximation Algorithms on Large Graphs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management (SSDBM '17)*. ACM.

[2] Josh Alman and Virginia Vassilevska Williams. 2021. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 522–539.

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

[4] 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.

[5] Peter L. Bartlett and Shahar Mendelson. 2002. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research* 3, Nov (2002), 463–482.

[6] Alex Bavelas. 1950. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America* 22, 6 (1950), 725–730.

[7] George Bennett. 1962. Probability inequalities for the sum of independent random variables. *J. Amer. Statist. Assoc.* 57, 297 (1962), 33–45.

[8] 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.

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

[10] 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.

[11] Paolo Boldi and Sebastiano Vigna. 2014. Axioms for centrality. *Internet Mathematics* 10, 3-4 (2014), 222–262.

[12] Francesco Bonchi, Gianmarco De Francisci Morales, and Matteo Riondato. 2016. Centrality measures on big graphs: Exact, approximated, and distributed algorithms. In *Proceedings of the 25th international conference companion on world wide web*. 1017–1020.
[13] Michele Borassi and Emanuele Natale. 2019. KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation. *Journal of Experimental Algorithmics* 24, 1 (2019), 1–35.
[14] Stephen P. Borgatti and Martin G. Everett. 2006. A Graph-theoretic perspective on centrality. *Soc. Netw.* 28, 4 (2006), 466–484.
[15] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. 2000. A sharp concentration inequality with application. *Random Structures & Algorithms* 16, 3 (2000), 277–292.
[16] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. 2013. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press.
[17] Olivier Bousquet. 2002. A Bennett concentration inequality and its application to suprema of empirical processes. *Comptes Rendus Mathematique* 334, 6 (2002), 495–500.
[18] Ulrik Brandes. 2001. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology* 25, 2 (2001), 163–177.
[19] Ulrik Brandes. 2008. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks* 30, 2 (2008), 136–145.
[20] Ulrik Brandes and Christian Pich. 2007. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos* 17, 7 (2007), 2303–2318.
[21] Mostafa Haghir Chehreghani, Albert Bifet, and Talel Abdessalem. 2018. Efficient Exact and Approximate Algorithms for Computing Betweenness Centrality in Directed Graphs. In *Advances in Knowledge Discovery and Data Mining*, Dinh Phung, Vincent S. Tseng, Geoffrey I. Webb, Bao Ho, Mohadeseh Ganji, and Lida Rashidi (Eds.). Springer International Publishing, Cham, 752–764.
[22] Flavio Chierichetti, Anirban Dasgupta, Ravi Kumar, Silvio Lattanzi, and Tamás Sarlós. 2016. On sampling nodes in a network. In *Proceedings of the 25th International Conference on World Wide Web*. 471–481.
[23] Flavio Chierichetti and Shahrzad Haddadan. 2018. On the Complexity of Sampling Vertices Uniformly from a Graph. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*.
[24] Cyrus Cousins, Shahrzad Haddadan, and Eli Upfal. 2020. Making mean-estimation more efficient using an MCMC trace variance approach: DynaMITE. *arXiv:2011.11129* (2020).
[25] Cyrus Cousins and Matteo Riondato. 2020. Sharp uniform convergence bounds through empirical centralization. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 15123–15132. https://proceedings.neurips.cc/paper/2020/file/ac457ba972fb63b7994befc83f774746-Paper.pdf
[26] Cyrus Cousins, Chloe Wohlgemuth, and Matteo Riondato. 2021. Betweenness Centrality Approximation with Variance-Aware Rademacher Averages. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '21)*. ACM, 196–206.
[27] Alane M. de Lima, Murilo V. G. da Silva, and André L. Vignatti. 2020. Estimating the Percolation Centrality of Large Networks through Pseudo-dimension Theory. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.
[28] Lorenzo De Stefani and Eli Upfal. 2019. A Rademacher Complexity Based Method for Controlling Power and Confidence Level in Adaptive Statistical Analysis. *CoRR abs 1910.03493* (2019).
[29] Shlomi Dolev, Yuval Elovici, and Rami Puzis. 2010. Routing betweenness centrality. *J. ACM* 57, 4, Article 25 (May 2010), 27 pages.
[30] 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.
[31] Changjun Fan, Li Zeng, Yuhui Ding, Muhao Chen, Yizhou Sun, and Zhong Liu. 2019. Learning to Identify High Betweenness Centrality Nodes from Scratch. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM. https://doi.org/10.1145/3357384.3357979
[32] Linton C. Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* 40 (1977), 35–41.
[33] 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.
[34] Jay Ghurye and Mihai Pop. 2016. Better Identification of Repeats in Metagenomic Scaffolding. In *WABI 2016: Algorithms in Bioinformatics*. Springer, 174–184.
[35] Oded Green, Robert 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.
[36] David Haussler. 1995. Sphere packing numbers for subsets of the Boolean n-cube with bounded Vapnik-Chervonenkis dimension. *Journal of Combinatorial Theory, Series A* 69, 2 (1995), 217–232.

[37] 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.
[38] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. American Statistical Assoc.* 58, 301 (1963), 13–30.
[39] 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.
[40] George H. John and Pat Langley. 1996. Static Versus Dynamic Sampling for Data Mining. In *Proc. 2nd Int. Conf. Knowl. Disc. Data Mining (KDD '96)*. The AAAI Press, Menlo Park, CA, USA, 367–370.
[41] 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.
[42] Liran Katzir, Edo Liberty, Oren Somekh, and Ioana A Cosma. 2014. Estimating sizes of social networks via biased sampling. *Internet Mathematics* 10, 3-4 (2014), 335–359.
[43] Vladimir Koltchinskii. 2001. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory* 47, 5 (July 2001), 1902–1914.
[44] Aryeh Kontorovich. 2016. Agnostic PAC lower bound. https://www.cs.bgu.ac.il/~asml162/wiki.files/agnostic-pac-lb.pdf
[45] 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.
[46] 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.
[47] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.
[48] Yixia Li, Shudong Li, Yanshan Chen, Peiyan He, Xiaobo Wu, and Weihong Han. 2019. Electric Power Grid Invulnerability Under Intentional Edge-Based Attacks. In *DependSys 2019: Dependability in Sensor, Cloud, and Big Data Systems and Applications*. Springer Singapore, 454–461.
[49] Yeon-sup Lim, Daniel S. Menasche, Bruno Ribeiro, Don Towsley, and Prithwish Basu. 2011. Online estimating the k central nodes of a network. In *IEEE Network Science Workshop (NSW'11)*. 118–122.
[50] Arun S. Maiya and Tanya Y. Berger-Wolf. 2010. Online Sampling of High Centrality Individuals in Social Networks. In *Advances in Knowl. Disc. Data Mining*. Springer Berlin Heidelberg, 91–98.
[51] John Matta, Gunes Ercal, and Koushik Sinha. 2019. Comparing the speed and accuracy of approaches to betweenness centrality approximation. *Computational Social Networks* 6, 1 (2019), 2.
[52] Adam McLaughlin and David A. Bader. 2014. Scalable and High Performance Betweenness Centrality on the GPU. *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis* (Nov 2014).
[53] Mark E. J. Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E* 69 (Feb. 2004). Issue 2.
[54] Tore Opsahl, Filip Agneessens, and John Skvoretz. 2010. Node centrality in weighted networks: Generalizing degree and shortest paths. *Soc. Netw.* 32, 3 (2010), 245–251.
[55] Leonardo Pellegrina, Cyrus Cousins, Fabio Vandin, and Matteo Riondato. 2020. MCRapper: Monte-Carlo Rademacher Averages for Poset Families and Approximate Pattern Mining. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 2165–2174. https://doi.org/10.1145/3394486.3403267
[56] Jürgen Pfeffer and Kathleen M. Carley. 2012. k-Centralities: local approximations of global measures based on shortest paths. In *Proc. 21st Int. Conf. Companion on World Wide Web (WWW '12 Companion)*. ACM, New York, NY, USA, 1043–1050.
[57] David Pollard. 1984. *Convergence of stochastic processes*. Springer-Verlag.
[58] 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.
[59] Dimitrios Prountzos and Keshav Pingali. 2013. Betweenness centrality: algorithms and implementations. In *Proc. 18th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming* (Shenzhen, China) *(PPoPP '13)*. ACM, New York, NY, USA, 35–46.
[60] Matteo Riondato and Evgenios M. Kornaropoulos. 2016. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery* 30, 2 (2016), 438–475.
[61] Matteo Riondato and Eli Upfal. 2018. ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. *ACM Trans. Knowl. Disc. from Data* 12, 5 (2018), 61.
[62] Ahmet Erdem Sarıyüce, Kamer Kaya, Erik Saule, and Ümit V. Çatalyürek. 2017. Graph Manipulations for Fast Centrality Computation. *ACM Transactions on Knowledge Discovery from Data* 11, 3 (2017), 1–25.

