Fast Approximation of Betweenness Centrality through Sampling [*][†]

Matteo Riondato[‡]and Evgenios M. Kornaropoulos
Department of Computer Science – Brown University
{matteo,evgenios}@cs.brown.edu

July 7, 2014

Abstract

Betweenness centrality is a fundamental measure in social network analysis, expressing the importance or influence of individual vertices in a network in terms of the fraction of shortest paths that pass through them. Exact computation in large networks is prohibitively expensive and fast approximation algorithms are required in these cases. We present two efficient randomized algorithms for betweenness estimation. The algorithms are based on random sampling of shortest paths and offer probabilistic guarantees on the quality of the approximation. The first algorithm estimates the betweenness of all vertices: all approximated values are within an additive factor ε from the real values, with probability at least $1-\delta$. The second algorithm focuses on the top-K vertices with highest betweenness and approximate their betweenness within a multiplicative factor ε, with probability at least $1-\delta$. This is the first algorithm that can compute such approximation for the top-K vertices. We use results from the VC-dimension theory to develop bounds to the sample size needed to achieve the desired approximations. By proving upper and lower bounds to the VC-dimension of a range set associated with the problem at hand, we obtain a sample size that is independent from the number of vertices in the network and only depends on a characteristic quantity that we call the vertex-diameter, that is the maximum number of vertices in a shortest path. In some cases, the sample size is completely independent from any property of the graph. The extensive experimental evaluation that we performed using real and artificial networks shows that our algorithms are significantly faster and much more scalable as the number of vertices in the network grows than previously presented algorithms with similar approximation guarantees.

1 Introduction

Centrality indices are fundamental metrics for network analysis. They express the relative importance of a vertex in the network. Some of them, e.g., degree centrality, reflect local properties of the underlying graph, while others, like betweenness centrality, give information about the global network structure, as they are based on shortest path computation and counting [30]. In this work we are interested in *betweenness centrality* [3, 15], that is, for every vertex in the graph, the fraction of shortest paths that goes through that vertex (see Section 3 for formal definitions) , and on some variants of it [8, 10, 32]. Betweenness centrality has been used to analyze social and protein interaction networks, to evaluate traffic in communication networks, and to identify important intersections in road networks [16, 30]. There exist polynomial-time algorithms to compute the exact betweenness centrality [9], but they are not practical for the analysis of the very large networks that are of interest these days. Graphs representing online social networks, communication networks, and the web graph have millions of nodes and billions of edges, making a polynomial-time algorithm too expensive in practice. Given that data mining is exploratory in nature, approximate results are usually

[*] This work is supported in part by the National Science Foundation under award IIS-1247581.

[†] This is the extended version of the article accepted for publication at WSDM'14 [35]. The most up-to-date version of this document can be downloaded from http://cs.brown.edu/~matteo/papers/RiondatoKornarop-BetweennessSampling.pdf.

[‡] Contact author

sufficient, especially if the approximation error is guaranteed to be within user-specified limits. In practice, the user is interested in the relative ranking of the vertices according to their betweenness, rather than the actual value of the betweenness, so a very good estimation of the value of each vertex is sufficiently informative for most purposes. It is therefore natural to develop algorithms that trade off accuracy for speed and efficiently compute high-quality approximations of the betweenness of the vertices. Nevertheless, in order for these algorithms to be practical it is extremely important that they scale well and have a low runtime dependency on the size of the network (number of vertices and/or edges).

Our contributions We present two randomized algorithms to approximate the betweenness centrality (and some of its variants) of the vertices of a graph. The first algorithm guarantees that the estimated betweenness values for all vertices are within an *additive* factor ε from the real values, with probability at least $1-\delta$. The second algorithm focuses on the top-K vertices with highest betweenness and returns a *superset* of the top-K, while ensuring that the estimated betweenness for all returned vertices is within a *multiplicative* factor ε from the real value, with probability at least $1-\delta$. This is the first algorithm to reach such a high-quality approximation for the set of top-K vertices. The algorithms are based on random sampling of shortest paths. The analysis to derive the sufficient sample size is novel and uses notions and results from VC-dimension theory. We define a range set associated with the problem at hand and prove strict bounds to its VC-dimension. The resulting sample size *does not depend on the size of the graph*, but only on the maximum number of vertices in a shortest path, a *characteristic quantity* of the graph that we call the *vertex-diameter*. For some networks, we show that the VC-dimension is actually at most a constant and so the sample size depends *only on the approximation parameters* and not on any property of the graph, a somewhat surprising fact that points out interesting insights. Thanks to the lower runtime dependency on the size of the network, our algorithms are *much faster and more scalable* than previous contributions [11, 16, 19], while offering the same approximation guarantees. Moreover, the amount of work performed by our algorithms per sample is also less than that of others algorithms. We extensively evaluated our methods on real graphs and compared their performances to the exact algorithm for betweenness centrality [9] and to other sampling-based approximation algorithms [11, 16, 19], showing that our methods achieve a huge speedup (3 to 4 times faster) and scale much better as the number of vertices in the network grows.

We present related work in Sect. 2. Section 3 introduces all the basic definitions and results that we will use throughout the paper. A range set for the problem at hand and the bounds to its VC-dimension are presented in Sect. 4. Based on these results we develop and analyze algorithms for betweenness estimation that we present in Sect. 5. Extensions of our methods to various variants of the problem are presented in Sect. 6. Section 7 reports the methodology and the results of our extensive experimental evaluation.

2 Related work

Over the years, a number of centrality measures have been defined [30]. In this work we focus on betweenness centrality and some of its variants.

Betweenness centrality was introduced in the sociology literature [3, 15]. Brandes [10] presents a number of minor variants. A particularly interesting one, k-bounded-distance betweenness, limits the length of the shortest paths considered when computing the centrality [8, 10, 33]. This is not to be confused with k-path betweenness centrality defined by Kourtellis et al. [21], which considers simple random walks that are not necessarily shortest paths. Dolev et al. [13] present a generalization of betweenness centrality which takes into account routing policies in the network. Opsahl et al. [32] define a new distance function between pair of vertices in order to penalize paths with a high number of hops in weighted network. This function induces a generalized and parametrized definition of betweenness.

The need of fast algorithms to compute the betweenness of vertices in a graph arose as large online social networks started to appear. Brandes [9] presents the first efficient algorithm for the task, running in time $O(nm)$ on unweighted graphs and $O(nm + n^2 \log n)$ on weighted ones. The algorithm computes, for each vertex v, the shortest path to every other vertex and then traverses these paths backwards to efficiently compute the contribution of the shortest paths from v to the betweenness of other vertices. For very large networks, the cost of this algorithm would still be prohibitive in practice, so many approximation algorithms were developed [5, 11, 16, 19, 24, 26]. The use of random sampling was one of the more natural approaches to speed up the computation of betweenness. Inspired by the work of Eppstein and Wang [14], Jacob et al. [19] and independently Brandes and Pich [11] present an algorithm that mimics the exact one, with the difference that, instead of computing the contribution of all vertices to the betweenness of the others, it only considers the contributions of some vertices sampled uniformly at random. To guarantee that all estimates are within ε from their real value with probability at least $1-\delta$, the algorithm from [11, 19] needs $O(\log(n/\delta)/\varepsilon^2)$ samples. The analysis for the derivation of the sample size uses Hoeffding bounds [18] and the union bound [28]. Geisberger et al. [16] noticed that this can lead to an overestimation of the betweenness of vertices that are close to the sampled ones and introduced different unbiased estimators that are experimentally shown to have smaller variance and do not suffer from this overshooting. Our algorithm is different from these because we sample, each time, a single random shortest path. This leads to a much smaller sample size and less work done for each sample, resulting in a much faster way to compute approximations of the betweenness with the same probabilistic guarantees. Although existing algorithms [11, 16, 19] can be extended to return a superset of the top-K vertices with highest betweenness, they only offer an *additive* approximation guarantee, while our algorithm for the top-K vertices offers a *multiplicative* factor guarantee, which is much stricter. We delve more in the comparisons with these algorithms in Sect. 5.3 and 7.

A number of works explored the use of adaptive sampling, in contrast with the previous algorithms (and ours) which use a fixed sample size. Bader et al. [5] present an adaptive sampling algorithm which computes good estimations for the betweenness of high-centrality vertices, by keeping track of the partial contribution of each sampled vertex, obtained by performing a single-source shortest paths computation to all other vertices. Maiya and Berger-Wolf [26] use concepts from expander graphs to select a connected sample of vertices. They estimate the betweenness from the sample, which includes the vertices with high centrality. They build the connected sample by adding the vertex which maximizes the number of connections with vertices not already in the sample. Modified versions of this algorithm and an extensive experimental evaluation appeared in [24]. The algorithm does not offer any guarantee on the quality of the approximations. Compared to these adaptive sampling approaches, our methods ensure that the betweenness of all (or top-K) vertices is well approximated, while using a fixed, predetermined amount of samples. Sarıyüce et al. [37] present an algorithm that pre-processes the network in multiple ways by removing degree-1 vertices and identical vertices, and splits it in separate components where the computation of betweenness can be performed independently and then aggregated. They do not present an analysis of the complexity of the algorithm.

In the analysis of our algorithm we use results from VC-dimension theory [38], a key component of statistical learning theory. We compute an upper bound to the VC-dimension of a range set defined on shortest paths. Kranakis et al. [22] present a number of results on the VC-dimension of various range sets for graphs (stars, connected sets of vertices, sets of edges), but do not deal with shortest paths. Abraham et al. [1] use VC-dimension to speed up shortest path computation but their range set is different from the one we use: their ground set is the set of vertices while ours is defined on shortest paths.

The present work extends substantially the preliminary version [35]. The proof of all the theorems and lemmas are presented here for the first time. Examples of betweenness centrality and of the concept of VC-dimension have been added to Sect. 3. A major new contribution is a tighter analysis of the algorithm from [11], presented in Sect. 5.3. We also discuss many other variants of betweenness, including edge betweenness in Sect. 6, where we also present a new tighter analysis of the algorithm for k-path betweenness centrality by [21].

3 Preliminaries

In this section we introduce the definitions and lemmas that we will use to develop and analyze our results throughout the paper.

3.1 Graphs and betweenness centrality

Let $G = (V, E)$ be a graph, where $E \subseteq V \times V$, with $n = |V|$ vertices and $m = |E|$ edges. The graph G can be directed or undirected. We assume that there are no self-loops from one vertex to itself and no multiple edges between a pair of vertices. Each edge $e \in E$ has a non-negative weight $\mathsf{w}(e)$. Given a pair of distinct vertices $(u, v) \in V \times V$, $u \neq v$, a *path* $p_{uv} \subseteq V$ *from* u *to* v is an ordered sequence of vertices $p_{uv} = (w_1, \ldots, w_{|p_{uv}|})$ such that $w_1 = u$, $w_{|p_{uv}|} = v$ and for each $1 \leq i < |p_{uv}|$, $(w_i, w_{i+1}) \in E$. The vertices u and v are called the *end points* of p_{uv} and the vertices in $\mathsf{Int}(p_{uv}) = p_{uv} \setminus \{u, v\}$ are the *internal vertices of* p_{uv}. The *weight* $\mathsf{w}(p_{uv})$ of a path $p_{uv} = (u = w_1, w_2, \cdots, w_{p_{|uv|}} = v)$ from u to v is the sum of the weights of the edges composing the path: $\mathsf{w}(p_{uv}) = \sum_{i=1}^{|p_{uv}|-1} \mathsf{w}((w_i, w_{i+1}))$. We denote with $|p_{uv}|$ the number of vertices composing the path and call this the *size of the path* p_{uv}. Note that if the weights are not all unitary, it is not necessarily true that $\mathsf{w}(p_{uv}) = |p_{uv}| - 1$. A special and degenerate path is the *empty path* $p_\emptyset = \emptyset$, which by definition has weight $\mathsf{w}(p_\emptyset) = \infty$, no end points, and $\mathsf{Int}(p_\emptyset) = \emptyset$.

Given two distinct vertices $(u, v) \in V \times V$, the *shortest path distance* d_{uv} between u and v is the weight of a path with minimum weight between u and v among all paths between u and v. If there is no path between u and v, $d_{uv} = \infty$. We call a path between u and v with weight d_{uv} a *shortest path between* u *and* v. There can be multiple shortest paths between u and v and we denote the set of these paths as $\mathcal{S}_{uv}$ and the number of these paths as $\sigma_{uv} = |\mathcal{S}_{uv}|$. If there is no path between u and v, then $\mathcal{S}_{uv} = \{p_\emptyset\}$[1]. We denote with $\mathbb{S}_G$ the union of all the $\mathcal{S}_{uv}$'s, for all pairs $(u, v) \in V \times V$ of distinct nodes $u \neq v$:

$$\mathbb{S}_G = \bigcup_{\substack{(u,v) \in V \times V \\ u \neq v}} \mathcal{S}_{uv} \enspace .$$

We now define a characteristic quantity of a graph that we will use throughout the paper.

Definition 1. Given a graph $G = (V, E)$, the *vertex-diameter* $\mathsf{VD}(G)$ *of* G is the size of the shortest path in G with maximum size:

$$\mathsf{VD}(G) = \max\{|p| \ : \ p \in \mathbb{S}_G\} \enspace .$$

The vertex-diameter is the maximum number of vertices among all shortest paths in G. If all the edge weights are unitary, then $\mathsf{VD}(G)$ is equal to $\mathsf{diam}(G) + 1$, where $\mathsf{diam}(G)$ is the number of edges composing the longest shortest path in G.

