BAVARIAN: Betweenness Centrality Approximation with Variance-Aware Rademacher Averages

Cyrus Cousins
Dept. of Computer Science
Brown University
Providence, RI, USA
ccousins@cs.brown.edu

Chloe Wohlgemuth
Dept. of Computer Science
Amherst College
Amherst, MA, USA
cwohlgemuth22@amherst.edu

Matteo Riondato
Dept. of Computer Science
Amherst College
Amherst, MA, USA
mriondato@amherst.edu

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

ABSTRACT

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 MCERA 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 MCERA, and it allows us to assess the different trade-offs between sample size and accuracy guarantee 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**.

KEYWORDS

Concentration bounds, Dynamic graphs, Percolation centrality, Random sampling, Sample complexity, Statistical learning theory

ACM Reference Format:
Cyrus Cousins, Chloe Wohlgemuth, and Matteo Riondato. 2021. BAVARIAN: Betweenness Centrality Approximation with Variance-Aware Rademacher Averages. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467354

KDD '21, August 14–18, 2021, Virtual Event, Singapore

1 INTRODUCTION

A centrality measure [11] 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 the literature, from simple ones like degree to sophisticated ones based on shortest paths, such as closeness centrality [5] and betweenness centrality [2, 26], 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 [10]: 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 [47], to the study of the resilience of the electrical grid [42], to genomics [28].

Computing the exact BC of every vertex/edge in a graph $G = (V, E)$ is quite expensive. Brandes's algorithm (BA) [14], 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 [16, 33]. 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 characterizations of this trade-off allows for stronger quality guarantees. Previous works studying the trade-off [12, 16, 18, 27, 53, 55] (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. 5). This unifying approach can be used for both static and progressive sampling algorithms (Sect. 4.2), and extended to variants of BC, to percolation centrality, and to dynamic graphs (Sect. 4.3).
- Our analysis of BAVARIAN uses Monte-Carlo Empirical Rademacher Averages (MCERAs) (Sect. 3.3), a key concept from statistical learning theory [60]. 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 on the *vertex diameter* (Sect. 4.1), a characteristic and easy-to-bound quantity of the graph. Previous estimator-specific bounds depend on the size of the largest connected component [55], or on the number of vertices [16], which are much larger than the vertex diameter. This result generalizes one by Riondato and Kornaropoulos [53, Coroll. 1] for a specific estimator. At its core, there are *SP-enforcing families*, a novel concept of independent interest. This result is not just of theoretical interest: we use it to develop sample schedules for practical progressive sampling algorithms (Sect. 4.2).
- Our experimental evaluation (Sect. 5) shows the 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 a first fair comparison enabled by our unifying framework.

Due to space limitations, some proofs are deferred to App. A.2, while others are presented only in the extended online version at http://bit.ly/bvext, together with additional experimental results.

2 RELATED WORK

We now discuss the contributions on BC most related to ours. BC was originally introduced in sociology [2, 26], and many variants have been developed since then [13, 15, 23, 39, 46, 48, 49]. In this work, we focus on vertex BC, but the versatility of our approach allows us to tackle other variants (see Sect. 4.3).

The first efficient algorithm for BC [14], 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 [24, 52, 56], its cost is still excessive even for non-humongous graphs. Approximation algorithms [8, 9, 12, 16, 18, 27, 33, 45, 53, 55] and heuristics [3, 25, 43, 44] 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 [53, Sect. 2] and [11] 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 [16] and independently Jacob et al. [33] 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, and the analysis of the sample size vs. quality trade-off uses Hoeffding's bound [32] and the union bound, thus it is quite loose. Geisberger et al. [27] present a slight refinement of the estimator, but the sample size is unchanged. Chehreghani et al. [18] 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. [33] and Brandes and Pich [16].

Riondato and Kornaropoulos [53] 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 [61] 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 [8] 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 [12].

Riondato and Upfal [55] introduce another novel estimator (see Sect. 3.2) and an approximation algorithm whose analysis uses Rademacher averages [37], 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 (5)) and variance-aware tight deviation bounds (see Thm. 3.1), to achieve both aforementioned goals.

Recently, Fan et al. [25] 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 [8, 9, 29, 31, 35, 40, 51, 55, 62]. We discuss how to adapt our results to the fully-dynamic case in Sect. 4.3.

3 PRELIMINARIES

We now state definitions and theorems used throughout the work.

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 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 [19, 20, 36]. We define a path p from a vertex u to a vertex v as an ordered sequence of *distinct* vertices $(u, z_1, \ldots, z_{|p|-2}, v)$. 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 $|p| - 1$ if the edges do not have weights and is the sum of the weights of the edges $(u, z_1), (z_1, z_2), \ldots, (z_{|p|-2}, v)$ otherwise. Any path p from u to v that has the minimum 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, and let $\sigma_{uv} = |\Gamma_{uv}|$. 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 [2, 26]

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

There exist many variants of BC [13, 15, 23, 39, 46, 48, 49]. For ease of presentation, we focus on the one for vertices, but our results can be extended to many of these variants (see Sect. 4.3).

Computing the exact BCs for all vertices is expensive [14] (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.,

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

and it 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

$$\Pr\left(\exists w \in V \text{ s.t. } \left|\mathsf{b}(w) - \tilde{\mathsf{b}}_{\mathsf{A},\mathcal{S}}(w)\right| > \mathsf{eps}_{\mathsf{A},m}(\mathcal{S}, \delta)\right) < \delta \ . \tag{2}$$

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 [53] introduce a BC estimator that is tailored to their use of VC-dimension [61] 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

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

Riondato and Kornaropoulos [53] 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

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

The same estimator is used by Borassi and Natale [12] for a progressive sampling algorithm for BC estimation.