[63] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press.
[64] Nathan Srebro and Karthik Sridharan. 2010. Note on refined Dudley integral covering number bound. (2010). http://www.cs.cornell.edu/~sridharan/dudley.pdf.
[65] Christian L. Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. 2016. NetworKit: An Interactive Tool Suite for High-Performance Network Analysis. *Network Science* 4, 4 (2016).
[66] Volker Strassen. 1969. Gaussian elimination is not optimal. *Numerische mathematik* 13, 4 (1969), 354–356.
[67] Vladimir N. Vapnik. 1998. *Statistical learning theory.* Wiley.
[68] Vladimir N. Vapnik and Alexey J. Chervonenkis. 1971. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications* 16, 2 (1971), 264–280.
[69] Yuichi Yoshida. 2014. Almost Linear-time Algorithms for Adaptive Betweenness Centrality Using Hypergraph Sketches. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14).* ACM, New York, NY, USA, 1416–1425.

A ADDITIONAL MATERIALS

A.1 Missing Proofs

We now seek to prove Thm. 3.1. In service of this result, we require a few intermediaries.

Definition A.1. A function $\mathsf{Z} \in \mathcal{D}^m \to \mathbb{R}$ is *self-bounding* with scale $\gamma \geq 0$, for some $\gamma \geq 0$ if for each $j = 1, \ldots, m$, there exists a function $\mathsf{Z}_j \in \mathcal{D}^m \to \mathbb{R}$ such that, for any $\mathcal{S} \in \mathcal{D}^m$ it holds that

(1) $\mathsf{Z}_j(\mathcal{S})$ does not depend on the j-th component $\mathcal{S}_j$ of $\mathcal{S}$; and
(2) it holds $\mathsf{Z}_j(\mathcal{S}) \leq \mathsf{Z}(\mathcal{S}) \leq \mathsf{Z}_j(\mathcal{S}) + \gamma$;

Additionally, the functions Z_j, $j = 1, \ldots, m$, must be such that, for any $\mathcal{S} \in \mathcal{D}^m$, it holds

$$\sum_{j=1}^{m} \left(\mathsf{Z}(\mathcal{S}) - \mathsf{Z}_j(\mathcal{S}) \right) \leq \mathsf{Z}(\mathcal{S}) \ .$$

Theorem A.2. *Let Z be a function from $\mathcal{D}^m$ to $\mathbb{R}$ that is self-bounding with scale γ. Let $\delta \in (0, 1)$, and $\mathcal{S}$ be a collection of m i.i.d. samples from $\mathcal{D}$. With prob. $\geq 1 - \delta$ over the choice of $\mathcal{S}$, it holds*

$$\mathbb{E}_{\mathcal{S}} \left[\mathsf{Z}(\mathcal{S}) \right] \leq \mathsf{Z}(\mathcal{S}) + \frac{2\gamma}{3} \ln \frac{1}{\delta} + \sqrt{\left(\frac{\gamma}{\sqrt{3}} \ln \frac{1}{\delta} \right)^2 + 2\gamma \mathsf{Z}(\mathcal{S}) \ln \frac{1}{\delta}} \ . \tag{23}$$

This result is derived using the *sub-gamma* form in [16, Sect. 2.1] of the stronger *sub-Poisson* self-bounding function inequality [15, Thm. 1]. This form appeared as [25, Thm. 6].

Theorem A.3. *With probability at least $1 - \eta$ over the choice of $\mathcal{S}$, it holds that*

$$\mathsf{v} \leq \beta + \frac{2 \ln \frac{5}{\eta}}{3m} + \sqrt{\left(\frac{\ln \frac{5}{\eta}}{\sqrt{3}m} \right)^2 + \frac{2\beta \ln \frac{5}{\eta}}{m}} \ .$$

Proof. We show this result by showing that an upper-bound on the raw wimpy variance is self-bounding with scale $\frac{1}{m}$. First note that by Jensen's inequality,

$$m\mathsf{v} = \sup_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{S}} \left[\sum_{i=1}^{m} \left(f(\mathcal{S}_i) \right)^2 \right] \leq \mathbb{E}_{\mathcal{S}} \left[\sup_{f \in \mathcal{F}} \left(f(\mathcal{S}_i) \right)^2 \right] = m \mathbb{E}_{\mathcal{S}} \left[\beta \right] \ .$$

The remainder of this proof thus seeks to bound the r.h.s., as this yields a bound on v, as desired.

Suppose $\mathcal{S} \in \mathcal{D}^m$, let $\mathcal{S}_{\setminus j}$ denote the vector $\mathcal{S}$, missing the jth component, note that

$$\mathsf{Z}(\mathcal{S}) = \sup_{f \in \mathcal{F}} \sum_{i=1}^{m} f^2(\mathcal{S}_i) \ ,$$

and take

$$\mathsf{Z}_j(\mathcal{S}) \doteq \sup_{f\in\mathcal{F}} \sum_{i=1,\ i\neq j}^{m} f^2(\mathcal{S}_i) = \sup_{f\in\mathcal{F}} \left(\sum_{i=1}^{m} f^2(\mathcal{S}_i)\right) - f^2(\mathcal{S}_j) \ .$$

Clearly we have that $0 \leq \mathsf{Z}(\mathcal{S})$ and $\forall j \in 1\ldots m:\ \mathsf{Z}(\mathcal{S}) - 1 \leq \mathsf{Z}_j(\mathcal{S}) \leq \mathsf{Z}(\mathcal{S})$, essentially because $\mathcal{F}^2(\mathcal{D}) \subseteq [0,1]$. We now show that $\mathsf{Z}(\cdot)$ obeys $(1,0)$-*self-boundedness*.