Given a vertex v, let $\mathcal{T}_v \subseteq \mathbb{S}_G$ be the set of all shortest paths that v is *internal* to:

$$\mathcal{T}_v = \{p \in \mathbb{S}_G \ : \ v \in \mathsf{Int}(p)\} \enspace .$$

In this work we are interested in the *betweenness centrality* of the vertices of a graph.

Definition 2. [3, 15] Given a graph $G = (V, E)$, the *betweenness centrality of a vertex* $v \in V$ is defined as[2]

$$\mathsf{b}(v) = \frac{1}{n(n-1)} \sum_{p_{uw} \in \mathbb{S}_G} \frac{\mathbb{1}_{\mathcal{T}_v}(p)}{\sigma_{uv}} \enspace .$$

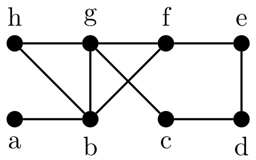
Figure 1 shows an example of betweenness values for the vertices of a (undirected, unweighted) graph. It is intuitive, by looking at the graph, that vertices b and g should have higher betweenness than the others, given that they somehow act as bridges between two sides of the network, and indeed that is the case.

It is easy to see that $\mathsf{b}(v) \in [0, 1]$. Brandes [9] presented an algorithm to compute the betweenness centrality for all $v \in V$ in time $O(nm)$ for unweighted graphs and $O(nm + n^2 \log n)$ for weighted graphs.

We present many variants of betweenness in Sect. 6.

[1] Note that even if $p_\emptyset = \emptyset$, the set $\{p_\emptyset\}$ is not empty. It contains one element.

[2] We use the normalized version of betweenness as we believe it to be more suitable for presenting approximation results.



(a) Example graph

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

(b) Betweenness values

Figure 1: Example of betweenness values

3.2 Vapnik-Chervonenkis dimension

The Vapnik-Chernovenkis (VC) dimension of a class of subsets defined on a set of points is a measure of the complexity or expressiveness of such class [38]. Given a probability distribution on the set of points, a finite bound on the VC-dimension of the class of subsets implies a bound on the number of random samples required to approximate the probability of each subset in the class with its empirical average. We outline here some basic definitions and results and refer the reader to the book by Mohri et al. [29] for an in-depth presentation.

Let D be a domain and $\mathcal{R}$ be a collection of subsets from D. We call $\mathcal{R}$ a *range set on* D. Given $B \subseteq D$, the *projection of* $\mathcal{R}$ *on* B is the set $P_{\mathcal{R}}(B) = \{B \cap A \ : \ A \in \mathcal{R}\}$. We say that the set B is *shattered* by $\mathcal{R}$ if $P_{\mathcal{R}}(B) = 2^B$.

Definition 3. The *Vapnik-Chervonenkis (VC) dimension of* $\mathcal{R}$, denoted as $\mathsf{VC}(\mathcal{R})$, is the cardinality of the largest subset of D that is shattered by $\mathcal{R}$.

Note that a range space (X, R) with an arbitrary large set of points X and an arbitrary large family of ranges R can have a bounded VC-dimension. A simple example is the family of intervals in $[0, 1]$ (i.e. X is all the points in $[0, 1]$ and R all the intervals $[a, b]$, such that $0 \leq a \leq b \leq 1$). Let $A = \{x, y, z\}$ be the set of three points $0 < x < y < z < 1$. No interval in R can define the subset $\{x, z\}$ so the VC-dimension of this range space is less than 3 [27, Lemma 10.3.1]. Another example is shown in Fig. 2.

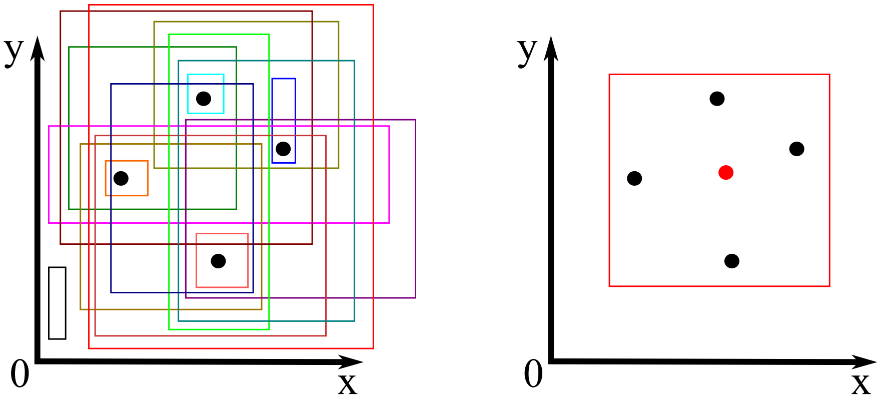


Figure 2: Example of range space and VC-dimension. The space of points is the plane $\mathbb{R}^2$ and the set R of ranges is the set of all *axis-aligned rectangles*. The figure on the left shows graphically that it is possible to shatter a set of four points using 16 rectangles. On the right instead, one can see that it is impossible to shatter five points, as, for any choice of the five points, there will always be one (the red point in the figure) that is internal to the convex hull of the other four, so it would be impossible to find an axis-aligned rectangle containing the four points but not the internal one. Hence $\mathsf{VC}((\mathbb{R}^2, R)) = 4$.

The main application of VC-dimension in statistics and learning theory is in computing the number of samples needed to approximate the probabilities of the ranges using their empirical averages as unbiased estimators. Formally, let $X_1^k = (X_1, \ldots, X_k)$ be a collection of independent identically distributed random variables taking values in D, sampled according to some distribution ϕ defined on the elements of D. For a set $A \subseteq D$, let $\phi(A)$ be the probability that a sample from ϕ belongs to the set A, and let the *empirical average* of $\phi(A)$ on X_1^k be

$$\phi_{X_1^k}(A) = \frac{1}{k} \sum_{j=1}^{k} \mathbb{1}_A(X_j) \ .$$

The empirical average of $\phi(A)$ can be used as an *unbiased* estimator for $\phi(A)$.

Definition 4. Let $\mathcal{R}$ be a range set on D and ϕ be a probability distribution on D. For $\varepsilon \in (0,1)$, an *ε-approximation to* $(\mathcal{R}, \phi)$ is a bag S of elements of D such that

$$\sup_{A \in \mathcal{R}} |\phi(A) - \phi_S(A)| \leq \varepsilon \ .$$

When an upper bound to the VC-dimension of $\mathcal{R}$ is available, it is possible to build an ε-approximation by sampling points of the domain according to the distribution ϕ.

Theorem 1 (Thm. 2.12 [17] (see also [23]))**.** *Let $\mathcal{R}$ be a range set on a domain D with $\mathsf{VC}(\mathcal{R}) \leq d$, and let ϕ be a distribution on D. Given $\varepsilon, \delta \in (0,1)$ let S be a collection of $|S|$ points from D sampled according to ϕ, with*

$$|S| = \frac{c}{\varepsilon^2}\left(d + \ln \frac{1}{\delta}\right) \tag{1}$$

where c is an universal positive constant. Then S is an ε-approximation to $(\mathcal{R}, \phi)$ with probability at least $1 - \delta$.

The constant c is approximately 0.5 [25]. It is possible to obtain *relative* guarantees on the approximation.

Definition 5. Let $\mathcal{R}$ be a range set on D and ϕ be a probability distribution on D. For $p, \varepsilon \in (0,1)$, a *relative (p, ε)-approximation to* $(\mathcal{R}, \phi)$ is a bag S of elements from D such that

- For any $A \in \mathcal{R}$ such that $\phi(A) \geq p$, we have

$$|\phi(A) - \phi_S(A)| \leq \varepsilon \phi(A) \ .$$

- For any $B \in \mathcal{R}$ such that $\phi(B) < p$, we have $\phi_S(B) \leq (1 + \varepsilon)p$.

Theorem 2 (Thm. 2.11 [17])**.** *Let $\mathcal{R}$ be a range set on a domain D with $\mathsf{VC}(\mathcal{R}) \leq d$, and let ϕ be a distribution on D. Given $\varepsilon, \delta, p \in (0,1)$ let S be a collection of $|S|$ points from D sampled according to ϕ, with*

$$|S| \geq \frac{c'}{\varepsilon^2 p}\left(d \log \frac{1}{p} + \log \frac{1}{\delta}\right) \tag{2}$$

where c' is an absolute positive constant. Then S is a relative (p, ε)-approximation to $(\mathcal{R}, \phi)$ with probability at least $1 - \delta$.

It is important to mention that if $\mathsf{VC}(\mathcal{R})$ and/or the upper bound d do not depend on $|D|$ or on $|\mathcal{R}|$ neither do the sample sizes presented in Thm. 1 and 2. This will make our algorithms scale well as the size of the network increases.

4 A range set of shortest paths

We now define a range set of the shortest paths of a graph $G = (V, E)$, and present a strict upper bound to its VC-dimension. We use the range set and the bound in the analysis of our algorithms for estimating the betweenness centrality of vertices of G.

The range set $\mathcal{R}_G$ is defined on the set $\mathbb{S}_G$ of all shortest paths between vertices of G. It contains, for each vertex $v \in V$, the set $\mathcal{T}_v$ of shortest paths that v is internal to:

$$\mathcal{R}_G = \{\mathcal{T}_v \ : \ v \in V\} \ .$$

Given a graph G let $\mathsf{H}(G)$ be the maximum integer d such that there are $h = \lfloor \log_2(d-2) \rfloor + 1$ shortest paths $\{p_1, \ldots, p_h\}$ in $\mathbb{S}_G$ of size at least d and such that the collection $\{\mathsf{Int}(p_1), \ldots, \mathsf{Int}(p_h)\}$ is an anti-chain.

Lemma 1. $\mathsf{VC}(\mathcal{R}_G) \leq \lfloor \log_2(\mathsf{H}(G) - 2) \rfloor + 1$.

Proof. Let $\ell > \lfloor \log_2(\mathsf{H}(G) - 2) \rfloor + 1$ and assume for the sake of contradiction that $\mathsf{VC}(\mathcal{R}_G) = \ell$. From the definition of VC-dimension there is a set $Q \subseteq \mathbb{S}_G$ of size ℓ that is shattered by $\mathcal{R}_G$. For any two shortest paths p', p'' in Q we must have neither $\mathsf{Int}(p') \subseteq \mathsf{Int}(p'')$ nor $\mathsf{Int}(p'') \subseteq \mathsf{Int}(p')$, otherwise one of the two paths would appear in all ranges where the other one appears, and so it would be impossible to shatter Q. Then Q must be such that the collection of the sets of internal vertices of the paths in Q form an anti-chain. From this and from the definition of $\mathsf{H}(G)$ we have that Q must contain a path p of size $|p| \leq \mathsf{H}(G)$. There are $2^{\ell-1}$ non-empty subsets of Q containing the path p. Let us label these non-empty subsets of Q containing p as $S_1, \ldots, S_{2^{\ell-1}}$, where the labelling is arbitrary. Given that Q is shattered, for each set S_i there must be a range R_i in $\mathcal{R}_G$ such that $S_i = Q \cap R_i$. Since all the S_i's are different from each other, then all the R_i's must be different from each other. Given that p belongs to each S_i, then p must also belong to each R_i, that is, there are $2^{\ell-1}$ distinct ranges in $\mathcal{R}_G$ containing p. But p belongs only to the ranges corresponding to internal vertices of p, i.e., to vertices in $\mathsf{Int}(p)$. This means that the number of ranges in $\mathcal{R}_G$ that p belongs to is equal to $|p| - 2$. But $|p| \leq \mathsf{H}(G)$ by definition of $\mathsf{H}(G)$, so p can belong to at most $\mathsf{H}(G) - 2$ ranges from $\mathcal{R}_G$. Given that $2^{\ell-1} > \mathsf{H}(G) - 2$, we reached a contradiction and there cannot be $2^{\ell-1}$ distinct ranges containing p, hence not all the sets S_i can be expressed as $Q \cap R_i$ for some $R_i \in \mathcal{R}_G$. Then Q cannot be shattered and $\mathsf{VC}(\mathcal{R}_G) \leq \lfloor \log_2(\mathsf{H}(G) - 2) \rfloor + 1$. □

Computing $\mathsf{H}(G)$ is not a viable option, but given that $\mathsf{H}(G) \leq \mathsf{VD}(G)$, we have the following corollary.

Corollary 1. $\mathsf{VC}(\mathcal{R}_G) \leq \lfloor \log_2(\mathsf{VD}(G) - 2) \rfloor + 1$.

Unique shortest paths In the restricted case when the graph is undirected and every pair of distinct vertices has either none or a unique shortest path between them, the VC-dimension of $\mathcal{R}_G$ reduces to a *constant*. This is a somewhat surprising result with interesting consequences. From a theoretical point of view, it suggests that there should be other characteristic quantities of the graph different from the vertex diameter that control the VC-dimension of the range set of shortest paths, and these quantities are constant on graph with unique shortest paths between vertices. From a more practical point of view, we will see in Sect. 5 that this result has an impact on the sample size needed to approximate the betweenness centrality of networks where the unique-shortest-path property is satisfied or even enforced, like road networks [16]. In particular, the resulting sample size will be *completely independent* from any characteristic of the network, and will only be a function of the parameters controlling the desired approximation guarantees.