The ABRA estimator. The ABRA algorithm [55] uses an estimator defined over the domain

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

i.e., over the pairs of different 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

$$f_w(u, v) \doteq \frac{\sigma_{uv}(w)}{\sigma_{uv}} \in [0, 1] \ . \tag{3}$$

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. [33], and independently Brandes and Pich [16] introduce a BC estimator that is closely related to how Brandes [14]'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

$$f_w(v) \doteq \frac{1}{n-1} \sum_{z \neq v} \frac{\sigma_{vz}(w)}{\sigma_{vz}} \in [0,1] \ . \tag{4}$$

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 [14]. Geisberger et al. [27] 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.

For technical reasons, the constant zero function must belong to $\mathcal{F}_{\mathsf{A}}$. This requirement is automatically satisfied when the graph contains at least one vertex with only one neighbor. When such a vertex does not exist, we just add this function to $\mathcal{F}_{\mathsf{A}}$.

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 discuss their relationship. We tailor the definitions to our settings, and refer the reader to the book by Shalev-Shwartz and Ben-David [57, Ch. 26] for in-depth discussion.

Let $\mathcal{F}$ be a family of functions from a domain $\mathcal{D}$ to $[0,1]$. Let π be a distribution over $\mathcal{D}$, and m be a positive natural. 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

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

The SD is the key concept in the study of empirical processes [50]. Sample-dependent quantities, such as the popular Empirical Rademacher average (ERA) [37], can be used to derive probabilistic upper bounds to the SD. In this work, we use a Monte-Carlo estimation of the ERA, first introduced by Bartlett and Mendelson [4]. 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

$$\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda}) \doteq \frac{1}{k} \sum_{j=1}^{k} \left(\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{x_i \in \mathcal{S}} \boldsymbol{\lambda}_{i,j} f(x_i) \right) \ . \tag{5}$$

The expectation of the k-MCERA w.r.t. both $\mathcal{S}$ and $\mathbf{\Lambda}$ is known as the *Rademacher Average* $\mathsf{R}_m(\mathcal{F}, \pi)$ [37], and is a cornerstone of statistical learning theory. While Rademacher averages control the *expected SD*, the sharpest probabilistic bounds 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 [6] 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

$$\mathsf{v} \doteq \sup_{f \in \mathcal{F}} \mathbb{E}_\pi\left[f^2\right] \text{ and } \beta \doteq \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{x_i \in \mathcal{S}} (f(x_i))^2 \ . \tag{6}$$

Clearly both v and β depend on $\mathcal{F}$, and the families discussed in Sect. 3.2 have different wimpy variances.

The following result (proof in the extended online version at http://bit.ly/bvext) 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 Cousins and Riondato [21, Thm. 4], wherein the authors derive a similar bound involving *centralized* variances and Rademacher averages. We eschew such an approach here, as when all centralities are near-zero, centralization greatly complicates the analysis and yields larger constant-factors, with negligible benefit.

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

$$\begin{aligned}
\gamma &\doteq \beta + \frac{2\ln\frac{5}{\eta}}{3m} + \sqrt{\left(\frac{\ln\frac{5}{\eta}}{\sqrt{3}m}\right)^2 + \frac{2\beta\ln\frac{5}{\eta}}{m}}, \\
\rho &\doteq \hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda}) + \frac{2\ln\frac{5}{\eta}}{3km} + \sqrt{\frac{4\beta\ln\frac{5}{\eta}}{km}}, \\
r &\doteq \rho + \frac{\ln\frac{5}{\eta}}{3m} + \sqrt{\left(\frac{\ln\frac{5}{\eta}}{2\sqrt{3}m}\right)^2 + \frac{\rho\ln\frac{5}{\eta}}{m}} \ , \\
\varepsilon &\doteq 2r + \frac{\ln\frac{5}{\eta}}{3m} + \sqrt{\frac{2(\gamma+4r)\ln\frac{5}{\eta}}{m}} \ ,
\end{aligned} \tag{7}$$

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

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

The quantity γ is an upper bound to v, while ρ is an upper bound to the ERA, which is the expected value of the k-MCERA w.r.t. $\mathbf{\Lambda}$, and r is an upper bound to the expectation of the ERA w.r.t. $\mathcal{S}$. The apparent complexity of this result (in part due to the need to fit the expressions into the text column width) 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 (6). 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 [16] and Riondato and Kornaropoulos [53], assume the worst-case possible variance, while the bound used by Riondato and Upfal [55] improves to an exponential average over variances. These approaches are not sensitive to any correlation between the quantities to estimates, whereas the k-MCERA is. This advantage is one of the reasons why we choose them: 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.