$$\begin{aligned}
\sum_{j=1}^{m} \mathsf{Z}(\mathcal{S}) - \mathsf{Z}_j(\mathcal{S}) &= m\mathsf{Z}(\mathcal{S}) - \sum_{j=1}^{m} \sup_{f\in\mathcal{F}} \left(\left(\sum_{i=1}^{m} f^2(\mathcal{S}_i)\right) - f^2(\mathcal{S}_j)\right) \\
&\leq m\mathsf{Z}(\mathcal{S}) - \sup_{f\in\mathcal{F}} \sum_{j=1}^{m} \left(\sum_{i=1}^{m} f^2(\mathcal{S}_i)\right) - f^2(\mathcal{S}_j) \\
&= m\mathsf{Z}(\mathcal{S}) - \sup_{f\in\mathcal{F}} (m-1) \sum_{i=1}^{m} f^2(\mathcal{S}_i) \\
&= m\mathsf{Z}(\mathcal{S}) - (m-1)\mathsf{Z}(\mathcal{S}) = \mathsf{Z}(\mathcal{S})
\end{aligned}$$

We may now conclude that $\mathsf{Z}(\cdot)$ is a self-bounding function. It thus follows that $\beta = \frac{1}{m}\mathsf{Z}(\mathcal{S})$ is self-bounding with scale $\frac{1}{m}$ The remainder of the result then follows via Thm. A.2. □

Theorem 3.1. *Let $\eta \in (0,1)$, and define the quantities*

$$\gamma \doteq \beta + \frac{2\ln\frac{5}{\eta}}{3m} + \sqrt{\left(\frac{\ln\frac{5}{\eta}}{\sqrt{3}m}\right)^2 + \frac{2\beta\ln\frac{5}{\eta}}{m}}, \tag{10}$$

$$\rho \doteq \hat{\mathsf{R}}^k_m(\mathcal{F},\mathcal{S},\mathbf{\Lambda}) + \frac{2\ln\frac{5}{\eta}}{3km} + \sqrt{\frac{4\beta\ln\frac{5}{\eta}}{km}},$$

$$r \doteq \rho + \frac{\ln\frac{5}{\eta}}{3m} + \sqrt{\left(\frac{\ln\frac{5}{\eta}}{2\sqrt{3}m}\right)^2 + \frac{\rho\ln\frac{5}{\eta}}{m}} \ ,$$

$$\varepsilon \doteq 2r + \frac{\ln\frac{5}{\eta}}{3m} + \sqrt{\frac{2\,(\gamma+4r)\ln\frac{5}{\eta}}{m}} \ , \tag{11}$$

Then, with probability at least $1-\eta$ over the choice of $\mathcal{S}$ and $\mathbf{\Lambda}$, it holds that

$$\mathsf{SD}(\mathcal{F},\mathcal{S}) \leq \varepsilon \ .$$

Proof. We show this result as a union bound over *exponential tail bounds* (hence the $\ln\frac{5}{\eta}$ in all bounds) on five events, each dependent on the draw of $\mathbf{\Lambda}$ and $\mathcal{S}$. We now enumerate these bounds, and note that where each bound depends on an unknown quantity, we substitute an upper-bound based on an empirical quantity.

The first is on γ, which upper-bounds the wimpy-variance via Thm. A.3. The next is on ρ, which holds via the Monte-Carlo estimation theorem [25, Thm. 5]. The third is on r, and holds via the self-bounding function inequality (Thm. A.2), using the well-known scale $1/2m$-self-bounding property of Rademacher averages [16, Ex. 3.12]. Finally, the fourth and fifth tail bounds are on ε (upper and lower tails), both via Bousquet's inequality [17], which bounds the SD as stated.

These five bounds then all hold simultaneously with probability $\geq 1 - 5\cdot\frac{\eta}{5} = 1-\eta$ via the *union bound*, yielding the desideratum. □

Relating Variances and Rademacher Averages of Estimators. We now show Thms. 4.1 and 4.2. Both results follow by expressing the BC as a nested sequence of conditional expectations over m samples (as in eq. (12)), and then commuting expectations outward through squaring (raw variance) and supremum (Rademacher average) operators.

Theorem 4.1. *For the raw variances of the three estimators it holds*

$$\forall w \in V : \ \left(\mathsf{b}(w)\right)^2 \leq \mathbb{E}_{\pi_{\mathsf{bp}}}\left[(f_w(u))^2\right] \leq \mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(u,v))^2\right] \leq \mathbb{E}_{\pi_{\mathsf{rk}}}\left[(f_w(p))^2\right] = \mathsf{b}(w) \ .$$

The same relationship extends to the wimpy variances of the families of each estimator.

Proof. Suppose random variables u, v and p, representing a pair of distinct vertices, uniformly distributed among such pairs, and a shortest path between them, uniformly distributed among the SPs from u to v, if at least one such path exists, and p is the special empty path (u, v) otherwise (the empty path is not really a path, but its salient characteristic is to have no interval vertices). Now, w.l.o.g., fix some $v \in V$. It then holds that

$$\mathbb{E}_{\pi_{\mathsf{bp}}}\left[(f_w(u))^2\right] = \mathbb{E}_u\left[f_w^2(u)\right] = \mathbb{E}_u\left[\left(\mathbb{E}_{v\neq u}\left[f_w(u,v)\right]\right)^2\right] \leq \mathbb{E}_{u,v}\left[f_w^2(u,v)\right] = \mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(u,v))^2\right],$$

which gives the second inequality from the left in the thesis, and the last steps follows from Jensen's inequality. Continuing from the last step, we obtain

$$\mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(u,v))^2\right] = \mathbb{E}_{u,v}\left[\mathbb{E}_{p\in\pi_{\mathsf{rk}}|u,v}\left[f_w^2(p)\right]\right] \leq \mathbb{E}_{u,v,p}\left[f_w^2(p)\right] = \mathbb{E}_p\left[f_w^2(p)\right] = \mathbb{E}_{\pi_{\mathsf{rk}}}\left[(f_w(p))^2\right],$$

where we again used Jensen's inequality, to obtain the third inequality from the left. To see the leftmost inequality, note that each estimator shares the common expectation $\mathsf{b}(w)$, and for any random variable X bounded on $[0, 1]$, it holds that

$$\left(\mathbb{E}\left[X\right]\right)^2 \leq \mathbb{E}\left[X^2\right] \leq \mathbb{E}\left[X\right] \ ,$$

which completes the results. □

Theorem 4.2. *For the Rademacher averages of the three estimators it holds*

$$\mathsf{R}_m(\mathcal{F}_{\mathsf{bp}}, \pi_{\mathsf{bp}}) \leq \mathsf{R}_m(\mathcal{F}_{\mathsf{ab}}, \pi_{\mathsf{ab}}) \leq \mathsf{R}_m(\mathcal{F}_{\mathsf{rk}}, \pi_{\mathsf{rk}}) \ .$$

Proof. The proof proceeds largely as in Thm. 4.1, except now our random variables are $\boldsymbol{u}, \boldsymbol{v}$, representing *vectors of* (u_i, v_i) pairs of distinct vertices, each uniformly distributed among such pairs, and $\boldsymbol{p}$, representing a vector of SPs p_i from u_i to v_i (or $p_i = (u_i, v_i)$, if there is no SP from u_i to v_i), each uniformly distributed among the SPs connecting u_i to v_i. Furthermore, as we are bounding Rademacher averages, we also have a vector $\boldsymbol{\lambda}$ of m i.i.d. Rademacher random variables.