Lemma 2. *Let $G = (V, E)$ be an undirected graph with $|\mathcal{S}_{uv}| \leq 1$ for all pairs $(u, v) \in V \times V$. Then* $\mathsf{VC}(\mathcal{R}_G) \leq 3$.

Proof. First of all, notice that in this restricted setting, if two different shortest paths p_1 and p_2 meet at a vertex u, then they either go on together or they separate never to meet again at any other vertex $v \neq u$. This is easy to see: if they could separate at u and then meet again at some v, then there would be two distinct shortest paths between u and v, which is a contradiction of the hypothesis. Let us denote this fact as F.

Assume now that $\mathsf{VC}(\mathcal{R}_G) > 3$, then there must be a set $Q = \{p_1, p_2, p_3, p_4\}$ of four shortest paths that can be shattered by $\mathcal{R}_G$. Then there is a vertex w such that $\mathcal{T}_w \cap Q = Q$, i.e., all paths in Q go through w. Let x be the farthest predecessor of w along p_1 that p_1 shares with some other path from Q, and let y be the farthest successor of w along p_1 that p_1 shares with some other path from Q. It is easy to see that if either x or y (or both) do not exist, then Q cannot be shattered, as we would incur in a contradiction of fact F.

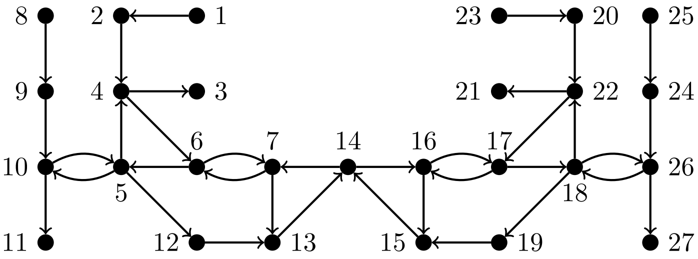


Figure 3: Directed graph $G = (V, E)$ with $|\mathcal{S}_{uv}| \leq 1$ for all pairs $(u, v) \in V \times V$ and such that it is possible to shatter a set of four paths.

Let us then assume that both x and y exist. Let $Q_x = \mathcal{T}_x \cap Q$ and $Q_y = \mathcal{T}_y \cap Q$. Because of fact F, all paths in Q_x must go through the same vertices between x and w and all paths in Q_y must go through the same vertices between w and y. This also means that all paths in $Q_x \cap Q_y$ must go through the same vertices between x and y. If $Q_x \cup Q_y \neq Q$, let $p^* \in Q \setminus (Q_x \cup Q_y)$. Then from the definition of x and y and from fact F we have that there is no vertex v such that $\mathcal{T}_v \cap Q = \{p_1, p^*\}$, which implies that Q can not be shattered.

Suppose from now on that $Q_x \cup Q_y = Q$. If $Q_x \cap Q_y = Q$, then all the paths in Q go through the same vertices between x and y. From this and the definition of x and y we have that there is no vertex v such that, for example, $\mathcal{T}_v \cap Q = \{p_1, p_2\}$, hence Q cannot be shattered. Suppose instead that $Q_x \cap Q_y \neq Q$ and let $S = (Q_x \cap Q_y) \setminus \{p_1\}$. If $S \neq \emptyset$ then there is at least a path $p' \in S$ which, from the definition of S and fact F, must go through all the same vertices as p_1 between x and y. Moreover, given that $Q_x \cap Q_y \neq Q$, there must be a path $p^* \in Q \setminus \{p_1\}$ different from p_1 such that $p^* \notin S$. Then, from the definition of x, y, and S, and from the existence of p', there can be no vertex v such that $\mathcal{T}_v \cap Q = \{p_1, p^*\}$, hence Q cannot be shattered. Assume now that $S = \emptyset$ and consider the case $Q_x = \{p_1, p_2, p_3\}$, $Q_y = \{p_1, p_4\}$ (all other cases follow by symmetry with this case). Consider the set $\{p_1, p_3\}$. From the definition of x and Q_x, and from fact F we have that there can not be a vertex v between the end point of p_1 before x and w such that $\mathcal{T}_v \cap Q = \{p_1, p_3\}$. At the same time, from the definition of y and from fact F, we have that such a v can not be between w and the end point of p_1 after y. This implies that Q can not be shattered.

We showed that in all possible cases we reached a contradiction and Q, which has size 4, can not be shattered by $\mathcal{R}_G$. Hence $\mathsf{VC}(\mathcal{R}_G) \leq 3$. □

It is natural to ask whether the above lemma or a similar result also holds for *directed* graphs. Fact F does not hold for directed graphs so the above proof does not extend immediately. In Fig. 3 we show a directed graph. For any pair of vertices in the graph, there is at most a unique shortest path connecting them. Consider now the following four directed shortest paths:

- $p_A = \{1, 2, 4, 6, 7, 13, 14, 16, 17, 18, 22, 21\}$
- $p_B = \{8, 9, 10, 5, 12, 13, 14, 16, 17, 18, 26, 27\}$
- $p_C = \{25, 24, 26, 18, 19, 15, 14, 7, 6, 5, 4, 3\}$
- $p_D = \{23, 20, 22, 17, 16, 15, 14, 7, 6, 5, 10, 11\}$

It is easy to check that the set $\{p_A, p_B, p_C, p_D\}$ is shattered, so the above lemma is not true for directed graphs. It is an open question whether it is true for a different constant.

4.1 Tightness

The bound presented in Corol. 1 (and therefore Lemma 1) is strict in the sense that for each $d \geq 1$ we can build a graph G_d with vertex-diameter $\mathsf{VD}(G_d) = 2^d + 1$ and such that the range set $\mathcal{R}_{G_d}$ associated to the set of shortest paths of G_d has VC-dimension exactly $d = \lfloor \log_2(\mathsf{VD}(G_d) - 2) \rfloor + 1$.

We now introduce a class $\mathcal{G} = (G_d)_{d\geq 1}$ of graphs indexed by d. The graphs in $\mathcal{G}$ are the ones for which we can show the tightness of the bound to the VC-dimension of the associated range set. We call the graph $G_d \in \mathcal{G}$ the d^{th} *concertina graph*. Figure 4 shows G_1, G_2, G_3, and G_4. The generalization to higher values of d is be straightforward. By construction, $\mathsf{VD}(G_d) = 2^d + 1$, so that $\lfloor \log_2(\mathsf{VD}(G_d) - 2) \rfloor + 1 = d$. The $3(2^{d-1})$ vertices of G_d can be partitioned into three classes, *top*, *bottom*, and *middle*, according to their location in a drawing of the graph similar to those in Fig. 4. G_d has $2^{d-1} - 1$ top vertices, $2^{d-1} - 1$ bottom vertices, and $2^{d-1} + 2$ middle vertices. For each top vertex v, let $\mathsf{f}(v)$ be the *corresponding bottom vertex*, i.e., the bottom vertex u whose neighbors are the same middle vertices that are neighbors of v. Among the middle vertices, the two with degree 1 are special and are called the *end vertices* of G_d and denoted as v_ℓ and v_r, where the labels can be arbitrarily assigned. We now build a set Q of d shortest paths from v_ℓ to v_r and show that it is shattered by $\mathcal{R}_{G_d}$, therefore proving that $\mathsf{VC}(\mathcal{R}_{G_d}) \geq d$. This fact, together with Coroll. 1, allows us to conclude that $\mathsf{VC}(\mathcal{R}_{G_d}) = d$.

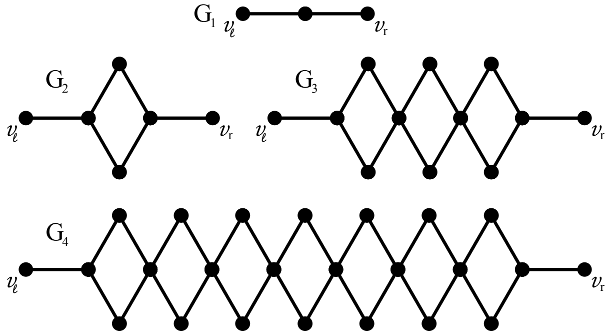


Figure 4: Examples of concertina graphs G_d for $d = 1, 2, 3, 4$.

Lemma 3. $\mathsf{VC}(\mathcal{R}_{G_d}) = d$.

Proof. As we said, the d paths in Q go from v_ℓ to v_r. From this and the definition of G_d it should be clear that they must go through all the middle vertices of G_d. Consider now the set $S = 2^Q \setminus \{Q, \emptyset\}$. We now build a map r from the elements of S to the set of top and bottom vertices of G_d. We can partition S in two sets A and B as follows: for each unordered pair (s', s'') of elements in S such that $s' \cap s'' = \emptyset$ and $s' \cup s'' = Q$ we put s' in A and s'' in B. It is easy to see that the size of A ($|A| = 2^{d-1} - 1$) equals the number of top vertices of G_d, and analogously for B and the number of bottom vertices. The bijection r will map the elements of A to the top vertices of G_d and the elements of B to the bottom vertices. For each $s' \in A$, let $\mathsf{c}(s')$ be the unique element s'' of B such that $s' \cap s'' = \emptyset$ and $s' \cup s'' = Q$ (i.e., $\mathsf{c}(s') = Q \setminus s'$). Let r_A be an arbitrary one-to-one map from the elements of A to the top vertices of G_d. Consider now the inverse map r_A^{-1} from the top vertices to the elements of A. We can create another map r_B from B to the bottom vertices of G_d that maps the element $\mathsf{c}(\mathsf{r}_A^{-1}(v))$ of B to the bottom vertex $\mathsf{f}(v)$ corresponding to v, for each top vertex v. A path $p \in Q$ goes through a top vertex v if and only if $p \in \mathsf{r}_A^{-1}(v)$. Analogously, p goes through a bottom vertex u if and only if $p \in \mathsf{r}_B^{-1}(u)$. It is easy to see that, if we combine r_A and r_B, we obtain a map r from S to the set of top and bottom vertices of G_d. An example of a possible r for G_3 is presented in Fig. 5.

We now show that for each $s \in S$, $s = Q \cap \mathcal{T}_{\mathsf{r}(s)}$. This is easy to see as $\mathsf{r}(s)$ is internal to all paths in s, by definition of $\mathsf{r}(s)$ and of the paths in Q. On the other end, no path from $\mathsf{c}(s)$ goes through $\mathsf{r}(s)$ because it goes through the corresponding vertex of $\mathsf{r}(s)$ (top if $\mathsf{r}(s)$ is a bottom vertex, bottom otherwise). It is also straightforward to see that, if we let v_Q be any arbitrary middle vertex different from v_ℓ or v_r, we have $Q = Q \cap \mathcal{T}_{v_Q}$, given that all paths in Q go through all the middle vertices. Also, given that v_ℓ is not internal to any path, we have $\emptyset = Q \cap \mathcal{T}_{v_\ell}$. Then all subsets of Q can be expressed as the intersection between Q and a range from $\mathcal{R}_{G_d}$, which means that Q can be shattered and therefore $\mathsf{VC}(\mathcal{R}_{G_d}) \geq d$.

From Corol 1 we know that $\mathsf{VC}(\mathcal{R}_{G_d}) \leq d$, so it must be $\mathsf{VC}(\mathcal{R}_{G_d}) = d$. □

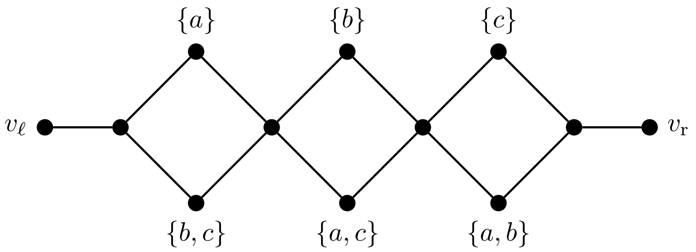


Figure 5: An example of the r map for G_3. The set next to each top and bottom vertex u is the set s such that $\mathsf{r}(s) = u$.

The upper bound presented in Lemma 2 for the case of unique shortest paths is also strict in the same sense.

Lemma 4. *There is a graph $G = (V, E)$ with $|\mathcal{S}_{uv}| \leq 1$ for all pairs $(u, v) \in V \times V$ such that the range set $\mathcal{R}_G$ associated to the shortest paths in G has VC-Dimension exactly* 3.

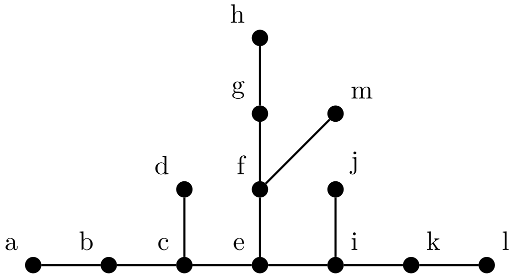


Figure 6: Graph G with $\mathsf{VC}(\mathcal{R}_G) = 3$.