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

Each f used for BC estimation has codomain $[0,1]$, so $\mathsf{v} = \mathbb{E}_\pi\left[f^2\right] \leq \mathbb{E}_\pi[f] \leq \max_{v\in V} \mathsf{b}(v)$. 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 (7), 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 (2) as

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

4 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, number of Monte Carlo trials k
Output: $(\tilde{B},\varepsilon)$ with the properties presented in Thm. 4.1

```
1  sums ← map from V to vectors of size k + 2
2  foreach v ∈ V do sums[v] ← (−∞, ..., −∞, 0, 0)   (k times)
3  for i from 1 to m do
4      s_i ← drawSample(A, G)
5      Z ← getFunctionValues(A, s_i)
6      λ_i ← drawRademacher(k)
7      foreach (v, f_v(s_i)) ∈ Z do
8          sums[v] ← sums[v] + f_v(s_i) · (λ_{i,1}, ..., λ_{i,k}, f_v(s_i), 1)
9  B̃ ← {(v, sums[v][k + 2]/m), v ∈ V}
10 ε ← getEpsilon(sums, m, δ)
11 return (B̃, ε)
```

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. 4.1 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 $v \in V$, where $\tilde{\mathsf{b}}_\mathsf{A}(v)$ is 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 4.1. *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[1] 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 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 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

$$\mathsf{sums}[v][z] = \begin{cases} \sum_{j=1}^i \boldsymbol{\lambda}_{j,z} f_v(s_j) & \text{for } z = 1,\ldots,k \\ \sum_{j=1}^i (f_v(s_j))^2 & \text{for } z = k+1 \\ \sum_{j=1}^i f_v(s_j) & \text{for } z = k+2; \end{cases} \tag{8}$$

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 (7): 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. 4.1.

PROOF OF THM. 4.1. Thanks to the maintained invariant (8), each estimate $\tilde{\mathsf{b}}_\mathsf{A}(v)$, $v \in V$, is a sample mean (over the sample $\mathcal{S} = \{s_1,\ldots,s_m\}$) of a specific function f_v, and an unbiased estimator for the exact BC $\mathsf{b}(w) = \mathbb{E}[f_v]$. Additionally, the invariant (8) ensures that all the quantities needed to compute ε as in (7) are available, and thus the thesis follows from Thm. 3.1. □

4.1 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 $\bar{\varepsilon} \in (0,1)$. In Sect. 4.1.1 discuss the choice of k as a function of m. These results are not just of theoretical interest: we use them in a *progressive-sampling* variant of BAVARIAN in Sect. 4.2.

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 (6) decrease, and as m and k increase. Consider a procedure zeroEpsilon(m,k,η) that, given m, k, and η, computes ε as in Thm. 3.1 with the assumption that $\hat{\mathsf{R}}^k_m(\mathcal{F},\mathcal{S},\boldsymbol{\lambda}) = 0$ (which is the minimum possible because of the requirement that the zero constant function belongs to $\mathcal{F}$), and $\beta = 0$, and let

$$\mathsf{m}_*(k,\eta,\bar{\varepsilon}) = \min\{m > 0 \,:\, \mathsf{zeroEpsilon}(m,k,\eta) \leq \bar{\varepsilon}\} . \tag{9}$$

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

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 a binary search on m using `zeroEpsilon`.

For $m \geq \mathsf{m}_*(k, \eta, \bar{\varepsilon})$, the algorithm *may* return $\varepsilon < \bar{\varepsilon}$. We leverage this property in Sect. 4.2, where we use (9) 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 [53, 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 [53, Coroll. 1] show that the *Vapnik-Chervonenkis (VC-) dimension* [61] 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 [53, 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. 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 4.2. *Let $d \geq \mathsf{vd}(G)$ or $d = 3$ if G is undirected and there is at most one SP between any pair of vertices in G. Let*

$$\mathsf{m}^*(\bar{\varepsilon}, \eta) \doteq \frac{1}{\bar{\varepsilon}^2}\left(4Cd + 4\sqrt{\frac{Cd \ln \frac{2}{\eta}}{2}} + \frac{\ln \frac{2}{\eta}}{2}\right), \tag{10}$$

where C is a universal constant [30, 38, 58]. 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. 4.1 would still hold.

The proof of this theorem (in App. A.2) relies on a novel family of functions that we call *SP-enforcing family* (Def. A.4), which we deem of independent interest, and on a result (Lemma A.2) to upper bound the ERA, i.e., the expectation w.r.t. $\boldsymbol{\lambda}$ of the k-MCERA, with the VC-dimension, used to adapt the result on the VC-dimension of BC-estimation with the rk estimator [53, Coroll. 1, Lemma 2] to Rademacher averages. Theorem 4.2 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. 4.2), which can now deterministically terminate earlier.

It is evident from the statement of Thm. 4.2 that an *upper bound* to the vertex-diameter of G is sufficient to compute the sufficient sample size. In unweighted graphs, it is possible to obtain a 2-approximation of the diameter (thus of the vertex-diameter) with one SSSP computation by summing the lengths of the two longest SPs. Bergamini and Meyerhenke [7, Sect. 3] show how to get a good upper bound to $\mathsf{vd}(G)$ on weighted graphs.

4.1.1 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 $\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\Lambda})$ is *asymptotically negligible* for *any choice* of k (i.e., k is absent from the asymptotic form). A value $k \geq 8$ 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 requires $\mathrm{O}(mk|V|)$ time for BC (often much less for the rk and ab estimators, see also Sect. 5). Assuming BFS with time complexity $\mathrm{O}(|E|)$, it makes sense to have $k \in \mathrm{O}(|E|/|V|)$ trials, and with Dijkstra's algorithm (time complexity $\mathrm{O}(|E| + |V| \log |V|)$), to have $k \in \mathrm{O}(|E|/|V| + \log |V|)$.

4.2 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. 4.1, 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 (9) and (10) 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 $m \geq m^*$. We now introduce a more flexible variant of Bavarian based on *progressive sampling*: rather than 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. 4.3, 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

$$m_{T-1} \geq \mathsf{m}^*(\bar{\varepsilon}, \delta/T) \tag{11}$$

for technical reasons explained in the proof of Thm. 4.3.