With this setup, we observe the result, as

$$\begin{aligned}\mathsf{R}_m(\mathcal{F}_{\mathsf{bp}}, \pi_{\mathsf{bp}}) &= \mathbb{E}_{\boldsymbol{u},\boldsymbol{\lambda}}\left[\sup_{f\in\mathcal{F}_{\mathsf{bp}}} \frac{1}{m}\sum_{i=1}^{m} f(u_i)\lambda_i\right] = \mathbb{E}_{\boldsymbol{u},\boldsymbol{\lambda}}\left[\sup_{w\in V} \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_{v_i\neq u_i}\left[f_w(u_i,v_i)\right]\lambda_i\right] \\ &\leq \mathbb{E}_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda}}\left[\sup_{w\in V} \frac{1}{m}\sum_{i=1}^{m} f_w(u_i,v_i)\lambda_i\right] = \mathsf{R}_m(\mathcal{F}_{\mathsf{ab}}, \pi_{\mathsf{ab}})\end{aligned}$$

which follows from Jensen's inequality and gives the leftmost inequality in the thesis. Continuing from the above we then have

$$\begin{aligned}\mathsf{R}_m(\mathcal{F}_{\mathsf{ab}}, \pi_{\mathsf{ab}}) &= \mathbb{E}_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda}}\left[\sup_{w\in V} \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_{p\in\pi_{\mathsf{rk}}|u_i,v_i}\left[f_w(p)\right]\lambda_i\right] \\ &\leq \mathbb{E}_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{p},\boldsymbol{\lambda}}\left[\sup_{w\in V} \frac{1}{m}\sum_{i=1}^{m} f_w(p_i)\lambda_i\right] = \mathbb{E}_{\boldsymbol{p},\boldsymbol{\lambda}}\left[\sup_{w\in V} \frac{1}{m}\sum_{i=1}^{m} f_w(p_i)\lambda_i\right] = \mathsf{R}_m(\mathcal{F}_{\mathsf{rk}}, \pi_{\mathsf{rk}}),\end{aligned}$$

where again we used Jensen's inequality to obtain the rightmost inequality in the thesis. □

Lemma 4.3. *For any $w \in V$, it holds*

$$\mathsf{b}(w)\mathbb{E}_{\pi_{\mathsf{ab}}}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\middle|\sigma_{vu}(w) > 0\right] \le \mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(u,v))^2\right] \le \mathsf{b}(w)\max_{v,u\neq v}\frac{\sigma_{vu}(w)}{\sigma_{vu}} \quad .$$

Proof. We first show the lower bound. Using the definition of $f_w(v,u)$ from (4), and then the law of total expectation and Jensen's inequality, we get

$$\begin{aligned}
\mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(u,v))^2\right] &= \mathbb{E}_{\pi_{\mathsf{ab}}}\left[\left(\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right)^2\right] = \Pr(\sigma_{vu}(w) > 0)\mathbb{E}_{\pi_{\mathsf{ab}}}\left[\left(\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right)^2\middle|\sigma_{vu}(w) > 0\right] \\
&\ge \Pr(\sigma_{vu}(w) > 0)\left(\mathbb{E}_{\pi_{\mathsf{ab}}}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\middle|\sigma_{vu}(w) > 0\right]\right)^2 \\
&= \mathsf{b}(w)\mathbb{E}_{\pi_{\mathsf{ab}}}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\middle|\sigma_{vu}(w) > 0\right],
\end{aligned}$$

where the last step is another application of the law of total expectation.

Now we show the upper bound. Using the definition of $f_w(v,u)$ and monotonicity, we obtain

$$\mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(u,v))^2\right] = \mathbb{E}_{\pi_{\mathsf{ab}}}\left[\left(\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right)^2\right] \le \mathbb{E}_{\pi_{\mathsf{ab}}}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right]\max_{v,u\neq v}\frac{\sigma_{vu}(w)}{\sigma_{vu}} = \mathsf{b}(w)\max_{u,v\neq u}\frac{\sigma_{vu}(w)}{\sigma_{vu}} \quad .$$

□

We now show Lemma 4.4.

Lemma 4.4. *For any $w \in V$, let v be drawn uniformly at random from V, and u be drawn uniformly at random from $V \setminus \{v\}$, it holds*

$$\mathsf{b}(w)\mathbb{E}_{\pi_{\mathsf{bp}}}\left[\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right]\middle|\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] > 0\right] \le \mathbb{E}_{\pi_{\mathsf{bp}}}\left[(f_w(v))^2\right] \le \mathsf{b}(w)\max_{v}\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] \quad .$$

Proof. From the definition of $f_w(v)$ in (5), it holds

$$f_w(v) = \mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] \quad . \tag{24}$$

We first show the lower bound. Using the definition of $f_w(v)$ from (5), and then the law of total expectation and Jensen's inequality, we get

$$\begin{aligned}
&\mathbb{E}_{\pi_{\mathsf{bp}}}\left[(f_w(v))^2\right] = \mathbb{E}_{\pi_{\mathsf{bp}}}\left[\left(\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right]\right)^2\right] \\
&= \Pr_v\left(\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] > 0\right)\mathbb{E}_{\pi_{\mathsf{bp}}}\left[\left(\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right]\right)^2\middle|\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] > 0\right] \\
&\ge \Pr_v\left(\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] > 0\right)\left(\mathbb{E}_{\pi_{\mathsf{bp}}}\left[\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right]\middle|\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] > 0\right]\right)^2 \\
&= \mathsf{b}(w)\mathbb{E}_{\pi_{\mathsf{bp}}}\left[\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right]\middle|\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] > 0\right] \quad .
\end{aligned}$$

where the last step is another application of the law of total expectation.

We now show the upper bound. Using eq. (24), and simple monotonicity, we get

$$\begin{aligned}\mathbb{E}_{\pi_{\mathsf{bp}}}\left[(f_w(v))^2\right] &= \mathbb{E}_{\pi_{\mathsf{bp}}}\left[\left(\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right]\right)^2\right] \\ &\leq \mathbb{E}_{\pi_{\mathsf{bp}}}\left[\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right]\right]\max_{v\in V}\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] \\ &= \mathsf{b}(w)\max_{v\in V}\mathbb{E}_{u|v}\left[\frac{\sigma_{vu}(w)}{\sigma_{vu}}\right] \quad . \end{aligned}$$

□

Lemma 4.5. *For any $w \in V$, it holds*

$$\frac{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{ab}}(w)]}{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{rk}}(w)]} \leq \max_{v,u\neq v}\frac{\sigma_{vu}(w)}{\sigma_{vu}} \qquad (\leq 1) \quad .$$

Proof. Using the fact that the estimators are sum of i.i.d. random variables, and then applying fundamental properties of the variance, it is easy to see that

$$\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{ab}}(w)] = \frac{1}{m}\left(\mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(v,u))^2\right] - (\mathsf{b}(w))^2\right)$$

and

$$\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{rk}}(w)] = \frac{1}{m}\left(\mathbb{E}_{\pi_{\mathsf{rk}}}\left[(f_w(p))^2\right] - (\mathsf{b}(w))^2\right) \quad .$$

A consequence of the above and of Lemma 4.3 is that $\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{ab}}(w)] \leq \mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{rk}}(w)]$. It also holds that they are both non-negative, and obviously $(\mathsf{b}(w))^2 \geq 0$. So we can use the fact that

$$\frac{a}{b} \leq \frac{a+c}{b+c} \text{ for any } 0 < a \leq b, 0 \leq c$$

and Lemma 4.3 to complete the proof as

$$\frac{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{ab}}(w)]}{\mathbb{V}[\tilde{\mathsf{b}}_{\mathsf{rk}}(w)]} \leq \frac{\mathbb{E}_{\pi_{\mathsf{ab}}}\left[(f_w(v,u))^2\right]}{\mathbb{E}_{\pi_{\mathsf{rk}}}\left[(f_w(p))^2\right]} \leq \max_{v,u\neq v}\frac{\sigma_{vu}(w)}{\sigma_{vu}} \quad .$$

□

Rademacher Averages of BC Estimators. In service of Thm. 5.4, we first show a technical lemma.