Proof. Consider the graph G in Fig. 6. Let $p_1 = (a, b, c, e, i, j)$, $p_2 = (m, f, e, i, k, l)$, $p_3 = (d, c, e, f, g, h)$ be three paths. We now show that $Q = \{p_1, p_2, p_3\}$ can be shattered by $\mathcal{R}_G$, which implies $\mathsf{VC}(\mathcal{R}_G) \geq 3$. We have $\emptyset = Q \cap \mathcal{T}_a$, $\{p_1\} = Q \cap \mathcal{T}_b$, $\{p_2\} = Q \cap \mathcal{T}_k$, $\{p_3\} = Q \cap \mathcal{T}_g$, $\{p_1, p_2\} = Q \cap \mathcal{T}_i$, $\{p_1, p_3\} = Q \cap \mathcal{T}_c$, $\{p_2, p_3\} = Q \cap \mathcal{T}_f$, $\{p_1, p_2, p_3\} = Q \cap \mathcal{T}_e$, Hence all subsets of Q can be expressed as the intersection between Q and some range in $\mathcal{R}_G$ which means that Q can be shattered and $\mathsf{VC}(\mathcal{R}_G) \geq 3$. Lemma 2 gives us an upper bound $\mathsf{VC}(\mathcal{R}_G) \leq 3$, so we can conclude that $\mathsf{VC}(\mathcal{R}_G) = 3$. □

Although the example in Fig. 6 is a tree, this is not a requirement for either Lemma 2 or Lemma 4: in a weighted graph with cycles (i.e., not a tree) the weights may be such that there is a unique shortest path between any pair of connected vertices.

5 Algorithms

In this section we present our algorithms to compute a set of approximations for the betweenness centrality of the (top-K) vertices in a graph through sampling, with probabilistic guarantees on the quality of the approximations.

5.1 Approximation for all the vertices

The intuition behind the algorithm to approximate the betweenness values of all vertices is the following. Given a graph $G = (V, E)$ with vertex-diameter $\mathsf{VD}(G)$ and two parameters $\varepsilon, \delta \in (0, 1)$ we first compute a sample size r using (1) with

$$d = \lfloor \log_2(\mathsf{VD}(G) - 2) \rfloor + 1 \ .$$

The resulting sample size is

$$r = \frac{c}{\varepsilon^2}\left(\lfloor \log_2(\mathsf{VD}(G) - 2) \rfloor + 1 + \ln \frac{1}{\delta} \right) \ . \tag{3}$$

This is sufficient to achieve the desired accuracy (expressed through ε) with the desired confidence (expressed through $1-\delta$). The algorithm repeats the following steps r times: *1.* it samples a pair u, v of distinct vertices uniformly at random, *2.* it computes the set $\mathcal{S}_{uv}$ of all shortest paths between u and v, *3.* it selects a path p from $\mathcal{S}_{uv}$ uniformly at random, *4.* it increases by $1/r$ the betweenness estimation of each vertex in $\mathsf{Int}(p)$. Note that if the sampled vertices u and v are not connected, we can skip steps 3 and 4 because we defined $\mathcal{S}_{uv} = \{p_\emptyset\}$. Denoting with S the set of the sampled shortest paths, the *unbiased* estimator $\tilde{\mathsf{b}}(w)$ for the betweenness $\mathsf{b}(w)$ of a vertex w is the sample average

$$\tilde{\mathsf{b}}(w) = \frac{1}{r}\sum_{p \in S} \mathbb{1}_{\mathsf{Int}(p)}(w) = \frac{1}{r}\sum_{p \in S} \mathbb{1}_{\mathcal{T}_w}(p) \ .$$

There are two crucial steps in this algorithm: the computation of $\mathsf{VD}(G)$ and the sampling of a path uniformly at random from $\mathcal{S}_{uv}$. We first deal with the latter, and then present a linear-time constant-factor approximation algorithm for $\mathsf{VD}(G)$. Algorithm 1 presents the pseudocode of the algorithm, including the steps to select a random path. The `computeAllShortestPaths(`u, v`)` on line 8 is a call to a modified Dijkstra's (or BFS) algorithm to compute the set $\mathcal{S}_{uv}$, with the same modifications as [9]. The `getDiameterApprox()` procedure computes an approximation for $\mathsf{VD}(G)$.

Unique shortest paths When, for each pair (u, v) of vertices of G, either there is a unique shortest path from u to v or v is unreachable from u, then one can apply Lemma 2 and obtain a smaller sample size

$$r = \frac{c}{\varepsilon^2}\left(3 + \ln \frac{1}{\delta}\right)$$

to approximate the betweenness values of all the vertices. This is an interesting result: the number of samples needed to compute a good approximation to all vertices is a *constant* and completely *independent from G*. Intuitively, this means that the algorithm is extremely fast on graphs with this property. Unique shortest paths are common or even enforced in road networks by slightly perturbing the edge weights or having a deterministic tie breaking policy [16].

Sampling a shortest path Our procedure to select a random shortest path from $\mathcal{S}_{uv}$ is inspired by the dependencies accumulation procedure used in Brandes' exact algorithm [9]. Let u and v be the vertices sampled by our algorithm (Step 7 of Alg. 1). We assume that u and v are connected otherwise the only possibility is to select the empty path $p_\emptyset$. Let y be any vertex belonging to at least one shortest path from u to v. Following Brandes [9], we can compute σ_{uy} and $\mathcal{S}_{uy}$ while we compute the set $\mathcal{S}_{uv}$ of all the shortest paths from u to v. We can then use this information to select a shortest path p uniformly at random from $\mathcal{S}_{uv}$ as follows. For each vertex w let $P_u(w)$ be the subset of neighbors of w that are *predecessors* of w along the shortest paths from u to w. Let $p^* = \{v\}$. Starting from v, we select one of its predecessors $z \in P_u(v)$ using weighted random sampling: each $z \in P_u(v)$ has probability $\sigma_{uz}/\sum_{w \in P_u(v)} \sigma_{uw}$ of being sampled. We add z to p^* and then repeat the procedure for z. That is, we select one of z's predecessors from $P_u(z)$ using weighted sampling and add it to p^*, and so on until we reach u. Note that we can update the estimation of the betweenness of the internal vertices along p^* (the only ones for which the estimation is updated) as we compute p^*.

Lemma 5. *The path p^* built according to the above procedure is selected uniformly at random among the paths in $\mathcal{S}_{uv}$.*

Proof. The probability of sampling $p^* = (u, z_1, \dots, z_{|p^*|-2}, v)$ equals to the product of the probabilities of sampling the vertices internal to p^*, hence

$$\Pr(p^*) = \frac{\sigma_{uz_{|p^*|-2}}}{\sigma_{uv}} \frac{\sigma_{uz_{|p^*|-3}}}{\sigma_{uz_{|p^*|-2}}} \cdots \frac{1}{\sigma_{uz_2}} = \frac{1}{\sigma_{uv}}$$

where we used [9, Lemma3] which tells us that for $w \neq u$,

$$\sigma_{uw} = \sum_{j \in P_u(w)} \sigma_{uj}$$

and the fact that for z_1, which is a neighbor of u, $\sigma_{uz_1} = 1$. □

Algorithm 1: Computes approximations $\tilde{\mathsf{b}}(v)$ of the betweenness centrality $\mathsf{b}(v)$ for all vertices $v \in V$.

```
   Input  : Graph $G = (V, E)$ with $|V| = n$, $\varepsilon, \delta \in (0,1)$
   Output: A set of approximations of the betweenness centrality of the vertices in $V$
 1 foreach $w \in V$ do
 2     $\tilde{\mathsf{b}}(v) \leftarrow 0$
 3 end
 4 $\mathsf{VD}(G) \leftarrow$getVertexDiameter($G$)
 5 $r \leftarrow (c/\varepsilon^2)(\lfloor \log_2(\mathsf{VD}(G) - 2) \rfloor + \ln(1/\delta))$
 6 for $i \leftarrow 1$ to $r$ do
 7     $(u, v) \leftarrow$sampleUniformVertexPair($V$)
 8     $\mathcal{S}_{uv} \leftarrow$computeAllShortestPaths($u, v$)
 9     if $\mathcal{S}_{uv} \neq \{p_\emptyset\}$ then
           //Random path sampling and estimation update
10         $j \leftarrow v$
11         $s \leftarrow v$
12         $t \leftarrow v$
13         while $t \neq u$ do
14             sample $z \in P_s(t)$ with probability $\sigma_{uz}/\sigma_{us}$
15             if $z \neq u$ then
16                 $\tilde{\mathsf{b}}(z) \leftarrow \tilde{\mathsf{b}}(z) + 1/r$
17                 $s \leftarrow t$
18                 $t \leftarrow z$
19             end
20         end
21     end
22 end
23 return $\{(v, \tilde{\mathsf{b}}(v)), v \in V\}$
```

Approximating the vertex-diameter The algorithm presented in the previous section requires the value of the vertex-diameter $\mathsf{VD}(G)$ of the graph G (line 4 of Alg. 1). Computing the exact value of $\mathsf{VD}(G)$ could be done by solving the All Pair Shortest Paths (APSP) problem, and taking the shortest path with the maximum size. Algorithms for exactly solving APSP problem such as Johnson's which runs in $O(V^2 \log V + VE)$ or Floyd-Warshall's ($\Theta(V^3)$), would defeat our purposes: once we have all the shortest paths for the computation of the diameter, we could as well compute the betweenness of all the vertices exactly. Given that Thm. 1 (and Thm. 2) only requires an upper bound to the VC-dimension of the range set, an approximation of the vertex-diameter would be sufficient for our purposes. Several refined algorithms for approximating the diameter are known [2, 7, 36], with various running times and quality of approximations. We briefly present a well-known and simple approximation algorithm that has the right balance of accuracy and speed for our purposes.

Let $G = (V, E)$ be an *undirected* graph where *all the edge weights are equal.* It is a well-known result that one can obtain a 2-approximation $\widetilde{\mathsf{VD}}(G)$ of the vertex-diameter $\mathsf{VD}(G)$ of G in time $O(V + E)$ in the following way: 1. select a vertex $v \in V$ uniformly at random, 2. compute the shortest paths from v to all other vertices in V, and 3. finally take $\widetilde{\mathsf{VD}}(G)$ to be the sum of the lengths of the two shortest paths with maximum size (which equals to the two longest shortest paths) from v to two distinct other nodes u and w. The approximation guarantee follows from the next lemma.

Lemma 6. $\mathsf{VD}(G) \leq \widetilde{\mathsf{VD}}(G) \leq 2\mathsf{VD}(G)$.

Proof. Let $v \in V$ be a vertex that we choose uniformly at random from the set V. Let also $u, w \in V$ be the two vertices such that the sum of the sizes of the shortest paths p_{vu} and p_{vw} is maximized among all the shortest paths that have v as a source. We have $\widetilde{\mathsf{VD}}(G) \leq 2\mathsf{VD}(G)$ because $|p_{vu}|, |p_{vw}| \leq \mathsf{VD}(G)$, so $|p_{vu}| + |p_{vw}| \leq 2\mathsf{VD}(G)$. To see that $\widetilde{\mathsf{VD}}(G) \geq \mathsf{VD}(G)$, consider a pair of vertices x and z such that the length of a shortest path between x and z is equal to $\mathsf{VD}(G)$. Let p_{xv} be a shortest path between x and v and let p_{vz} be a shortest path between v and z. From the properties of the shortest paths p_{vu}, p_{vw} we have $|p_{vu}| + |p_{vw}| \geq |p_{vx}| + |p_{vz}|$. Since the graph is undirected $|p_{s,t}| = |p_{t,s}|$ for every $s, t \in V$. Therefore:

$$\widetilde{\mathsf{VD}}(G) = |p_{vu}| + |p_{vw}| \geq |p_{vx}| + |p_{vz}| = |p_{xv}| + |p_{vz}| \geq \mathsf{VD}(G) \ .$$

For the last inequality we used the fact that since $\mathsf{VD}(G)$ is the size of the shortest path from x to z, then every other path (in this case p'_{xz} which is the merge of p_{xv} and p_{vz}) has greater or equal length from $p_{x,z}$. □

In case we have multiple connected components in G, we compute an upper bound to the vertex diameter of each component separately by running the above algorithm on each component, and then taking the maximum. The connected components can be computed in $O(n + m)$ by traversing the graph in a Breadth-First-Search (BFS) fashion starting from a random v. The time complexity of the approximation algorithm in the case of multiple connected components is again $O(n + m)$ since the sum of the vertices of individual components is n and the sum of edges is m.

The use of the above 2-approximation in the computation of the sample size from line 6 of Alg. 1 results in at most c/ε^2 additional samples than if we used the exact value $\mathsf{VD}(G)$. The computation of $\widetilde{\mathsf{VD}}(G)$ does not affect the running time of our algorithm: for the construction of the first sample we can reuse the shortest paths from the sampled vertex v that we used to obtain the approximation. Specifically, we can sample a new vertex $u \neq v$ and then choose with uniform probability one of the (already computed) shortest paths between v and u.

If the graph is directed and/or not all edge weights are equal, the computation of a good approximation to $\mathsf{VD}(G)$ becomes more problematic. In particular, notice that there is no relationship between $\mathsf{VD}(G)$ and $\mathsf{diam}(G)$ when G is weighted, as the shortest path with maximum size may not be the shortest path with maximum weight. In these cases, one can use the size (number of vertices) of the largest Weakly Connected Component (WCC), as a loose upper bound to $\mathsf{VD}(G)$. The WCC's can again be computed in $O(n + m)$ using BFS. This quantity can be as high as n but for the computation of the sample size we use its logarithm, mitigating the crudeness of the bound. In this case our sample size is comparable to that proposed by Brandes and Pich [11]. Nevertheless the amount of work done per sample by our algorithm is still much smaller (see Sect. 5.3 and 7 for more details). In practice, it is possible that the nature of the network suggests a much better upper bound to the vertex-diameter of the graph, resulting in a smaller sample size.