Bavarian-P first initializes the data structure sums exactly as in the static-sampling case (see Alg. 1 and its description), and

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. 4.3

1 sums $\leftarrow$ map from V to vectors of size $k+2$
2 **foreach** $v \in V$ **do** 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 $\tilde{B} \leftarrow \{(v, \text{sums}[v][k+2]/m_i), v \in V\}$
11 **return** $(\tilde{B}, \varepsilon)$

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 (8). 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). 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 (11). The approximation $\tilde{B}$ is computed (line 10) and output together with ε. The proof of this theorem is in App. A.2.

THEOREM 4.3. *With probability at least $1-\delta$ (over the runs of the algorithm), the pair $(\tilde{B}, \varepsilon)$ returned by BAVARIAN-P is such that*

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

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 (11), and, given what we discussed in Sect. 4.1, it makes no sense to use a larger sample size than the quantity in the r.h.s. of (11). 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 (9). It is not necessarily appropriate to use *exactly* this sample size as the first one, for the reason argued in Sect. 4.1, but this quantity acts as a lower bound. 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 [34]. 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 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 (9) with $\eta = \delta/T^*$) and the largest m_{T^*-1} (from (11), potentially smaller than $m_0 \theta^{T^*-1}$).

4.3 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. Due to space limitations, we only remark that the extensions to other BC variants would follow the approach discussed in [53, Sect. 6], and de Lima et al. [22] show how to estimate the percolation centrality via sampling by adapting the work of Riondato and Kornaropoulos [53] and Riondato and Upfal [55]. 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 [55, Sect. 5].

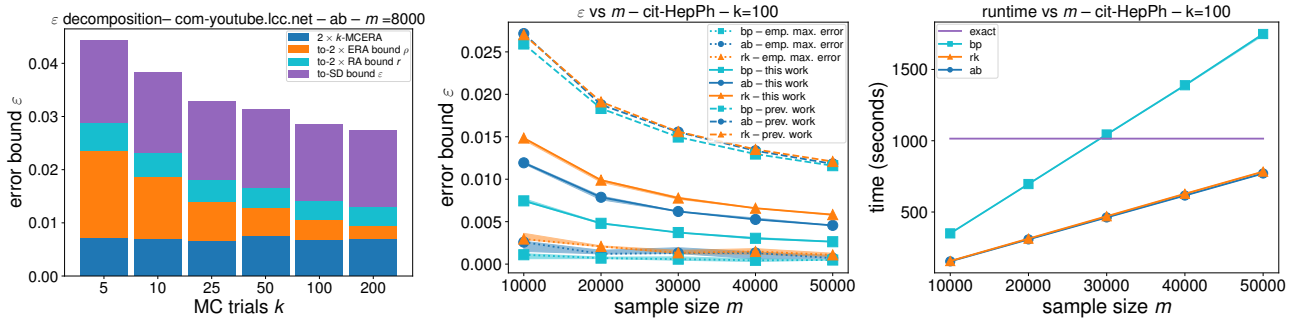
An interesting direction for future work is understanding the requirements for centrality measures to be amenable to approximation using the k-MCERA and/or similar tools. The extension is less straightforward than it may seem at first, due to the need for efficient sampling schemes from the appropriate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

5 EXPERIMENTAL EVALUATION

We now present the results of our experimental evaluation of BAVARIAN and of the comparison between the different estimators presented in Sect. 3.2. We do not discuss the performance of BAVARIAN-P separately, as it is tightly coupled with that of BAVARIAN. We report here only a subset of the results. Results for all datasets and parameters are in the online version at http://bit.ly/bvext, together with all code and datasets for reproducibility (see also App. A.1).

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 [16, 53, 55] and the experimental comparisons by Matta et al. [45] and AlGhamdi et al. [1].

Implementation, environment, and datasets. We implemented BAVARIAN in C++20 as an extension of NetworKit [59], and compared it with the implementation of Brandes's algorithm [14] included in the same suite. The code for reproducing all experiments and figures is at http://bit.ly/bvext. We run them 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



(a) Error bound ε decomposition **(b) Error bound ε vs sample size m** **(c) Runtime vs sample size m**

Figure 1: Experimental results. See description in the text.

Table 1: Dataset characteristics

Graph	Vertices	Edges
ca-AstroPh	17,903	197,031
cit-HepPh	34,546	421,578
email-Enron	36,682	183,831
p2p-Gnutella31	62,586	147,892
soc-Epinions1	75,879	508,837
com-dblp	317,080	1,049,866
com-youtube	1,134,890	2,987,624

(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 [41]. Their salient characteristics are reported in Table 1.

5.1 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.

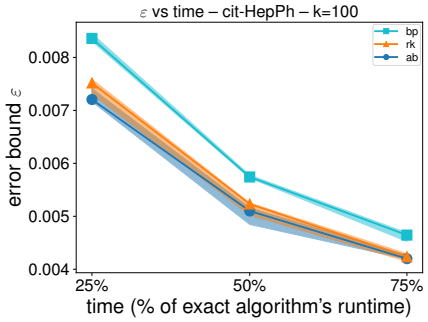


Figure 2: Error bound ε vs time.