Lemma A.4. *Suppose a function family $\mathcal{F}$ with codomain $\{0,1\}$ of VC dimension d, and a sample $\mathcal{S}$ such that $\mathcal{F}$ has empirical wimpy variance β on $\mathcal{S}$. Then*

$$\hat{\mathsf{R}}_m(\mathcal{F},\mathcal{S}) \leq \sqrt{\frac{288d\beta\ln\frac{e^3}{\sqrt{\beta}}}{m}} \quad .$$

Furthermore, for some universal constant $C < 262$, it holds that

$$\hat{\mathsf{R}}_m(\mathcal{F},\mathcal{S}) \leq \sqrt{\frac{Cd}{m}} \quad .$$

Proof. We first state a few standard results that we use to show the desideratum. The entropy integral theorem [64] states that

$$\hat{\mathsf{R}}_m(\mathcal{F},\mathcal{S}) \leq 12\int_{\lim 0^+}^{\sqrt{\beta}}\sqrt{\frac{\mathcal{N}_2(\mathcal{F},\gamma)}{m}}\,\mathrm{d}\gamma \quad ,$$

where as usual β is the empirical raw wimpy variance. Haussler's bound [36] on the γ-VC entropy [see 16, lemma 13.6] states that for a function family $\mathcal{F}$ with codomain $\{0,1\}$ of VC dimension d, it holds that

$$\ln \mathcal{N}_2(\mathcal{F},\gamma) \leq 2d \ln \frac{e^2}{\gamma} \ .$$

Putting these results together yields the following inequality chain.

$$\begin{aligned}
\hat{\mathsf{R}}_m(\mathcal{F},\mathcal{S}) &\leq 12 \int_{\lim_{0^+}}^{\sqrt{\beta}} \sqrt{\frac{\mathcal{N}_2(\mathcal{F},\gamma)}{m}}\,\mathrm{d}\gamma && \text{Entropy Integral Theorem} \\
&\leq 12 \int_{\lim_{0^+}}^{\sqrt{\beta}} \sqrt{\frac{2d \ln \frac{e^2}{\gamma}}{m}}\,\mathrm{d}\gamma && \text{Haussler Bound} \\
&= 12\sqrt{\frac{2d}{m}} \int_{\lim_{0^+}}^{\sqrt{\beta}} \sqrt{\ln \frac{e^2}{\gamma}}\,\mathrm{d}\gamma \\
&= 12\sqrt{\frac{2d}{m}} \cdot \left(\gamma \sqrt{\ln \tfrac{e^2}{\gamma}} - \frac{e^2\sqrt{\pi}}{2} \operatorname{erf}\left(\sqrt{\ln \tfrac{e^2}{\gamma}} \right) \Big|_{\lim_{\gamma\to 0^+}}^{\gamma=\sqrt{\beta}} \right) && \text{Improper Integration} \\
&= 12\sqrt{\frac{2d}{m}} \cdot \left(\sqrt{\beta} \sqrt{\ln \tfrac{e^2}{\sqrt{\beta}}} + \frac{e^2\sqrt{\pi}}{2} \left(1 - \operatorname{erf}\left(\sqrt{\ln \tfrac{e^2}{\sqrt{\beta}}} \right) \right) \right) \\
&\leq 12\sqrt{\frac{2d}{m}} \sqrt{\beta \ln \frac{e^3}{\sqrt{\beta}}} \ . && \forall u \geq 1 : \ \sqrt{\ln u} + u\frac{\sqrt{\pi}}{2} \operatorname{erfc}(\sqrt{\ln u}) \leq \sqrt{\ln(eu)} \\
&= \sqrt{\frac{288 d \beta \ln \frac{e^3}{\sqrt{\beta}}}{m}} \ .
\end{aligned}$$

This yields the first inequality.

The second result, with the constant $C < 262$, has a few additional subtleties. First, note that range-centralizing, i.e., translating the function family by $-\frac{1}{2}$, yields a new family with raw wimpy variance $\beta' \leq \frac{1}{4}$. However, by translation invariance, this operation not change empirical Rademacher averages or the covering numbers. Performing this range-centralization step before applying a truncation of the above result yields the second inequality. In particular, the result follows by range-centralizing, taking $\beta' = \frac{1}{4}$, and evaluating the intermediate form

$$12\sqrt{\frac{2d}{m}} \cdot \left(\sqrt{\beta'} \sqrt{\ln \tfrac{e^2}{\sqrt{\beta'}}} + \frac{e^2\sqrt{\pi}}{2} \left(1 - \operatorname{erf}\left(\sqrt{\ln \tfrac{e^2}{\sqrt{\beta'}}} \right) \right) \right) < \sqrt{\frac{262d}{m}} \ .$$

□

We now show Thm. 5.4.

Theorem 5.4. *Let $d \geq \lfloor \log_2(\mathsf{vd}(G) - 2) \rfloor + 1$ or $d = 3$ if G is undirected and there is at most one SP between any pair of vertices in G.*

$$\mathsf{R}_m(\mathsf{rk}) \leq \sqrt{\frac{288 d \beta \ln \frac{e^3}{\sqrt{\beta}}}{m}} \tag{17}$$

Furthermore, for some universal constant $C < 262$, it holds that

$$\mathsf{R}_m(\mathsf{rk}) \leq \sqrt{\frac{Cd}{m}} \ .$$

PROOF. The thesis follows from the fact that d is an upper bound to the VC-dimension [68] ([63, Def. 6.5]) of the BC estimation task on G using the rk estimator, as shown by Riondato and Kornaropoulos [60, Thm. 2], and then application of Lemma A.4. □

A.2 Additional Experimental Results


ε decomposition– cit-HepPh – rk – m =10000
ε decomposition– cit-HepPh – rk – m =20000
ε decomposition– cit-HepPh – rk – m =30000
ε decomposition– cit-HepPh – rk – m =40000
ε decomposition– cit-HepPh – rk – m =50000
ε decomposition– cit-HepPh – ab – m =10000
ε decomposition– cit-HepPh – ab – m =20000
ε decomposition– cit-HepPh – ab – m =30000
ε decomposition– cit-HepPh – ab – m =40000
ε decomposition– cit-HepPh – ab – m =50000
ε decomposition– cit-HepPh – bp – m =10000
ε decomposition– cit-HepPh – bp – m =20000
ε decomposition– cit-HepPh – bp – m =30000
ε decomposition– cit-HepPh – bp – m =40000
ε decomposition– cit-HepPh – bp – m =50000
ε decomposition– p2p-Gnutella31 – rk – m =10000
ε decomposition– p2p-Gnutella31 – rk – m =20000
ε decomposition– p2p-Gnutella31 – rk – m =30000
ε decomposition– p2p-Gnutella31 – rk – m =40000
ε decomposition– p2p-Gnutella31 – ab – m =10000
ε decomposition– p2p-Gnutella31 – ab – m =20000
ε decomposition– p2p-Gnutella31 – ab – m =30000
ε decomposition– p2p-Gnutella31 – ab – m =40000
ε decomposition– p2p-Gnutella31 – bp – m =10000
ε decomposition– p2p-Gnutella31 – bp – m =20000
ε decomposition– p2p-Gnutella31 – bp – m =30000
ε decomposition– p2p-Gnutella31 – bp – m =40000
2 × k-MCERA
to-2 × ERA bound ρ
to-2 × RA bound r
to-SD bound r
error bound ε
MC trials k