Analysis Algorithm 1 offers probabilistic guarantees on the quality of all approximations of the betweenness centrality.

Lemma 7. *With probability at least $1-\delta$, all the approximations computed by the algorithm are within ε from their real value:*

$$\Pr\left(\exists v \in V \ s.t. \ |\mathsf{b}(v) - \tilde{\mathsf{b}}(v)| > \varepsilon\right) < \delta \enspace .$$

Proof. For each $p_{uv} \in \mathbb{S}_G$ let

$$\pi_G(p_{uv}) = \frac{1}{n(n-1)} \frac{1}{\sigma_{uv}} \enspace .$$

It is easy to see that π_G is a probability distribution and $\pi_G(p_{uv})$ is the probability of sampling the path p_{uv} during an execution of the loop on line 6 in Alg. 1, given the way that the vertices u and v are selected and Lemma 5.

Consider the range set $\mathcal{R}_G$ and the probability distribution π_G. Let S be the set of paths sampled during the execution of the algorithm. For r as in (3), Thm. 1 tells us that the sample S is a ε-approximation to $(\mathcal{R}_G, \pi_G)$ with probability at least $1-\delta$. Suppose that this is indeed the case, then from Def. 4 and the definition of $\mathcal{R}_G$ we have that

$$\left| \pi_G(\mathcal{T}_v) - \frac{1}{r} \sum_{p \in S} \mathbb{1}_{\mathcal{T}_v}(p) \right| = \left| \pi_G(\mathcal{T}_v) - \tilde{\mathsf{b}}(v) \right| \le \varepsilon, \forall v \in V \enspace .$$

From the definition of π_G we have

$$\pi_G(\mathcal{T}_v) = \frac{1}{n(n-1)} \sum_{p_{uw} \in \mathcal{T}_v} \frac{1}{\sigma_{uw}} = \mathsf{b}(v),$$

which concludes the proof. □

Time and space complexity. Clearly the runtime of the algorithm is dominated by the computation of the shortest path at each step, which takes time $O(|V| + |M|)$ if the graph is unweighted (BFS algorithm) and time $O(|E| + |V| \log |V|)$ otherwise (Dijkstra's algorithm with Fibonacci heap). This time must then be multiplied by r as in (3) to obtain the final time complexity. The space requirements are dominated by the amount of memory needed to store the graph, so they are either $O(|V|^2)$ if using an adjacency matrix, or $O(|V| + |E|)$ if using $|V|$ adjacency lists.

5.2 High-quality approximation of the top-K betweenness vertices

Very often in practice one is interested only in identifying the vertices with the highest betweenness centrality, as they are the "primary actors" in the network. We present here an algorithm to compute a very high-quality approximation of the set $\mathsf{TOP}(K,G)$ of the top-K betweenness vertices in a graph $G=(V,E)$. Formally, let $v_1, \dots, v_n$ be a labelling of the vertices in V such that $\mathsf{b}(v_i) \ge \mathsf{b}(v_j)$ for $1 \le i < j \le n$. Then $\mathsf{TOP}(K,G)$ is defined as the set of vertices with betweenness at least $\mathsf{b}(v_K)$:

$$\mathsf{TOP}(K,G) = \{(v, \mathsf{b}(v)), \ : \ v \in V \text{ and } \mathsf{b}(v) \ge \mathsf{b}(v_K)\} \enspace .$$

Note that $\mathsf{TOP}(K,G)$ may contain more than K vertices.

Our algorithm works in two phases. Each phase is basically a run of the algorithm for approximating the betweenness of all vertices. The two phases differ in the way they compute the number of paths to sample and the additional operations at the end of each phase. In the first phase, we compute a lower bound ℓ' to $\mathsf{b}(v_K)$. In the second phase we use ℓ' to compute the number of samples r needed to obtain a relative (ℓ', ε)-approximation to $(\mathcal{R}_G, \pi_G)$. We use r samples to approximate the betweenness of all vertices again, and return a collection of vertices that is, with high probability, a superset of $\mathsf{TOP}(K,G)$.

Let $\widetilde{\mathsf{VD}}(G)$ be an upper bound to the vertex-diameter of G. Given $\varepsilon, \delta \in (0,1)$, let δ', δ'' be two positive reals such that $(1-\delta')(1-\delta'') \ge (1-\delta)$. Let

$$r' = \frac{c}{\varepsilon^2}\left(\lfloor \log_2(\widetilde{\mathsf{VD}}(G) - 2) \rfloor + 1 + \log \frac{1}{\delta'} \right) \enspace .$$

Let $\tilde{\mathsf{b}}'_k$ be the K-th highest estimated betweenness obtained using Algorithm 1 where $r = r'$, and let $\ell' = \tilde{\mathsf{b}}_K - \varepsilon$, and

$$r'' = \frac{c'}{\varepsilon^2 \ell'}\left((\lfloor \log_2(\widetilde{\mathsf{VD}}(G) - 2)\rfloor + 1)\log\frac{1}{\ell'} + \log\frac{1}{\delta''}\right) \enspace .$$

We run Algorithm 1 with $r = r''$ and let $\tilde{\mathsf{b}}''_K$ be the so-obtained K-th highest estimated betweenness. Let $\ell'' = \min\{\tilde{\mathsf{b}}''(v)/(1+\varepsilon) \ : \ v \in V \text{ s.t. } \tilde{\mathsf{b}}''(v) \geq \tilde{\mathsf{b}}''_K\}$. We return the collection $\widetilde{\mathsf{TOP}}(K,G)$ of vertices v such that $\tilde{\mathsf{b}}''(v) * (1+\varepsilon)/(1-\varepsilon) \geq \ell''$:

$$\widetilde{\mathsf{TOP}}(K,G) = \left\{ v \in V \ : \ \tilde{\mathsf{b}}''(v)\frac{1+\varepsilon}{1-\varepsilon} \geq \ell'' \right\} \enspace .$$

The pseudocode of the algorithm is presented in Algorithm 2.

```
Algorithm 2: High-quality approximation of the top-$K$ betweenness vertices
  Input  : a graph $G = (V,E)$ with $|V| = n$, a positive integer $K \leq n$, real values $\varepsilon, \delta \in (0,1)$
  Output: a superset of $\mathsf{TOP}(K,G)$, with high-quality estimation of the betweenness for the vertices in
          the returned set.
  $\delta', \delta'' \leftarrow$ two positive reals such that $(1-\delta')(1-\delta'') \geq (1-\delta)$
  $\widetilde{\mathsf{VD}}(G) \leftarrow$ upper bound to $\mathsf{VD}(G)$
  //First phase
  $r' \leftarrow \frac{c}{\varepsilon^2}\left(\lfloor \log_2(\widetilde{\mathsf{VD}}(G) - 2)\rfloor + 1 + \log\frac{1}{\delta''}\right)$
  $B' = \{(v, \tilde{\mathsf{b}}'(v)) \ : \ v \in V\} \leftarrow$ output of Algorithm 1 with $r = r'$
  $\tilde{\mathsf{b}}'_K \leftarrow$ $K$-th highest betweenness value from $B'$, ties broken arbitrarily
  $\ell' \leftarrow \tilde{\mathsf{b}}'_K - \varepsilon$
  $r'' \leftarrow \frac{c'}{\varepsilon^2\ell'}\left((\lfloor \log_2(\widetilde{\mathsf{VD}}(G) - 2)\rfloor + 1)\log\frac{1}{\ell'} + \log\frac{1}{\delta''}\right)$
  //Second phase
  $B'' = \{(v, \tilde{\mathsf{b}}''(v)) \ : \ v \in V\} \leftarrow$ output of Algorithm 1 with $r = r''$
  $\tilde{\mathsf{b}}''_K \leftarrow$ $K$-th highest betweenness value from $B''$, ties broken arbitrarily
  $\ell'' \leftarrow \min\{\tilde{\mathsf{b}}''(v)/(1+\varepsilon) \ : \ v \text{ s.t. } \tilde{\mathsf{b}}''(v) \geq \tilde{\mathsf{b}}''_K\}$
  return $\{(v, \tilde{\mathsf{b}}''(v)) \ : \ v \in V \text{ s.t. } \tilde{\mathsf{b}}''(v)\frac{1+\varepsilon}{1-\varepsilon} \geq \ell''\}$
```

Analysis The following lemma shows the properties of the collection $\widetilde{\mathsf{TOP}}(K,G)$.

Lemma 8. *With probability at least* $1-\delta$*,*

1. $\mathsf{TOP}(K,G) \subseteq \widetilde{\mathsf{TOP}}(K,G)$*, and*
2. *for all* $v \in \widetilde{\mathsf{TOP}(K,G)}$ *we have* $|\tilde{\mathsf{b}}''(v) - \mathsf{b}(v)| \leq \varepsilon \mathsf{b}(v)$*, and*
3. *no vertex* $u \in \widetilde{\mathsf{TOP}}(K,G) \setminus \mathsf{TOP}(K,G)$ *has an estimated betweenness greater than* $\ell'(1+\varepsilon)$*.*

Proof. We start by proving 1. From Thm. 1 we know that, with probability at least $1-\delta'$, a sample of size r' is a ε-approximation to $(\mathcal{R}_G, \pi_G)$ and from Thm. 2 we have that with probability at least $1-\delta''$ a sample of size r'' is a relative (ℓ', ε)-approximation to $(\mathcal{R}_G, \pi_G)$. Suppose both these events occur, which happens with probability at least $1-\delta$. Then it is easy to see that $\ell' \leq \mathsf{b}(v_K)$, as there must be at least K vertices with exact betweenness greater or equal to ℓ'. Consider now ℓ''. Following the same reasoning as for ℓ', it should be clear that $\ell'' \leq \mathsf{b}(v_K)$. The vertices included in $\widetilde{\mathsf{TOP}}(K,G)$ are all and only those vertices that *may* have exact betweenness at least ℓ'', which implies that all vertices that have exact betweenness at least $\mathsf{b}(v_K)$ are included in $\widetilde{\mathsf{TOP}}(K,G)$. Points 2 and 3 in the thesis follow from the properties of the relative (ℓ', ε)-approximation (Def. 5). □

The advantage of using our algorithm to approximate the collection of top-K betweenness vertices consists in the very high-quality approximation of the betweenness values for the returned set of vertices: they are all within a multiplicative factor ε from their exact values. By reverse-sorting them according to the approximated betweenness, one can obtain a ranking that is very similar to the original exact one. Previous algorithms could not achieve such a good ranking as they were only able to approximate the betweenness values to within an additive error ε. The cost of computing the high quality approximation for the top-K vertices is the cost of an additional run of our algorithm to compute good approximations for all the vertices.

5.3 Discussion

Jacob et al. [19] and independently Brandes and Pich [11] present a sampling-based algorithm to approximate the betweenness centrality of all the vertices of the graph. The algorithm (which we call BP) creates a sample $S = \{v_1, \ldots, v_r\}$ of r vertices drawn uniformly at random and computes all the shortest paths between each v_i to all other vertices in the graph. Their estimation $\tilde{\mathsf{b}}_{\mathsf{BP}}(u)$ for $\mathsf{b}(u)$ is

$$\tilde{\mathsf{b}}_{\mathsf{BP}}(u) = \frac{1}{(n-1)r} \sum_{v_i \in S} \sum_{\substack{w \neq v_i \\ w \neq u}} \sum_{p \in \mathcal{S}_{v_i w}} \frac{\mathbb{1}_{\mathsf{Int}(p)}(u)}{|\mathcal{S}_{v_i w}|} \enspace .$$

As it was for Algorithm 1, the key ingredient to ensure a correct approximation for the betweenness centrality is the computation of the sample size r. Inspired by the work of Eppstein and Wang [14], Brandes and Pich [11] prove that, to obtain good (within ε) estimations for the betweenness of all vertices with probability at least $1 - \delta$, it must be

$$r \geq \frac{1}{2\varepsilon^2} \left(\ln n + \ln 2 + \ln \frac{1}{\delta} \right) \enspace .$$

From this expression it should be clear that this sample size is usually much larger than ours, as in practice $\mathsf{VD}(G) \ll n$. For the same reason, this algorithm would not scale well as the network size increases (see also Sect. 7).

Another interesting aspect in which our algorithm and BP differ is the *amount of work done per sample*. Our algorithm computes a single set $\mathcal{S}_{uv}$ for the sampled pair of vertices (u, v): it performs a run of Dijkstra's algorithm (or of BFS) from u, stopping when v is reached. BP instead computes all the sets $\mathcal{S}_{uw}$ from the sampled vertices u to all other vertices, again with a single run of Dijkstra or BFS, but without the "early-stopping condition" that we have when we reach v. Although in the worst case the two computations have the same time complexity[3], in practice we perform many fewer operations, as we can expect v not to always be very far from u and therefore we can terminate early. This fact has a huge impact on the running time. Our algorithm also touches *many fewer* edges than BP. The latter can touch all the edges in the graph *at every sample*, while our computation exhibits a much higher *locality*, exploring only a neighborhood of u until v is reached. The results of our experimental evaluation presented in Sect. 7 highlights this and other advantages of our method over the one from [11, 19]. In the future, we plan to investigate the possibility of using *bidirectional A^* search* [20, 34] to further speed up the computation for each sample of our algorithms.

The analysis of Algorithm 1 allow us to obtain a tighter analysis for the algorithm by Brandes and Pich [11] and Jacob et al. [19].