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. 1a, we show the decomposition, and observe the behavior as function of k, on the `com-youtube` graph for the `ab` estimator, with $m = 8000$ (results for other values of the parameters, other datasets, and other estimators are qualitatively similar). 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., $2\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda})$), the orange ($2^{\text{nd}}$ from bottom) portion is the difference between 2ρ and $2\hat{\mathsf{R}}^k_m(\mathcal{F}, \mathcal{S}, \mathbf{\Lambda})$, 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. Figures 1b and 1c show the impact of the sample size m on the error bound ε and on the runtime, for `cit-HepPh` with $k = 100$ (results for other datasets/parameters are qualitatively similar). As expected, the error bound decreases approximately as $1/\sqrt{m}$ and the runtime increases linearly with m. The use of the k-MCERA and of Thm. 3.1 allows BAVARIAN to obtain much smaller error bounds than previous works for all estimators, and much closer to the empirical maximum error measured on the sample. The error bounds can still be tightened, but the leap forward is significant. From just Fig. 1b, 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: the estimators sample from different populations, and do different work per sample, so their performance at equal sample sizes is not comparable. Figure 1c clearly shows that BAVARIAN with `bp` takes much longer at each sample size than with other estimators (which take approximately the same time because they do similar work per sample), and becomes slower than the exact algorithm earlier. Thus, per sample, `bp` is more "efficient" than the other estimators, but it is *less efficient per unit of time*. Fig. 2 shows how, when run for the same amounts of time, `bp` gives a much worse (i.e., larger) ε than the other estimators. We conclude that the `ab` estimator gives the best "bang for the buck," i.e., the best ε per unit of time, and should likely be preferred over the others.

6 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. 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 as more efficient than the others in terms of "unit of guarantee" per unit of time. In future work, we plan to generalize the application of the k-MCERA to other centrality measures, and to design more efficient progressive sampling algorithms, leveraging martingale approaches.

ACKNOWLEDGMENTS

Part of this work is supported by the National Science Foundation grant 2006765 and by the DARPA/ARFL grant FA8750.