ε decomposition– soc-Epinions1 – rk – m =10000
ε decomposition– soc-Epinions1 – rk – m =20000
ε decomposition– soc-Epinions1 – rk – m =30000
ε decomposition– soc-Epinions1 – rk – m =40000
ε decomposition– soc-Epinions1 – rk – m =50000
ε decomposition– soc-Epinions1 – ab – m =10000
ε decomposition– soc-Epinions1 – ab – m =20000
ε decomposition– soc-Epinions1 – ab – m =30000
ε decomposition– soc-Epinions1 – ab – m =40000
ε decomposition– soc-Epinions1 – ab – m =50000
ε decomposition– soc-Epinions1 – bp – m =10000
ε decomposition– soc-Epinions1 – bp – m =20000
ε decomposition– soc-Epinions1 – bp – m =30000
ε decomposition– soc-Epinions1 – bp – m =40000
ε decomposition– soc-Epinions1 – bp – m =50000
ε decomposition– loc-gowalla – rk – m =5000
ε decomposition– loc-gowalla – rk – m =10000
ε decomposition– loc-gowalla – rk – m =20000
ε decomposition– loc-gowalla – rk – m =30000
ε decomposition– loc-gowalla – rk – m =40000
ε decomposition– loc-gowalla – ab – m =5000
ε decomposition– loc-gowalla – ab – m =10000
ε decomposition– loc-gowalla – ab – m =20000
ε decomposition– loc-gowalla – ab – m =30000
ε decomposition– loc-gowalla – ab – m =40000
ε decomposition– loc-gowalla – bp – m =5000
ε decomposition– loc-gowalla – bp – m =10000
ε decomposition– loc-gowalla – bp – m =20000
ε decomposition– loc-gowalla – bp – m =30000
ε decomposition– loc-gowalla – bp – m =40000
ε decomposition– com-dblp.lcc.net – rk – m =3600
ε decomposition– com-dblp.lcc.net – rk – m =7300
ε decomposition– com-dblp.lcc.net – rk – m =11000
ε decomposition– com-dblp.lcc.net – rk – m =14700
ε decomposition– com-dblp.lcc.net – ab – m =3600
ε decomposition– com-dblp.lcc.net – ab – m =7300
ε decomposition– com-dblp.lcc.net – ab – m =11000
ε decomposition– com-dblp.lcc.net – ab – m =14700
2 · k-MCERA
to-2 · ERA bound ρ
to-2 · RA bound r
to-SD bound ε
error bound ε
MC trials k

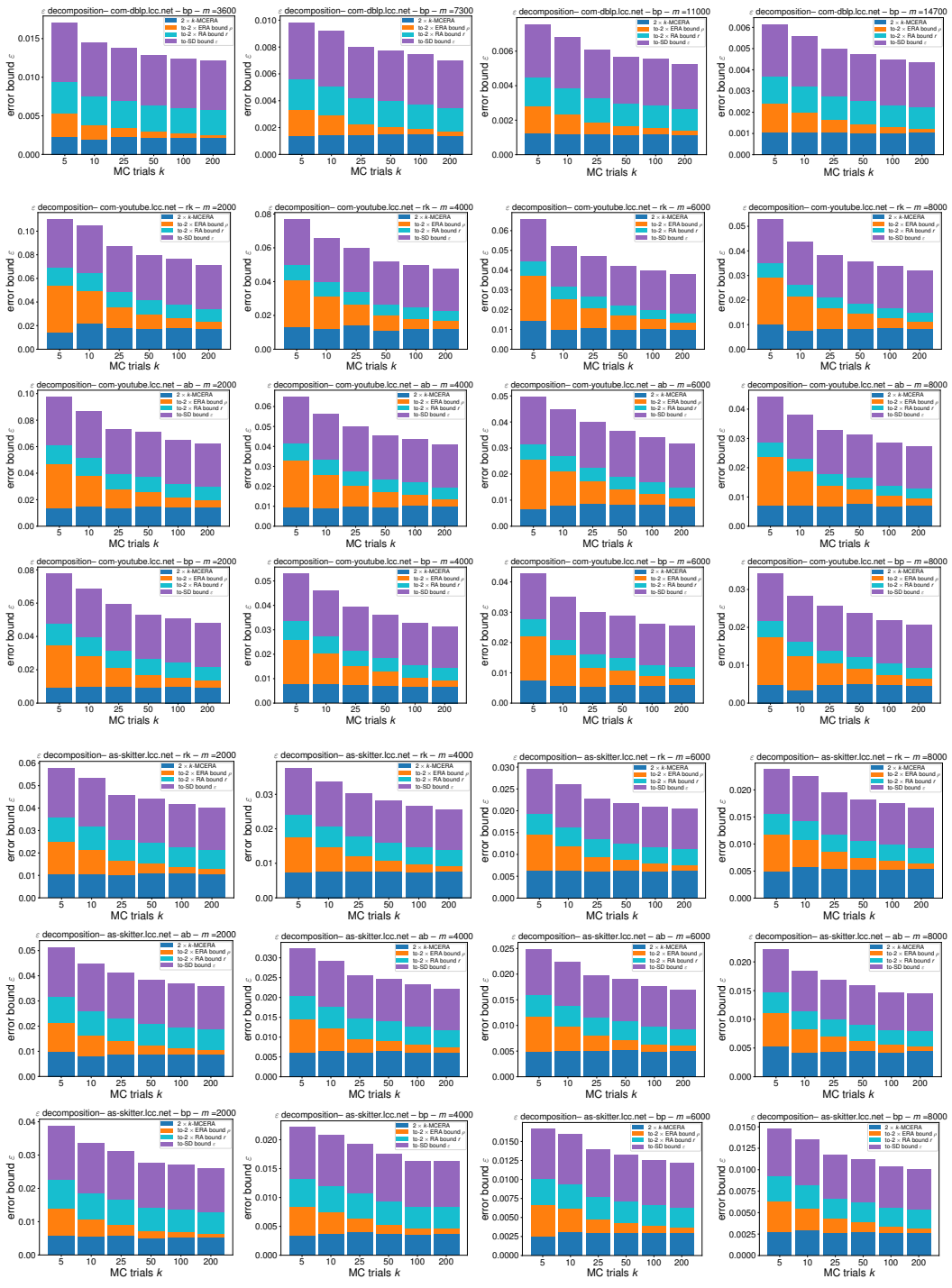


Fig. 15. Error bound ε decomposition.