Lemma 9. *Fix $r > 0$ and let $w \in V$. Let $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ be the estimation of the betweenness $\mathsf{b}(w)$ computed by BP using r samples, and let $\tilde{\mathsf{b}}(w)$ be the estimation computed by Algorithm 1 using r samples. The estimator $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ is* uniformly better *than the estimator $\tilde{\mathsf{b}}(w)$. I.e., for any value of $\mathsf{b}(w)$ and r, we have*

$$\mathrm{Var}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)] \leq \mathrm{Var}[\tilde{\mathsf{b}}(w)] \enspace .$$

[3] It is a well-known open problem whether there is an algorithm to perform a single s,t-shortest path computation between a pair of vertices with smaller worst-case time complexity than the Single Source Shortest Path computation.

Proof. Consider the quantity

$$\mathrm{Var}[\tilde{\mathsf{b}}(w)] - \mathrm{Var}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)] \ .$$

We will show that the above quantity is greater than or equal to 0, proving the thesis. Since $\mathbb{E}[\tilde{\mathsf{b}}(w)] = \mathbb{E}[\tilde{\mathsf{b}}_{\mathsf{BF}}(w)] = \mathsf{b}(w)$ and using the definition of variance, we only need to show

$$\mathbb{E}[(\tilde{\mathsf{b}}(w))^2] - \mathbb{E}[(\tilde{\mathsf{b}}_{\mathsf{BF}}(w))^2] \geq 0 \ . \tag{4}$$

Let start from computing $\mathbb{E}[(\tilde{\mathsf{b}}_{\mathsf{BF}}(w))^2]$. For every vertex v, we define α_v as

$$\alpha_v = \frac{1}{n-1} \sum_{\substack{u \in V \\ u \neq v}} \sum_{p \in \mathcal{S}_{vu}} \frac{\mathbb{1}_{\mathsf{Int}(p)}(w)}{\sigma_{vu}} \ .$$

Note that

$$\mathsf{b}(w) = \frac{1}{n} \sum_{v \in V} \alpha(v) \ . \tag{5}$$

Consider, for $1 \leq i \leq r$, the contribution X_i to $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ of the paths computed from the i^{th} sampled vertex. X_i is a random variable that takes value α_v with probability $1/n$, for all $v \in V$. We have

$$\mathbb{E}[X_i] = \sum_{v \in V} \frac{1}{n} \alpha_v = \mathsf{b}(w) \text{ and } \mathbb{E}[X_i^2] = \frac{1}{n} \sum_{v \in V} \alpha_v^2, \forall 1 \leq i \leq r \ . \tag{6}$$

Clearly

$$\tilde{\mathsf{b}}_{\mathsf{BP}}(w) = \frac{1}{r} \sum_{i=1}^{r} X_i \ .$$

The variables X_i's are independent and identically distributed so

$$\mathbb{E}[(\tilde{\mathsf{b}}_{\mathsf{BP}}(w))^2] = \frac{1}{r^2} \mathbb{E}\left[\left(\sum_{i=1}^{r} X_i\right)^2\right] = \frac{1}{r^2} \sum_{i=1}^{r} \left(\mathbb{E}[X_i^2] + \sum_{j=i+1}^{r} (\mathbb{E}[X_i]\mathbb{E}[X_j])\right) = \frac{1}{r}\frac{1}{n} \sum_{v \in V} \alpha_v^2 + \frac{r-1}{r}(\mathsf{b}(w))^2, \tag{7}$$

where we used (6).

We now compute $\mathbb{E}[(\tilde{\mathsf{b}}(w))^2]$. Consider the contribution Y_i to $\tilde{\mathsf{b}}(w)$ of the i^{th} sampled path by Algorithm 1, for $1 \leq i \leq r$. Y_i is a random variable that takes value $\mathbb{1}_{\mathsf{Int}(p)}(w)$ with probability $\pi(p)$, for every shortest path $p \in \mathbb{S}_G$. We have

$$\mathbb{E}[Y_i] = \mathbb{E}[Y_i^2] = \mathsf{b}(w), \forall 1 \leq i \leq r \ . \tag{8}$$

By definition,

$$\tilde{\mathsf{b}}(w) = \frac{1}{r} \sum_{i=1}^{r} Y_i \ .$$

The random variables Y_i are independent and identically distributed so

$$\mathbb{E}[(\tilde{\mathsf{b}}(w))^2] = \frac{1}{r^2} \mathbb{E}\left[\left(\sum_{i=1}^{r} Y_i\right)^2\right] = \frac{1}{r^2} \sum_{i=1}^{r} \left(\mathbb{E}[Y_i^2] + \sum_{j=i+1}^{r} \mathbb{E}[Y_i]\mathbb{E}[Y_j]\right) = \frac{1}{r}\mathsf{b}(w) + \frac{r-1}{r}(\mathsf{b}(w))^2, \tag{9}$$

where we used (8).

We can now rewrite the left side of (4) using (5), (7), and (9):

$$\mathbb{E}[(\tilde{\mathsf{b}}(w))^2] - \mathbb{E}[(\tilde{\mathsf{b}}_{\mathsf{BF}}(w))^2] = \frac{1}{r}\mathsf{b}(w) + \frac{r-1}{r}(\mathsf{b}(w))^2 - \frac{1}{r}\frac{1}{n}\sum_{v \in V} \alpha_v^2 - \frac{r-1}{r}(\mathsf{b}(w))^2 = \frac{1}{r}\frac{1}{n}\sum_{v \in V} (\alpha_v - \alpha_v^2) \ .$$

Since $\alpha_v \in [0,1]$, we have $\alpha_v - \alpha_v^2 \geq 0$ for all v, and our proof is complete. □

The following lemma is an easy consequence of the above.

Lemma 10. *BP has lower expected Mean Squared Error than Algorithm 1:*

$$\mathbb{E}[\mathrm{MSE}_{\mathsf{BP}}] \le \mathbb{E}[\mathrm{MSE}] \ .$$

Proof. We have

$$\mathbb{E}[\mathrm{MSE}_{\mathsf{BP}}] = \mathbb{E}\left[\frac{1}{n}\sum_{v\in V}\left(\tilde{\mathsf{b}}_{\mathsf{BP}}(w) - \mathsf{b}(v)\right)^2\right] = \frac{1}{n}\sum_{v\in V}\mathbb{E}\left[\left(\tilde{\mathsf{b}}_{\mathsf{BP}}(w) - \mathsf{b}(v)\right)^2\right] = \frac{1}{n}\sum_{v\in V}\mathrm{Var}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)],$$

where we used the linearity of expectation and the fact that the estimator $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ is unbiased for $\mathsf{b}(w)$ ($\mathbb{E}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)] = \mathsf{b}(w)$). Analogously for Algorithm 1:

$$\mathbb{E}[\mathrm{MSE}] = \frac{1}{n}\sum_{v\in V}\mathrm{Var}[\tilde{\mathsf{b}}(w)] \ .$$

Hence

$$\mathbb{E}[\mathrm{MSE}] - \mathbb{E}[\mathrm{MSE}_{\mathsf{BP}}] = \frac{1}{n}\sum_{v\in V}\left(\mathrm{Var}[\tilde{\mathsf{b}}(w)] - \mathrm{Var}[\tilde{\mathsf{b}}_{\mathsf{BP}}(w)]\right),$$

and from Lemma 9 we have that each addend of the sum is non-negative and so is the above expression, concluding our proof. □

The estimator $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ has lower variance and MSE than $\tilde{\mathsf{b}}(w)$, which means that it can give better estimations of the betweenness values in practice using the same number of samples. Before always opting for BP (or for the algorithm by Geisberger et al. [16], whose estimators have even lower variance than those of BP) with a number of samples equal to the one in (3), one should nevertheless take into account two facts. Firstly, we do not currently have a proof that BP (or the algorithm from [16]) can compute, with a number of samples as in (3), a good (within $\pm\varepsilon$) approximation of the betweenness of all vertices with probability at least $1-\delta$. We conjecture this fact could be proven using pseudodimension [4, Chap. 11]. Secondly we already argued that per sample, the computation of $\tilde{\mathsf{b}}_{\mathsf{BP}}(w)$ requires more time than the one for $\tilde{\mathsf{b}}_{\mathrm{A1}}(w)$. The difference is even larger if Algorithm 1 uses bidirectional search [20, 34].

We can conclude this discussion stating that Algorithm 1, BP, and the algorithm by Geisberger et al. [16] share the same design principles, but choose different trade-offs between accuracy and speed.

6 Variants of betweenness centrality

In this section we discuss how to extends our results to some variants of betweenness centrality.

6.1 k-bounded-distance betweenness

A "local" variant of betweenness, called *k-bounded-distance betweenness*[4] only considers the contribution of shortest paths of size up to $k+1$ [8, 10]. For $k > 1$ and any pair of distinct vertices $u, v \in V$, $u \neq V$, let $\mathcal{S}_{uv}^{(k)} \subseteq \mathcal{S}_{uv}$ be the set of shortest paths from u to v of size at most $k+1$, with $\sigma_{uv}^{(k)} = |\mathcal{S}_{uv}^{(k)}|$, and let $\mathbb{S}_G^{(k)}$ be the union of all the $\mathcal{S}_{uv}^{(k)}$. Let $\mathcal{T}_v^{(k)} \subseteq \mathcal{T}_v$ be the set of all shortest paths *of size up to* k that v is internal to, for each $v \in V$.

[4] Bounded-distance betweenness is also known as k-betweenness. We prefer the former denomination to avoid confusion with k-path betweenness from [21].

Definition 6. [8, 10] Given a graph $G = (V, E)$ and an integer $k > 1$, the *k-bounded-distance betweenness centrality of a vertex $v \in V$* is defined as

$$\mathsf{bb}^{(k)}(v) = \frac{1}{n(n-1)} \sum_{p_{uw} \in \mathbb{S}_G^{(k)}} \frac{\mathbb{1}_{\mathcal{T}_v^{(k)}}(p)}{\sigma_{uw}^{(k)}} \enspace .$$

For the case of k-bounded-distance betweenness, if we let $\mathcal{R}_G^{(k)} = \{\mathcal{T}_v^{(k)} \ : \ v \in V\}$, it is easy to bound $\mathsf{VC}(\mathcal{R}_G^{(k)})$ following the same reasoning as in Lemma 1.

Lemma 11. $\mathsf{VC}(\mathcal{R}_G^{(k)}) \le \lfloor \log_2(k-1) \rfloor + 1$.

Given this result, the sample size on line 6 of Alg. 1 can be reduced to

$$r = \frac{c}{\varepsilon^2} \left(\lfloor \log_2(k-1) \rfloor + 1 + \ln \frac{1}{\delta} \right)$$

and the computation of the shortest paths on line 8 can be stopped after we reached the vertices that are k "hops" far from u.

6.2 α-weighted betweenness

Opsahl et al. [32] defined a parametric variant of betweenness centrality for weighted networks that can be seen as a generalization of the classical definition. The goal behind this new definition is to give the analyst the possibility of "penalizing" shortest paths with many vertices, given the intuition that there is a cost to be paid to have a message go through a vertex. The core of the new definition is a different weight function of a path $p_{uv} = (w_1 = u, w_2, \ldots, w_{|p_{uv}|} = v)$ between two vertices (u, v), parametrized by a real parameter $\alpha \ge 0$:

$$d_{\alpha,uv} = \sum_{i=1}^{|p_{uv}|-1} (\mathsf{w}((w_i, w_{i+1})))^\alpha \enspace . \tag{10}$$

Clearly for $\alpha = 0$, the weights are considered all equal to 1, and for $\alpha = 1$, the definition falls back to d_{uv}. For $0 < \alpha < 1$, paths with fewer vertices and higher weights are favored over longer paths with lighter edges[32]. The definition of shortest path distance, shortest path, and of betweenness centrality follow as before from (10). Dijkstra's shortest paths algorithm can be adapted to compute shortest paths according to the distance function in (10). To avoid confusion, we call the betweenness centrality computed using the shortest paths according to the distance function in (10), the *α-weighted betweenness* and denote it with $\mathsf{b}_\alpha(w)$.

For α-weighted betweenness we have that the VC-dimension of the range set defined on the shortest paths according to the distance from (10) is still bounded by $\lfloor \log_2(\mathsf{VD}(G) - 2) \rfloor + 1$. Clearly, the vertex-diameter $\mathsf{VD}(G)$ is now defined as the maximum number of vertices in a shortest path according to (10).

No special arrangements are needed to obtain an approximation of α-weighted betweenness for all vertices. The sample size is the same as in (3).

6.3 k-path betweenness

Another interesting variant of betweenness centrality does not consider *shortest* paths, but rather *simple random walks* of size up to $k+1$, expressing the intuition that information in a network does not necessarily spread across shortest paths but has a high probability of "fading out" after having touched at most a fixed number of vertices.

Definition 7 ([21])**.** Given a graph $G = (V, E)$ and a positive integer k, the *k-path centrality* $\mathsf{pb}^{(k)}(v)$ of $v \in V$ is defined as the average[5], over all possible source vertices s, of the probability that a simple random

[5] We take the average, rather than the sum, as defined in the original work by Kourtellis et al. [21], to normalize the value so that it belongs to the interval $[0, 1]$. This has no consequences on the results.

walk originating from s and extinguishing after having touched (at most) $k+1$ vertices (s included) goes through v.