REFERENCES

[1] Z. AlGhamdi et al. 2017. A Benchmark for Betweenness Centrality Approximation Algorithms on Large Graphs. In SSDBM'17.
[2] J. M. Anthonisse. 1971. *The rush in a directed graph.* Technical Report BN 9/71. Stichting Mathematisch Centrum, Amsterdam, Netherlands.
[3] D. A. Bader et al. 2007. Approximating Betweenness Centrality. In *Algorithms and Models for the Web-Graph*, Springer, 124–137.
[4] P. L. Bartlett and S. Mendelson. 2002. Rademacher and Gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.* 3, 463–482.
[5] A. Bavelas. 1950. Communication patterns in task-oriented groups. *J. Acoust. Soc. Am.* 22, 6 (1950), 725–730.
[6] G. Bennett. 1962. Probability inequalities for the sum of independent random variables. *J. Amer. Statist. Assoc.* 57, 297 (1962), 33–45.
[7] E. Bergamini and H. Meyerhenke. 2015. Fully-dynamic Approximation of Betweenness Centrality. In ESA'15. 155–166.
[8] E. Bergamini and H. Meyerhenke. 2016. Approximating Betweenness Centrality in Fully-dynamic Networks. *Internet Math.* 12, 5 (2016), 281–314.
[9] E. Bergamini et al. 2015. Approximating Betweenness Centrality in Large Evolving Networks. In *ALENEX '15*. SIAM, 133–146.
[10] P. Boldi and S. Vigna. 2014. Axioms for centrality. *Internet Math.* 10, 3-4 (2014), 222–262.
[11] F. Bonchi et al. 2016. Centrality measures on big graphs: Exact, approximated, and distributed algorithms. In *WWW'15*. 1017–1020.
[12] M. Borassi and E. Natale. 2019. KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation. *J. Exp. Alg.* 24, 1 (2019).
[13] S. P. Borgatti and M. G. Everett. 2006. A Graph-theoretic perspective on centrality. *Soc. Netw.* 28, 4 (2006), 466–484.
[14] U. Brandes. 2001. A faster algorithm for betweenness centrality. *J. Math. Sociol.* 25, 2 (2001), 163–177.
[15] U. Brandes. 2008. On variants of shortest-path betweenness centrality and their generic computation. *Soc. Netw.* 30, 2 (2008), 136–145.
[16] U. Brandes and C. Pich. 2007. Centrality estimation in large networks. *Int. J. Bifur. Chaos* 17, 7 (2007), 2303–2318.
[17] S. Cabello et al. . 2013. Multiple-source shortest paths in embedded graphs. *SIAM J. Comput.* 42, 4 (2013), 1542–1571.
[18] M. H. Chehreghani et al. . 2018. Efficient Exact and Approximate Algorithms for Computing Betweenness Centrality in Directed Graphs. In *PAKDD'18*, 752–764.
[19] F. Chierichetti et al. 2016. On sampling nodes in a network. In *WWW'16*. 471–481.
[20] F. Chierichetti and S. Haddadan. 2018. On the Complexity of Sampling Vertices Uniformly from a Graph. In *ICALP'18*.
[21] C. Cousins and M. Riondato. 2020. Sharp uniform convergence bounds through empirical centralization. In *NeurIPS'20*.
[22] A. M. de Lima et al. 2020. Estimating the Percolation Centrality of Large Networks through Pseudo-dimension Theory. In *KDD'20*. ACM.
[23] S. Dolev et al. 2010. Routing betweenness centrality. *J. ACM* 57, 4, Article 25 (May 2010), 27 pages.
[24] D. Erdős et al. 2015. A Divide-and-Conquer Algorithm for Betweenness Centrality. In *SDM '15*. 433–441.
[25] C. Fan et al. 2019. Learning to Identify High Betweenness Centrality Nodes from Scratch. In *CIKM'19*. ACM.
[26] L. C. Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* 40 (1977), 35–41.
[27] R. Geisberger et al. 2008. Better Approximation of Betweenness Centrality. In *ALENEX '08*. SIAM, 90–100.
[28] J. Ghurye and .M Pop. 2016. Better Identification of Repeats in Metagenomic Scaffolding. In *WABI 2016*. Springer, 174–184.
[29] O. Green et al. 2012. A Fast Algorithm for Streaming Betweenness Centrality. In *PASSAT '12*. IEEE, 11–20.
[30] D. Haussler. 1995. Sphere packing numbers for subsets of the Boolean n-cube with bounded Vapnik-Chervonenkis dimension. *J. Comb. Th., Ser. A* 69, 2 (1995).
[31] T. Hayashi et al. 2015. Fully Dynamic Betweenness Centrality Maintenance on Massive Networks. *VLDB'16* (2015).
[32] W. Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. Am. Stat. Assoc.* 58, 301 (1963), 13–30.
[33] R. Jacob et al. 2005. Algorithms for Centrality Indices. In *Network Analysis*. Springer, 62–82.
[34] G. H. John and P. Langley. 1996. Static Versus Dynamic Sampling for Data Mining. In *KDD '96*. The AAAI Press, Menlo Park, CA, USA, 367–370.
[35] M. Kas et al. 2013. Incremental Algorithm for Updating Betweenness Centrality in Dynamically Growing Networks. In *ASONAM '13*. IEEE/ACM, 33–40.
[36] L. Katzir et al. 2014. Estimating sizes of social networks via biased sampling. *Internet Math.* 10, 3-4 (2014).
[37] V. Koltchinskii. 2001. Rademacher penalties and structural risk minimization. *IEEE Trans. Inf. Th.* 47, 5 (July 2001), 1902–1914.
[38] A. Kontorovich. 2016. Agnostic PAC lower bound. https://www.cs.bgu.ac.il/~asml162/wiki.files/agnostic-pac-lb.pdf
[39] N. Kourtellis et al. 2012. Identifying high betweenness centrality nodes in large social networks. *Soc. Netw. Anal. Mining* 3, 4 (2012), 899–914.
[40] N. Kourtellis et al. 2015. Scalable Online Betweenness Centrality in Evolving Graphs. *IEEE Trans. Knowl. Data Eng.* 27, 9 (2015), 2494–2506.
[41] J. Leskovec and A. Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.
[42] Y. Li et al. 2019. Electric Power Grid Invulnerability Under Intentional Edge-Based Attacks. In *DependSys'19*, 454–461.
[43] Y.-S. Lim et al. 2011. Online estimating the k central nodes of a network. In *IEEE Netw. Sci. Work. (NSW'11)*. 118–122.
[44] A. S. Maiya and T. Y. Berger-Wolf. 2010. Online Sampling of High Centrality Individuals in Social Networks. In *PAKDD'10*, 91–98.
[45] J. Matta et al. 2019. Comparing the speed and accuracy of approaches to betweenness centrality approximation. *Comp. Soc. Netw.* 6, 1 (2019), 2.
[46] A. McLaughlin and D. A. Bader. 2014. Scalable and High Performance Betweenness Centrality on the GPU. *SC'14* (Nov 2014).
[47] M. E. J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E* 69 (Feb. 2004). Issue 2.
[48] T. Opsahl et al. 2010. Node centrality in weighted networks: Generalizing degree and shortest paths. *Soc. Netw.* 32, 3 (2010), 245–251.
[49] J. Pfeffer and K. M. Carley. 2012. k-Centralities: local approximations of global measures based on shortest paths. In *WWW '12*. ACM, 1043–1050.
[50] D. Pollard. 1984. *Convergence of stochastic processes.* Springer-Verlag.
[51] M. Pontecorvi and V. Ramachandran. 2015. Fully Dynamic Betweenness Centrality. In *ISAAC '15*. 331–342.
[52] D. Prountzos and K. Pingali. 2013. Betweenness centrality: algorithms and implementations. In *PPoPP '13*. ACM 35–46.
[53] M. Riondato and E. M. Kornaropoulos. 2016. Fast approximation of betweenness centrality through sampling. *Data Min. Knowl. Disc.* 30, 2 (2016), 438–475.
[54] M. Riondato and E. Upfal. 2015. Mining Frequent Itemsets through Progressive Sampling with Rademacher Averages. In *KDD '15*. ACM, 1005–1014.
[55] M. Riondato and E. Upfal. 2018. ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. *ACM Trans. Knowl. Disc. Data* 12, 5 (2018), 61.
[56] A. E. Sarıyüce et al. 2017. Graph Manipulations for Fast Centrality Computation. *ACM Trans. Knowl. Disc. Data* 11, 3 (2017), 1–25.
[57] S. Shalev-Shwartz and S. Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press.
[58] N. Srebro and K. Sridharan. 2010. Note on refined Dudley integral covering number bound. (2010). http://www.cs.cornell.edu/~sridharan/dudley.pdf.
[59] C. L. Staudt et al., 2016. NetworKit: An Interactive Tool Suite for High-Performance Network Analysis. *Netw. Sci.* 4, 4 (2016).
[60] V. N. Vapnik. 1998. *Statistical learning theory.* Wiley.
[61] V. N. Vapnik and A. J. Chervonenkis. 1971. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Th. Prob. Appl.* 16, 2 (1971), 264–280.
[62] Y. Yoshida. 2014. Almost Linear-time Algorithms for Adaptive Betweenness Centrality Using Hypergraph Sketches. In *KDD '14*. ACM, 1416–1425.

A ADDITIONAL MATERIALS FOR BAVARIAN

A.1 Reproducibility and Additional Results

All the code and datasets can be found in the archive containing also the extended version of this article at http://bit.ly/bvext. It includes a `README.txt` file with instructions, and a script to run all our experiments and to generate all the figures. Additional experimental results can be found in the extended version of this article in the same archive.