Table 5. Progressive sampling results — ca-AstroPh

method	$\bar{\varepsilon}$	θ	ε	$\varepsilon/\bar{\varepsilon}$	prev. work ε	m	iter	max. iters
ab	0.010	1.25	0.009324	0.932397	0.022913	15415	7	34
		1.50	0.009150	0.915035	0.022549	14985	4	19
		2.00	0.007464	0.746441	0.018910	21632	3	11
	0.015	1.25	0.013551	0.903433	0.031203	8122	6	32
		1.50	0.011671	0.778053	0.027734	9963	4	18
		2.00	0.014268	0.951207	0.033064	7232	2	11
	0.020	1.25	0.018814	0.940710	0.041593	4860	5	31
		1.50	0.018265	0.913230	0.041506	4941	3	17
		2.00	0.017323	0.866135	0.039001	5344	2	10
bp	0.010	1.25	0.009410	0.941016	0.028464	7892	4	34
		1.50	0.009873	0.987296	0.030985	6660	2	19
		2.00	0.007400	0.740044	0.024314	10816	2	11
	0.015	1.25	0.014326	0.955040	0.039220	4157	3	32
		1.50	0.013479	0.898573	0.038001	4428	2	18
		2.00	0.009427	0.628478	0.029735	7232	2	11
	0.020	1.25	0.018067	0.903335	0.045343	3110	3	31
		1.50	0.016362	0.818105	0.044059	3294	2	17
		2.00	0.019516	0.975815	0.048919	2672	1	10
rk	0.010	1.25	0.009546	0.954649	0.019467	19269	8	34
		1.50	0.008697	0.869663	0.018024	22478	5	19
		2.00	0.008940	0.894019	0.018373	21632	3	11
	0.015	1.25	0.014365	0.957660	0.026819	10153	7	32
		1.50	0.014069	0.937913	0.027073	9963	4	18
		2.00	0.011266	0.751060	0.022469	14464	3	11
	0.020	1.25	0.019257	0.962830	0.034671	6075	6	31
		1.50	0.017255	0.862730	0.031388	7412	4	17
		2.00	0.013454	0.672715	0.026139	10688	3	10

Table 6. Progressive sampling results — cit-HepPh

method	$\bar{\varepsilon}$	θ	ε	$\varepsilon/\bar{\varepsilon}$	prev. work ε	m	iter	max. iters
ab	0.010	1.25	0.009218	0.921844	0.021877	15415	7	34
		1.50	0.009557	0.955690	0.022411	14985	4	19
		2.00	0.007810	0.780952	0.018986	21632	3	11
	0.015	1.25	0.014077	0.938433	0.032104	8122	6	32
		1.50	0.012200	0.813347	0.027622	9963	4	18
		2.00	0.009838	0.655855	0.022974	14464	3	11
	0.020	1.25	0.018906	0.945305	0.039917	4860	5	31
		1.50	0.018704	0.935190	0.039879	4941	3	17
		2.00	0.017789	0.889430	0.038128	5344	2	10
bp	0.010	1.25	0.008746	0.874628	0.029187	7892	4	34
		1.50	0.009898	0.989796	0.031772	6660	2	19
		2.00	0.007345	0.734479	0.024931	10816	2	11
	0.015	1.25	0.014297	0.953160	0.040215	4157	3	32
		1.50	0.013186	0.879067	0.038965	4428	2	18
		2.00	0.009558	0.637214	0.030489	7232	2	11
	0.020	1.25	0.017654	0.882680	0.046494	3110	3	31
		1.50	0.016563	0.828155	0.045177	3294	2	17
		2.00	0.019113	0.955665	0.050160	2672	1	10
rk	0.010	1.25	0.008886	0.888604	0.017412	24087	9	34
		1.50	0.009050	0.905042	0.018024	22478	5	19
		2.00	0.009360	0.935954	0.018373	21632	3	11
	0.015	1.25	0.014637	0.975767	0.026819	10153	7	32
		1.50	0.014955	0.996987	0.027073	9963	4	18
		2.00	0.011968	0.797893	0.022469	14464	3	11
	0.020	1.25	0.018073	0.903650	0.031010	7594	7	31
		1.50	0.017934	0.896710	0.031388	7412	4	17
		2.00	0.014027	0.701360	0.026139	10688	3	10

Table 7. Progressive sampling results — p2p-Gnutella31

method	$\bar{\varepsilon}$	θ	ε	$\varepsilon/\bar{\varepsilon}$	prev. work ε	m	iter	max. iters
ab	0.010	1.25	0.009333	0.933297	0.025424	6313	3	34
		1.50	0.008584	0.858424	0.023901	6660	2	19
		2.00	0.006188	0.618843	0.018545	11072	2	12
	0.015	1.25	0.012963	0.864233	0.033321	4219	3	33
		1.50	0.012040	0.802693	0.030759	4428	2	18
		2.00	0.013918	0.927880	0.034465	3616	1	11
	0.020	1.25	0.019034	0.951725	0.044553	2488	2	31
		1.50	0.019927	0.996345	0.044832	2220	1	18
		2.00	0.017588	0.879400	0.041052	2704	1	11
bp	0.010	1.25	0.008567	0.856684	0.041685	4040	1	34
		1.50	0.007895	0.789483	0.039763	4440	1	19
		2.00	0.006707	0.670652	0.035610	5536	1	12
	0.015	1.25	0.012334	0.822273	0.050990	2700	1	33
		1.50	0.011491	0.766040	0.048765	2952	1	18
		2.00	0.009662	0.644145	0.044061	3616	1	11
	0.020	1.25	0.016283	0.814150	0.059394	1990	1	31
		1.50	0.014958	0.747885	0.056233	2220	1	18
		2.00	0.012737	0.636860	0.050952	2704	1	11
rk	0.010	1.25	0.008745	0.874528	0.032435	7892	4	34
		1.50	0.009620	0.962005	0.035308	6660	2	19
		2.00	0.006853	0.685274	0.027384	11072	2	12
	0.015	1.25	0.014561	0.970767	0.044361	4219	3	33
		1.50	0.013689	0.912627	0.043301	4428	2	18
		2.00	0.014963	0.997527	0.047917	3616	1	11
	0.020	1.25	0.016653	0.832630	0.051669	3110	3	31
		1.50	0.016694	0.834680	0.049933	3330	2	18
		2.00	0.019183	0.959160	0.055412	2704	1	11

Table 8. Progressive sampling results — soc-Epinions1

method	$\bar{\varepsilon}$	θ	ε	$\varepsilon/\bar{\varepsilon}$	prev. work ε	m	iter	max. iters
ab	0.010	1.25	0.009474	0.947417	0.023600	12332	6	34
		1.50	0.008431	0.843106	0.021566	14985	4	19
		2.00	0.006609	0.660918	0.017381	21632	3	11
	0.015	1.25	0.014307	0.953793	0.032939	6497	5	32
		1.50	0.014034	0.935607	0.032870	6642	3	18
		2.00	0.012931	0.862087	0.031322	7232	2	11
	0.020	1.25	0.019930	0.996515	0.043657	3888	4	31
		1.50	0.016890	0.844495	0.038935	4941	3	17
		2.00	0.016049	0.802455	0.036733	5344	2	10
bp	0.010	1.25	0.009999	0.999866	0.037539	5050	2	34
		1.50	0.007731	0.773050	0.032688	6660	2	19
		2.00	0.009098	0.909849	0.036275	5408	1	11
	0.015	1.25	0.013671	0.911420	0.046263	3325	2	32
		1.50	0.014318	0.954533	0.049099	2952	1	18
		2.00	0.012499	0.833273	0.044362	3616	1	11
	0.020	1.25	0.019747	0.987325	0.059800	1990	1	31
		1.50	0.018290	0.914500	0.056926	2196	1	17
		2.00	0.015038	0.751900	0.051607	2672	1	10
rk	0.010	1.25	0.009478	0.947762	0.021765	15415	7	34
		1.50	0.009691	0.969120	0.022076	14985	4	19
		2.00	0.007794	0.779378	0.018373	21632	3	11
	0.015	1.25	0.013991	0.932747	0.029985	8122	6	32
		1.50	0.012796	0.853073	0.027073	9963	4	18
		2.00	0.009877	0.658449	0.022469	14464	3	11
	0.020	1.25	0.017457	0.872855	0.034671	6075	6	31
		1.50	0.019446	0.972285	0.038444	4941	3	17
		2.00	0.019213	0.960655	0.036966	5344	2	10