We can define another range set $\mathcal{R}_G^{\mathrm{p},k}$ if we are interested in k-path betweenness. The domain B of the range set now is the set of all simple random walks starting from any vertex of and of size up to $k+1$. For each vertex $v \in V$, the range R_v is the subset of B containing all and only the random walks from B that have v as internal vertex. It is easy to see that $\mathsf{VC}(\mathcal{R}_G^{\mathrm{p},k})$ is at most $\lfloor \log_2(k-1)+1$ following the same reasoning as in Lemma 1 and Lemma 11.

For the case of k-path betweenness, the algorithm is slightly different: for

$$r = \frac{c}{\varepsilon^2}\left(\lfloor \log_2(k-1) \rfloor + 1 + \ln\frac{1}{\delta}\right)$$

iterations, rather than sampling a pair of vertices (uv), computing $\mathcal{S}_{uv}$, and then sampling a shortest path between them, we first sample a single vertex and an integer ℓ uniformly at random from $[1, k+1]$. We then sample a simple random walk touching ℓ vertices, while updating the estimated betweenness of each touched vertex by adding $1/r$ to its estimation. The method to sample a simple random walk is described in [21].

Kourtellis et al. [21] show that this algorithm can estimate, with probability at least $1-1/n^2$, all the k-path betweenness values to within an additive error $n^{-1/2+\alpha}/(n-1)$ using $2n^{1-2\alpha}k^2 \ln n$ samples, for $\alpha \in [-1/2, 1/2]$. We can obtain the same guarantees with a *much lower* number r of samples, precisely

$$r = 2n^{1-2\alpha}\left(\ln n + \frac{\lfloor \log_2(k-1) \rfloor + 1}{2}\right) \ .$$

In conclusion, we presented a tighter analysis of the algorithm by Kourtellis et al. [21].

6.4 Edge betweenness

Until now we focused on computing the betweenness centrality of vertices. It is also possible to define a betweenness centrality index for *edges* of a graph $G = (V, E)$ [3, 10]. Edge betweenness is useful, for example, to develop heuristics for community detection [31]. Given $e \in E$, the edge betweenness $\mathsf{eb}(e)$ of e is defined as the fraction of shortest paths that *contain* e, meaning that if $e = (u, v)$, then a path p contains e if u and v appear consecutively in p. Formally,

$$\mathsf{eb}(e) = \frac{1}{n(n-1)} \sum_{p_{uv} \in \mathbb{S}_G} \frac{\mathbb{1}_{p_{uv}}(e)}{\sigma_{uv}}, \forall e \in E \ .$$

We can then define a range space $\mathcal{ER}_G$ that contains $|E|$ ranges R_e, one for each $e \in E$, where R_e is the set of shortest paths containing the edge e. By following the same reasoning as Lemma 1 we have that $\mathsf{VC}(\mathcal{ER}_G) \le \lfloor (\log_2(\mathsf{VD}(G)-1) \rfloor + 1$. Using this result we can adapt the sample sizes for Alg. 1 and 2 to compute good approximations of betweenness for the (top-k) edges.

7 Experimental evaluation

We conducted an experimental evaluation of our algorithms, with two major driving goals in mind: study the behavior of the algorithms presented in this paper and compare it with that of other related algorithms [9, 11, 16, 19], in terms of accuracy of the estimation, execution time, work performed, and scalability as function of the network size.

Implementation and environment We implemented our algorithms, the one presented in [11, 19] and the linear scaling version from [16] in C, by extending the implementation of the exact algorithm [9] contained in igraph [12][6]. The implementations are similarly engineered, given that they are based on the same

[6] The implementations are available at `http://cs.brown.edu/~matteo/centrsampl.tar.bz2`.

				$\frac{\text{Time}_{\mathsf{BP}}}{\text{Time}_{\mathsf{VC}}}$	
	Graph Properties			diam-2approx	
Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\mathsf{VD}(G)$	min	max
oregon1-010331	10,670	22,002	9	4.39	4.75
oregon1-010526	11,174	23,409	10	4.26	4.73
ca-HepPh	12,008	237,010	13	3.06	3.33
ca-AstroPh	18,772	396,160	14	3.26	3.76
ca-CondMat	23,133	186,936	15	3.75	4.08
email-Enron	36,692	421,578	12	3.60	4.16

(a) Undirected graphs

				$\frac{\text{Time}_{\mathsf{BP}}}{\text{Time}_{\mathsf{VC}}}$			
	Graph Properties			diam-exact		diam-UB	
Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\mathsf{VD}(G)$	min	max	min	max
wiki-Vote	7,115	103,689	7	3.35	3.69	1.05	1.27
p2p-Gnutella25	22,687	54,705	11	5.45	5.78	1.94	2.09
cit-HepTh	27,770	352,807	14	3.58	3.83	1.39	1.61
cit-HepPh	34,546	421,578	12	4.91	5.01	1.60	1.71
p2p-Gnutella30	36,682	88,328	10	5.02	5.46	2.08	2.22
soc-Epinions1	75,879	508,837	13	4.20	4.25	1.35	1.38

(b) Directed graphs

Figure 7: Graph properties and running time ratios.

subroutines for the computation of the shortest path (Dijkstra's algorithm for weighted graphs, BFS for unweighted ones), and they received similar amounts of optimization. We exposed our implementations through Python 3.3.1, which was used for running the simulations. We run the experiments on a quad-core AMD Phenom™II X4 955 Processor with 16GB of RAM, running Debian *wheezy* with a Linux kernel version 3.2.0.

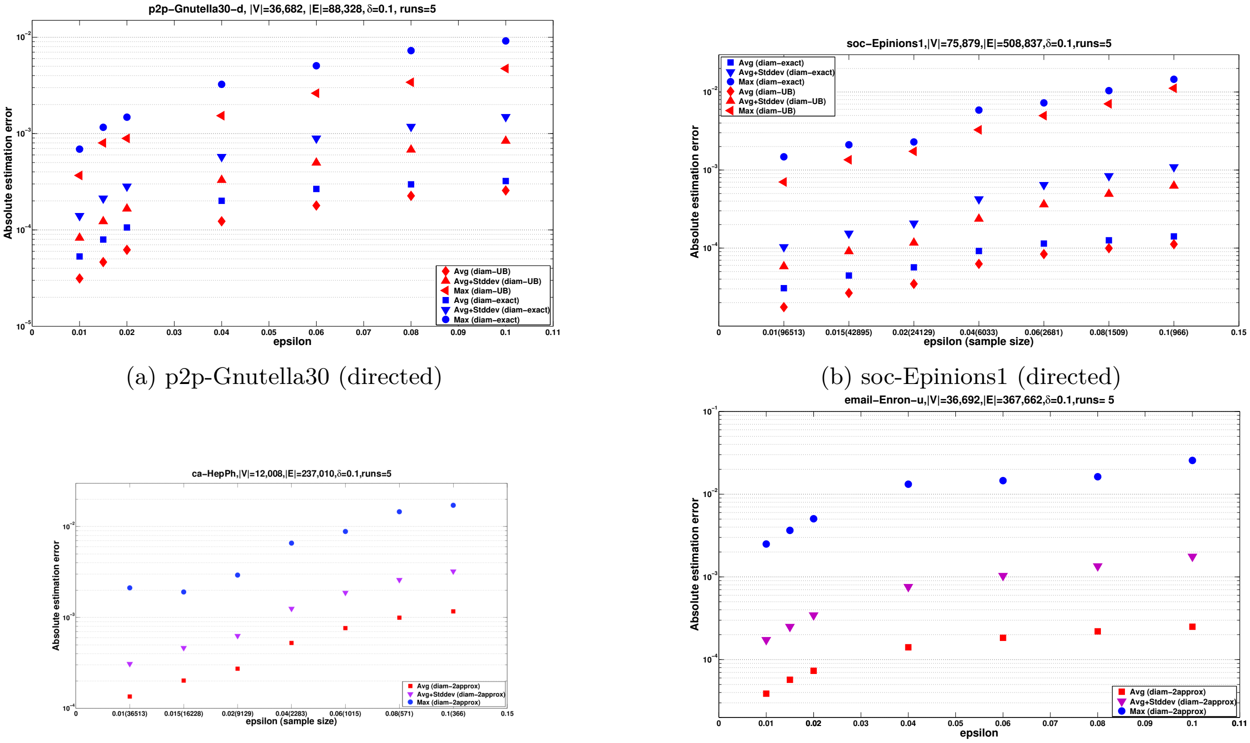
Datasets In our evaluation we used a number of graphs from the Stanford Large Network Dataset Collection[7]. These are all real world datasets including online social networks, communication (email) networks, scientific citation and academic collaboration networks, road networks, Amazon frequent co-purchased product networks, and more. Basic information about the graphs we used are reported in the two leftmost columns of Figs. 7b and 7a. We refer the reader to the SLNDC website for additional details about each dataset. To evaluate scalability we also created a number of artificial graphs of different sizes (1,000 to 100,000 vertices) using the Barabási-Albert model [6] as implemented by igraph [12].

Diameter approximation As we discussed in the previous sections the number of samples that the proposed algorithm requires depends on the vertex-diameter of the graph. For the computation of the vertex-diameter in case of undirected graphs we used the 2-approximation algorithm that we briefly described in Sect. 5. We denote this as "diam-2-approx" when reporting results in this section. For directed graphs, we computed the number of samples using both the exact value of the vertex-diameter (indicated as diam-exact) as well as the trivial upper bound $|V|-2$ (indicated as diam-UB).

[7] `http://snap.stanford.edu/data/index.html`

7.1 Accuracy

Our theoretical results from Sect. 5 guarantee that, with probability at least $1-\delta$, all estimations of the betweenness values for all vertices in the graph are within ε for their real value. We run Algorithm 1 five times for each graph and each value of ε in $\{0.01, 0.015, 0.02, 0.04, 0.06, 0.08, 0.1\}$. The parameter δ was fixed to 0.1 and we used $c = 0.5$ in (1) to compute the sample size, as suggested by Löffler and Phillips [25]. As far as the *confidence* is concerned, we report that in all the hundreds of runs we performed, the guarantee on the quality of approximation was *always* satisfied, not just with probability $1-\delta$ $(= 0.9)$. We evaluated how good the estimated values are by computing the average *estimation error* $(\sum_{v\in V} |\mathsf{b}(v) - \tilde{\mathsf{b}}(v)|)/|V|$ across five runs of our algorithm and taking the average and the standard deviation of this measure, for different values of ε. We also compute the maximum error $|\mathsf{b}(v) - \tilde{\mathsf{b}}(v)|$ overall. The results are reported in Fig. 8a for the directed graph p2p-Gnutella30, in Fig. 8c for the directed graph ca-HepPh, in Fig. 8b for the undirected graph soc-Epinions1, and in Fig. 8d for the undirected graph email-Enron. It is evident that the maximum error is an error of magnitude smaller than the guaranteed value of ε and that the average error is almost two orders of magnitude smaller than the guarantees, and the `Avg+Stddev` points show that the estimation are quite concentrated around the average. We can conclude that in practice the algorithm performs even *better than guaranteed*, achieving higher accuracy and confidence than what the theoretical analysis indicates. This is due to a number of factors, like the fact that we use an *upper bound* to the VC-dimension of the range set.



(a) p2p-Gnutella30 (directed)

(b) soc-Epinions1 (directed)

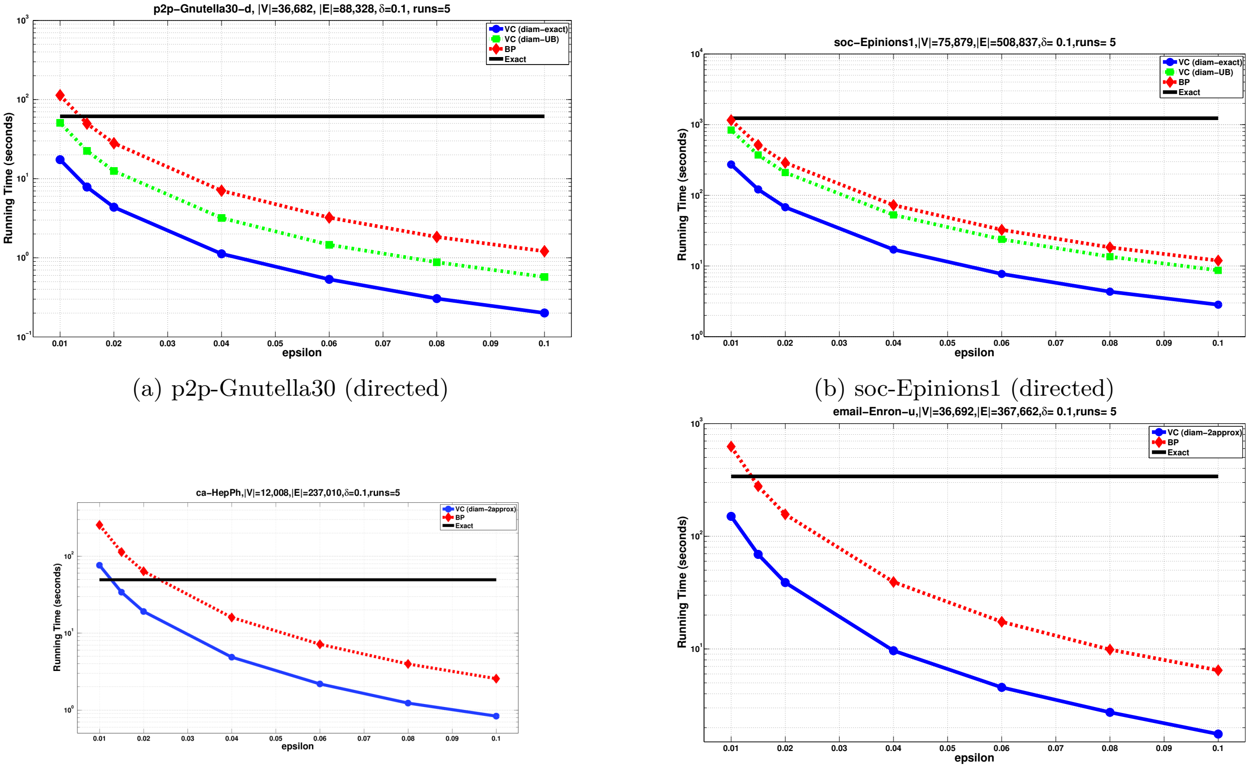
(c) ca-HepPh (directed)