A.2 Missing Proofs

The proof of Thm. 3.1 is in App. A.2 of the extended online version available at http://bit.ly/bvext.

We now prove Thm. 4.2.

The *Rademacher average* of a family $\mathcal{F}$ of functions w.r.t. the probability distribution π over the domain $\mathcal{D}$ of the functions in $\mathcal{F}$ is the expectation, w.r.t. both the sample $\mathcal{S}$ (which is drawn according to π^m) and the bag of m k-dimensional Rademacher vectors $\mathbf{\Lambda}$ of the k-MCERA:

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

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

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

$$\mathsf{SD}(\mathcal{F}, \mathcal{S}) \leq 2\mathsf{R}_m(\mathcal{F}, \pi) + \sqrt{\frac{\ln \frac{2}{\eta}}{2m}} \ . \tag{12}$$

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 $\mathsf{R}_m(\mathcal{F}_\mathsf{A}, \pi_\mathsf{A})$.

Lemma A.2. *Let $d \geq \mathsf{vd}(G)$ or $d = 3$ if G is undirected and there is at most one SP between any pair of vertices in G. For some universal constant C [30, 38, 58], 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 [61] of the BC estimation task on G, as shown by Riondato and Kornaropoulos [53, Thm. 2], and from a standard result from statistical learning theory connecting the Rademacher average to the VC-dimension [30, 58]. □

The proof of Thm. 4.2 for the `rk` estimator then follows from using the upper bound in Lemma A.2 as an upper bound to the Rademacher average in Thm. A.1, requiring the l.h.s. of (12) to be at most $\bar{\varepsilon}$, and solving for m to obtain the expression for $\mathsf{m}^*(\bar{\varepsilon}, \eta)$.

We show next that Thm. 4.2 holds also for the estimators `ab` and `bp`. To this end, we need to introduce the concepts of *unique-SP-enforcers* and *unique-SP-enforcing families*.

Definition A.3. A *(unique-SP-)enforcer* t is a function from $V \times V$ to the set of all SPs in the graph G, i.e.,

$$\mathsf{t} : V \times V \to \bigcup_{(u,v)\in V\times V} \Gamma_{uv},$$

mapping each pair $(u, v) \in V \times V$ to a SP $p \in \Gamma_{uv}$ if such a path exists (i.e., if $\Gamma_{uv} \neq \emptyset$), and to $\emptyset$ otherwise.

There are

$$q \doteq \prod_{\substack{(u,v)\in V\times V \\ \text{s.t. } \sigma_{uv}\geq 1}} \sigma_{uv} \tag{13}$$

different enforcers. Let $\mathcal{Q}$ denote the set of all enforcers. Clearly $q = |\mathcal{Q}|$.

Definition A.4. Given an enforcer $\mathsf{t} \in \mathcal{Q}$, the *(unique-SP-)enforcing family $\mathcal{T}_\mathsf{t}$ associated to* t is a family of functions from to $\{0, 1\}$, such that $\mathcal{T}$ contains one function $f_{\mathsf{t},w}$ for each $w \in V$ defined as

$$f_{\mathsf{t},w}(u, v) \doteq \begin{cases} 1 & \text{if } w \in \mathsf{int}(\mathsf{t}(u, v)) \\ 0 & \text{otherwise} \end{cases} \ .$$

The following result holds for every enforcing family $\mathcal{T}_\mathsf{t}$, $\mathsf{t} \in \mathcal{Q}$.

Lemma A.5. *For any $\mathsf{t} \in \mathcal{Q}$, the VC-dimension of $\mathcal{T}_\mathsf{t}$ is at most $\mathsf{vd}(G)$, unless $q = 1$, in which case the VC-dimension of the only $\mathcal{T}_\mathsf{t}$ is exactly 3.*

Proof. The proof for the general case is exactly the same as the one for [53, Thm. 2], while the one for the case $q = 1$ is the one for [53, Lemma 2]. □

A family $\mathcal{T}_\mathsf{t}$ cannot generally be seen as the family resulting from imposing a set of weights on the edges of G enforcing unique shortest paths between every pair of connected vertices [17]. The reason is that t may be such that there exists $(u, v) \in V \times V$ such that there are two vertices (w, z) on the enforced SP $\mathsf{t}(u, v)$ for which $\mathsf{t}(w, z)$ is not a subset of $\mathsf{t}(u, v)$. In other words, t does not enforce "consistent" SPs, which instead must happen when imposing a set of weights as above. For this reason, the result from [53, Lemma 2] can only be used when there is a single enforcer in $\mathcal{Q}$, as it relies on such consistency of the SPs.

The following result connects the set $\{\mathcal{T}_\mathsf{t}, \mathsf{t} \in \mathcal{Q}\}$ of enforcing families to the family $\mathcal{F}_\mathsf{ab}$ used by the `ab` estimator (see (3)).