Table 9. Progressive sampling results — loc-gowalla

method	$\bar{\varepsilon}$	θ	ε	$\varepsilon/\bar{\varepsilon}$	prev. work ε	m	iter	max. iters
		1.25	0.009087	0.908739	0.019617	114863.0	16.0	34.0
	0.010	1.50	0.008933	0.893260	0.019703	113796.0	9.0	19.0
		2.00	0.007008	0.700752	0.015891	173056.0	6.0	11.0
		1.25	0.013957	0.930487	0.030495	48419.0	14.0	32.0
ab	0.015	1.50	0.013525	0.901673	0.029869	50441.0	8.0	18.0
		2.00	0.012176	0.811740	0.027877	57856.0	5.0	11.0
		1.25	0.019917	0.995825	0.044654	23179.0	12.0	31.0
	0.020	1.50	0.016183	0.809135	0.035002	37524.0	8.0	17.0
		2.00	0.014604	0.730200	0.032514	42752.0	5.0	10.0
		1.25	0.009655	0.965488	0.011362	58809.0	13.0	34.0
	0.010	1.50	0.008327	0.832730	0.010004	75864.0	8.0	19.0
		2.00	0.007665	0.766471	0.009367	86528.0	5.0	11.0
		1.25	0.013636	0.909073	0.015653	30988.0	12.0	32.0
bp	0.015	1.50	0.012854	0.856933	0.015026	33627.0	7.0	18.0
		2.00	0.013651	0.910073	0.016200	28928.0	4.0	11.0
		1.25	0.018267	0.913350	0.020235	18543.0	11.0	31.0
	0.020	1.50	0.019853	0.992660	0.021337	16677.0	6.0	17.0
		2.00	0.016437	0.821860	0.018846	21376.0	4.0	10.0
		1.25	0.009830	0.982982	0.007973	114863.0	16.0	34.0
	0.010	1.50	0.009814	0.981435	0.008011	113796.0	9.0	19.0
		2.00	0.007852	0.785161	0.006496	173056.0	6.0	11.0
		1.25	0.013739	0.915927	0.010984	60524.0	15.0	32.0
rk	0.015	1.50	0.012344	0.822907	0.009824	75662.0	9.0	18.0
		2.00	0.013880	0.925333	0.011235	57856.0	5.0	11.0
		1.25	0.018442	0.922120	0.014200	36218.0	14.0	31.0
	0.020	1.50	0.017665	0.883255	0.013950	37524.0	8.0	17.0
		2.00	0.016198	0.809875	0.013069	42752.0	5.0	10.0

Table 10. Progressive sampling results — com-youtube

method	$\bar{\varepsilon}$	θ	ε	$\varepsilon/\bar{\varepsilon}$	prev. work ε	m	iter	max. iters
		1.25	0.009022	0.902180	0.022163	73512	14	34
	0.010	1.50	0.008594	0.859402	0.021761	75864	8	19
		2.00	0.007917	0.791696	0.020418	86528	5	11
		1.25	0.014309	0.953900	0.034785	30988	12	32
ab	0.015	1.50	0.013453	0.896880	0.033310	33627	7	18
		2.00	0.014783	0.985513	0.036112	28928	4	11
		1.25	0.018689	0.934440	0.044999	18543	11	31
	0.020	1.50	0.016518	0.825900	0.038838	25016	7	17
		2.00	0.017062	0.853090	0.042035	21376	4	10
		1.25	0.009515	0.951468	0.015000	37637	11	34
	0.010	1.50	0.008287	0.828735	0.012940	50576	7	19
		2.00	0.008758	0.875805	0.013991	43264	4	11
		1.25	0.014222	0.948147	0.020665	19832	10	32
bp	0.015	1.50	0.013143	0.876173	0.019436	22418	6	18
		2.00	0.011082	0.738820	0.017110	28928	4	11
		1.25	0.018312	0.915575	0.026714	11867	9	31
	0.020	1.50	0.018539	0.926970	0.027599	11118	5	17
		2.00	0.018664	0.933180	0.028149	10688	3	10
		1.25	0.009706	0.970636	0.008915	91890	15	34
	0.010	1.50	0.008430	0.843022	0.008011	113796	9	19
		2.00	0.009328	0.932777	0.009187	86528	5	11
		1.25	0.014879	0.991907	0.013730	38735	13	32
rk	0.015	1.50	0.012820	0.854673	0.012032	50441	8	18
		2.00	0.012079	0.805240	0.011235	57856	5	11
		1.25	0.019997	0.999860	0.017750	23179	12	31
	0.020	1.50	0.019511	0.975530	0.017086	25016	7	17
		2.00	0.013911	0.695550	0.013069	42752	5	10

Table 11. Progressive sampling results — as-skitter

method	$\bar{\varepsilon}$	θ	ε	$\varepsilon/\bar{\varepsilon}$	prev. work ε	m	iter	max. iters
		1.25	0.009987	0.998724	0.025307	15415.0	7.0	34.0
	0.010	1.50	0.008194	0.819447	0.021279	22478.0	5.0	19.0
		2.00	0.008265	0.826487	0.021189	22144.0	3.0	12.0
		1.25	0.014813	0.987507	0.034859	8242.0	6.0	33.0
ab	0.015	1.50	0.012863	0.857547	0.031232	9963.0	4.0	18.0
		2.00	0.010234	0.682280	0.025580	14464.0	3.0	11.0
		1.25	0.018048	0.902410	0.042448	6075.0	6.0	31.0
	0.020	1.50	0.015931	0.796550	0.037128	7493.0	4.0	18.0
		2.00	0.019077	0.953865	0.044658	5408.0	2.0	11.0
		1.25	0.009325	0.932549	0.029645	9865.0	5.0	34.0
	0.010	1.50	0.009262	0.926167	0.029458	9990.0	3.0	19.0
		2.00	0.008573	0.857300	0.027982	11072.0	2.0	12.0
		1.25	0.014085	0.938980	0.040544	5274.0	4.0	33.0
bp	0.015	1.50	0.011910	0.794033	0.036128	6642.0	3.0	18.0
		2.00	0.011385	0.759000	0.034623	7232.0	2.0	11.0
		1.25	0.017380	0.869010	0.047220	3888.0	4.0	31.0
	0.020	1.50	0.019055	0.952775	0.051024	3330.0	2.0	18.0
		2.00	0.013462	0.673095	0.040038	5408.0	2.0	11.0
		1.25	0.009067	0.906748	0.018566	24087.0	9.0	34.0
	0.010	1.50	0.009565	0.956499	0.019219	22478.0	5.0	19.0
		2.00	0.009683	0.968274	0.019363	22144.0	3.0	12.0
		1.25	0.013275	0.884973	0.025390	12879.0	8.0	33.0
rk	0.015	1.50	0.014972	0.998107	0.028868	9963.0	4.0	18.0
		2.00	0.012438	0.829220	0.023959	14464.0	3.0	11.0
		1.25	0.018483	0.924155	0.033065	7594.0	7.0	31.0
	0.020	1.50	0.018660	0.932985	0.033287	7493.0	4.0	18.0
		2.00	0.014584	0.729220	0.027706	10816.0	3.0	11.0