(d) email-Enron (undirected)

Figure 8: Betweenness estimation error $|\tilde{\mathsf{b}}(v) - \mathsf{b}(v)|$ evaluation for directed and undirected graphs

7.2 Runtime

We compared the running time of Algorithm 1 (denoted in the following as VC to that of the algorithm from [11, 16, 19] (denoted as BP), and to that of the exact algorithm [9]. As VC and BP give the same guarantees on the accuracy and confidence of the computed estimations, it makes sense to compare their



(a) p2p-Gnutella30 (directed)

(b) soc-Epinions1 (directed)

(c) ca-HepPh (directed)

(d) email-Enron (undirected)

Figure 9: Running time (seconds) comparison between VC, BP, and the exact algorithm.

running times to evaluate which is faster in achieving the goal. The performances of the algorithm proposed in [16] takes the same time as BP, because it follows the same sampling approach and only differs in the definiton of the estimator for the betweenness, so we do not report those. The algorithms VC and BP take parameters ε and δ and compute the sample size accordingly. We run each experiments five times for each value of ε, and measured the average running time across the runs. The results are presented in Figs. 7 and 9. In Fig. 7a we report the minimum and the maximum ratio of the running time of BP over VC, taken over the ratios obtained by running the algorithms with the different values of ε. As it can be seen from this table our algorithm performs significantly faster, more than 300%. Similar results are reported for directed graphs in Fig. 7b. The diam-UB and the diam-exact values can be seen as the two extremes for the performance of Algorithm 1 in terms of runtime. In the case of the diam-exact we have as few samples as possible (for VC) since we use the exact value of the vertex-diameter, whereas in the case of diam-UB we have as many samples as possibles because we use the worst case estimation for the vertex-diameter of the graph. From Fig. 7a we can see that the value for the vertex-diameter that we consider in the case of diam-UB ($|V| - 2$) is many orders of magnitudes greater than the actual value, which translates in a significant increase of the number of samples. But even in the case of this crude vertex-diameter approximation (diam-UB), the VC algorithm performs uniformly faster than BP. In the case where the exact value of the diameter was used, we can see that our algorithm computes an estimation of the betweenness that satisfies the desired accuracy and confidence guarantees *3 to 5 times faster* than BP. In Fig. 9a we study the directed graph p2p-Gnutella30 and we present the measurements of the average running time of the algorithms for different values of ε, using the exact algorithm from [9] as baseline. The VC algorithm requires significantly less time than the BP algorithm. The figure also shows that there are values of ε for which BP takes more time than the exact algorithm, because the resulting sample size is larger than the graph size. Given that VC uses fewer samples and does fewer operations per sample, it can be used with lower ε than BP, while still saving time compared to the exact computation. Figure 9d shows the average running time of the algorithms for the undirected graph email-Enron. The behavior is similar to that for the undirected case. Algorithm 1 is faster than BP for two reasons, both originating from from our use of results from the VC-dimension theory: 1) we use a significantly smaller amount of samples and 2) VC performs the same amount of computations *per sample* as BP only in the worst case. Indeed our algorithm needs only to find the shortest path between a sampled pair of vertices, whereas the algorithms from [11, 16] need to compute the shortest paths between a sampled source and all the other vertices. In our experimental evaluation we found out that the running time of the algorithms is directly proportional to the number of edges touched during the shortest path computation. The use of bidirectional A* search [20, 34] can help in lowering the number of touched edges for VC and therefore the runtime of our algorithm (BP would not benefit from this improvement). We plan to explore this in future work.

7.3 Scalability

In Sect. 5.3 we argued about the reasons why Algorithm 1 is more scalable than BP, while still offering the same approximation guarantees. To evaluate our argument in practice, we created a number of graphs of increasing size (1,000 to 100,000 vertices) using the Barabási-Albert [6] and run the algorithms on them, measuring their running time. We report the results in Fig. 10. The most-scalable algorithm would be completely independent from the size (number of vertices) of the graph, corresponding to a flat (horizontal) line in the plot. Therefore, the less steep the line, the more independent from the network size would be the corresponding algorithm. From the figure, we can appreciate that this is the case for VC, which is much more scalable and independent from the size of the sample than BP. This is very important, as today's networks are not only huge, but they also grow rapidly, and algorithms to mine them must scale well with graph size.

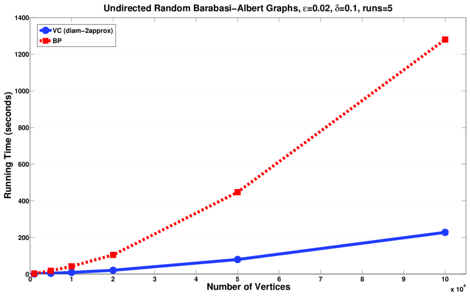


Figure 10: Scalability on random Barabási-Albert [6] graphs.

8 Conclusions

In this work we presented two random-sampling-based algorithms for accurately and efficiently estimate the betweenness centrality of the (top-K) vertices in a graph, with high probability. Our algorithms are based on a novel application of VC-dimension theory, and therefore take a different approach than previous ones achieving the same guarantees [11, 16, 19]. The number of samples needed to approximate the betweenness with the desired accuracy and confidence does not depend on the number of vertices in the graph, but rather on a characteristic quantity of the network that we call *vertex-diameter*. In some cases, the sample size is completely independent from any property of the graph. Our methods can be applied to many variants of betweenness, including edge betweenness. Our algorithms perform much less work than previously presented methods. As a consequence, they are much faster and scalable, as verified in the extensive experimental evaluation using many real and artificial graphs. In future work we would like to explore the possibility of using bidirectional A* search [20, 34] to further speed up our algorithms.

Acknowledgements

This project was supported, in part, by the National Science Foundation under award IIS-1247581. We are thankful to Eli Upfal for his guidance and to the anonymous reviewers whose comments helped us improving this work.

References

[1] I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, and R. F. Werneck. VC-dimension and shortest path algorithms. In *Automata, Languages and Programming*, volume 6755 of *Lecture Notes in Computer Science*, pages 690–699. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-22006-7_58.

[2] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999. doi: 10.1137/S0097539796303421.

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

[4] M. Anthony and P. L. Bartlett. *Neural Network Learning - Theoretical Foundations.* Cambridge University Press, New York, NY, USA, 1999. ISBN 978-0-521-57353-5.

[5] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail. Approximating betweenness centrality. In A. Bonato and F. Chung, editors, *Algorithms and Models for the Web-Graph*, volume 4863 of *Lecture Notes in Computer Science*, pages 124–137. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77003-9. doi: 10.1007/978-3-540-77004-6_10.

[6] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[7] K. Boitmanis, K. Freivalds, P. Lediņš, and R. Opmanis. Fast and simple approximation of the diameter and radius of a graph. In C. Alvarez and M. Serna, editors, *Experimental Algorithms*, volume 4007 of *Lecture Notes in Computer Science*, pages 98–108. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-34597-8. doi: 10.1007/11764298_9.

[8] S. P. Borgatti and M. G. Everett. A graph-theoretic perspective on centrality. *Soc. Netw.*, 28(4): 466–484, 2006. ISSN 0378-8733. doi: 10.1016/j.socnet.2005.11.005.

[9] U. Brandes. A faster algorithm for betweenness centrality. *J. Math. Sociol.*, 25(2):163–177, 2001. doi: 10.1080/0022250X.2001.9990249.

[10] U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Soc. Netw.*, 30(2):136–145, 2008. ISSN 0378-8733. doi: 10.1016/j.socnet.2007.11.001.

[11] U. Brandes and C. Pich. Centrality estimation in large networks. *Int. J. Bifurcation and Chaos*, 17(7): 2303–2318, 2007. doi: 10.1142/S0218127407018403.

[12] G. Csárdi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL `http://igraph.sf.net`.

[13] S. Dolev, Y. Elovici, and R. Puzis. Routing betweenness centrality. *J. ACM*, 57(4):25:1–25:27, May 2010. ISSN 0004-5411. doi: 10.1145/1734213.1734219.

[14] D. Eppstein and J. Wang. Fast approximation of centrality. *J. Graph Algorithms Appl.*, 8(1):39–45, 2004.

[15] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[16] R. Geisberger, P. Sanders, and D. Schultes. Better approximation of betweenness centrality. In J. I. Munro and D. Wagner, editors, *Algorithm Eng. & Experiments (ALENEX'08)*, pages 90–100. SIAM, 2008.

[17] S. Har-Peled and M. Sharir. Relative (p, ε)-approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011. ISSN 0179-5376. doi: 10.1007/s00454-010-9248-1.

[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statistical Assoc.*, 58(301):13–30, 1963.

[19] R. Jacob, D. Koschützki, K. Lehmann, L. Peeters, and D. Tenfelde-Podehl. Algorithms for centrality indices. In U. Brandes and T. Erlebach, editors, *Network Analysis*, volume 3418 of *Lecture Notes in Computer Science*, pages 62–82. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-24979-5. doi: 10.1007/978-3-540-31955-9_4. URL `http://dx.doi.org/10.1007/978-3-540-31955-9_4`.

[20] H. Kaindl and G. Kainz. Bidirectional heuristic search reconsidered. *J. Artif. Intell. Res. (JAIR)*, 7: 283–317, 1997. doi: 10.1613/jair.460.

[21] N. Kourtellis, T. Alahakoon, R. Simha, A. Iamnitchi, and R. Tripathi. Identifying high betweenness centrality nodes in large social networks. *Soc. Netw. Anal. and Mining*, pages 1–16, 2012. ISSN 1869-5450. doi: 10.1007/s13278-012-0076-6.

[22] E. Kranakis, D. Krizanc, B. Ruf, J. Urrutia, and G. Woeginger. The VC-dimension of set systems defined by graphs. *Discrete Applied Mathematics*, 77(3):237–257, 1997. ISSN 0166-218X. doi: 10.1016/S0166-218X(96)00137-0.

[23] Y. Li, P. M. Long, and A. Srinivasan. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62(3):516–527, 2001. ISSN 0022-0000. doi: 10.1006/jcss.2000.1741.

[24] Y.-s. Lim, D. S. Menasche, B. Ribeiro, D. Towsley, and P. Basu. Online estimating the k central nodes of a network. In *IEEE Network Science Workshop*, NSW'11, pages 118–122, 2011. doi: 10.1109/NSW.2011.6004633.

[25] M. Löffler and J. M. Phillips. Shape fitting on point sets with probability distributions. In A. Fiat and P. Sanders, editors, *Algorithms - ESA 2009*, volume 5757 of *Lecture Notes in Computer Science*, pages 313–324. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-04128-0_29.

[26] A. S. Maiya and T. Y. Berger-Wolf. Online sampling of high centrality individuals in social networks. In M. Zaki, J. Yu, B. Ravindran, and V. Pudi, editors, *Advances in Knowl. Disc. Data Mining*, volume 6118 of *Lecture Notes in Computer Science*, pages 91–98. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13656-6. doi: 10.1007/978-3-642-13657-3_12.

[27] J. Matoušek. *Lectures on Discrete Geometry.* Springer-Verlag, Secaucus, NJ, USA, 2002. ISBN 0387953744.

[28] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005.

[29] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning.* The MIT Press, 2012.

[30] M. E. J. Newman. *Networks – An Introduction.* Oxford University Press, 2010.

[31] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb. 2004. doi: 10.1103/PhysRevE.69.026113.

[32] T. Opsahl, F. Agneessens, and J. Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Soc. Netw.*, 32(3):245–251, 2010. doi: 10.1016/j.socnet.2010.03.006.

[33] J. Pfeffer and K. M. Carley. k-centralities: local approximations of global measures based on shortest paths. In *Proc. 21st Int. Conf. Companion on World Wide Web*, WWW '12 Companion, pages 1043–1050, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1230-1. doi: 10.1145/2187980.2188239.

[34] I. Pohl. *Bidirectional Heuristic Search in Path Problems.* PhD thesis, Stanford University, 1969.

[35] M. Riondato and E. M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. In C. Castillo and D. Metzler, editors, *Proc. 7th ACM Conf. Web Search Data Mining*, WSDM'14. ACM, 2014.

[36] L. Roditty and V. V. Williams. Approximating the diameter of a graph. *CoRR abs/1207.3622*, July 2012. URL `http://arxiv.org/abs/1207.3622`.

[37] A. E. Sarıyüce, E. Saule, K. Kaya, and U. V. Çatalyürek. Shattering and compressing networks for betweenness centrality. In *SIAM Data Mining Conf.*, 2013.

[38] V. N. Vapnik and A. J. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971. doi: 10.1137/1116025.