Lemma A.6. *For any $w \in V$, let $f_w \in \mathcal{F}_\mathsf{ab}$ be the function associated to w in this family, as defined in* (3)*. For any $(u, v) \in \mathcal{D}_\mathsf{ab}$, it holds*

$$f_w(u, v) = \frac{1}{q} \sum_{\mathsf{t}\in\mathcal{Q}} f_{\mathsf{t},w}(u, v) \ .$$

Proof. The thesis is immediate when $\sigma_{uv}(w) = 0$, as $f_{\mathsf{t},w}(u, v) = 0$ for each $\mathsf{t} \in \mathcal{Q}$. Let now $\Gamma_{uv}(w) \subseteq \Gamma_{uv}$ be the set of all SPs from u to v that w is internal to. From the definition of enforcer, it holds that for each SP $p \in \Gamma_{uv}(w)$, there are exactly

$$z \doteq \prod_{\substack{(x,y)\in V\times V\setminus\{(u,v)\} \\ \text{s.t. } \sigma_{xy}\geq 1}} \sigma_{xy} \tag{14}$$

enforcers in $\mathcal{Q}$ mapping (u, v) to p. Thus there are exactly $\sigma_{uv}(w) \cdot z$ enforcers that map (u, v) to a SP to which w is internal. It holds $f_{\mathsf{t},w}(u, v) = 1$ iff t is one of these $\sigma_{uv}(w) \cdot z$ enforcers. It follows that

$$\sum_{\mathsf{t}\in\mathcal{Q}} f_{\mathsf{t},w}(u, v) = \sigma_{uv}(w) \cdot z \ .$$

It also holds, from (14) and (13), that

$$\frac{1}{\sigma_{uv}} = \frac{z}{q} \ .$$

The thesis follows from these two identities as

$$\frac{\sigma_{uv}(w)}{\sigma_{uv}} = \frac{\sigma_{uv}(w)z}{q} = \frac{1}{q}\sum_{\mathsf{t}\in Q} f_{\mathsf{t},w}(u,v) \ .$$

□

The following lemma corresponds to Lemma A.2 for the ab estimator. By using it with Thm. A.1, we obtain Thm. 4.2 for ab.

Lemma A.7. *Let $d \geq \mathsf{vd}(G)$ or $d = 3$ if G is undirected and there is at most one SP between any pair of vertices in G. For some universal constant C [30, 38, 58], it holds that*

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

Proof. Let $\mathcal{S} = \{(u_1, v_1), \ldots, (u_m, v_m)\}$ be a sample from $\mathcal{D}_{\mathsf{ab}}$ and $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_m\}$ be a bag of m k-dimensional Rademacher vectors. From Lemma A.6 it holds

$$\hat{\mathsf{R}}^m_k(\mathcal{F}_{\mathsf{ab}}, \mathcal{S}, \mathbf{\Lambda}) = \frac{1}{mk}\sum_{j=1}^{k}\sup_{w\in V}\sum_{i=1}^{m}\left(\boldsymbol{\lambda}_{i,j}\frac{1}{q}\sum_{\mathsf{t}\in Q} f_{\mathsf{t},w}(u_i, v_i)\right) \ .$$

From here, thanks to the fact that the constant zero function belongs to $\mathcal{F}$, we can use the subadditivity of the supremum operator to get

$$\begin{aligned}\hat{\mathsf{R}}^m_k(\mathcal{F}_{\mathsf{ab}}, \mathcal{S}, \mathbf{\Lambda}) &\leq \frac{1}{q}\sum_{\mathsf{t}\in Q}\frac{1}{mk}\sum_{j=1}^{k}\sup_{w\in V}\sum_{i=1}^{m}\left(\boldsymbol{\lambda}_{i,j} f_{\mathsf{t},w}(u_i, v_i)\right) \\ &= \frac{1}{q}\sum_{\mathsf{t}\in Q}\hat{\mathsf{R}}^m_k(\mathcal{F}_{\mathsf{t}}, \mathcal{S}, \mathbf{\Lambda}) \ .\end{aligned}$$

The above inequality is true for every choice of $\mathcal{S}$ and $\mathbf{\Lambda}$, so it must also hold that

$$\mathsf{R}_m(\mathsf{ab}) \leq \frac{1}{q}\sum_{\mathsf{t}\in Q}\mathsf{R}_m(\mathcal{F}_{\mathsf{t}}, \pi_{\mathsf{ab}}) \ .$$

Lemma A.5 allows us to say that each summand on the r.h.s. is at most $\sqrt{(Cd)/m}$, and the proof is complete. □

We now complete the discussion of Thm. 4.2 by showing results equivalent to Lemmas A.6 and A.7 for the bp estimator.

Lemma A.8. *For any $w \in V$, let $f_w \in \mathcal{F}_{\mathsf{bp}}$ be the function associated to w in this family, as defined in* (4)*. For any $u \in \mathcal{D}_{\mathsf{bp}}$, it holds*

$$f_w(u) = \frac{1}{q}\frac{1}{|V|-1}\sum_{v\neq w}\sum_{\mathsf{t}\in Q} f_{\mathsf{t},w}(u,v) \ .$$

Proof. The proof follows from the definition of $f_w(u)$ and of $f_w(u,v)$ in (3) and from Lemma A.6 □

Lemma A.9. *Let $d \geq \mathsf{vd}(G)$ or $d = 3$ if G is undirected and there is at most one SP between any pair of vertices in G. For some universal constant C [30, 38, 58], it holds that*

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

Proof. The proof follows the same step as the one for Lemma A.7, but using Lemma A.8 in place of Lemma A.6. □

We now show the proof for Thm. 4.3.

Theorem 4.3. *With probability at least $1-\delta$ (over the runs of the algorithm), 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 $\pi\mathsf{A}$, 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 $\pi\mathsf{A}$ 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

$$\max_{v\in V}\left|\tilde{\mathsf{b}}_{\mathsf{A}}(v) - \mathsf{b}(v)\right| < \varepsilon \ . \tag{15}$$

The last sample size m_{T-1} must satisfy the requirement from (11) and when using this sample size, V outputs $\varepsilon = \bar{\varepsilon}$. Combining this fact with (15), 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. □