Shortest Path Centrality and the APSP problem via VC-dimension and Rademacher Averages

Alane M. de Lima
Department of Computer Science, Federal University of Paraná, Brazil
http://www.inf.ufpr.br/amlima
amlima@inf.ufpr.br

Murilo V. G. da Silva
Department of Computer Science, Federal University of Paraná, Brazil
http://www.inf.ufpr.br/murilo
murilo@inf.ufpr.br

André L. Vignatti
Department of Computer Science, Federal University of Paraná, Brazil
http://www.inf.ufpr.br/vignatti
vignatti@inf.ufpr.br

Abstract

In this paper we are interested in a version of the All-pairs Shortest Paths problem (APSP) that fits neither in the exact nor in the approximate case. We define a measure of centrality of a shortest path, related to the "importance" of such shortest path in the graph, and propose an algorithm that, for *any fixed constants* $0 < \epsilon, \delta < 1$, given an undirected graph G with non-negative edge weights, outputs with probability $1 - \delta$ the (exact) distance and the shortest path between every pair of vertices (u, v) that has centrality at least ϵ in expected time $\mathcal{O}(\lg \mathrm{Diam}_V(G) \cdot \max(m + n \log n, \mathrm{Diam}_V(G)^2)$, where $\mathrm{Diam}_V(G)$ is the vertex diameter of G. The algorithm progressively samples shortest paths in G making use of Rademacher Averages for deciding the point at which it should stop. Additionally, when the sample size reaches an upper bound of size $\lceil \frac{c}{\epsilon} \lfloor 2 \lg \mathrm{Diam}_V(G) + 1 \rfloor \ln(\frac{1}{\epsilon}) + \ln \frac{1}{\delta} \rceil$ the algorithm stops unconditionally. We use VC-dimension theory to show that a sample of this size suffices. We note that this bound for the sample size is exponentially smaller than bounds obtained by other standard techniques when the input graph has logarithmic vertex diameter, which is a common case for real-world graphs.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Shortest paths; Theory of computation → Approximation algorithms analysis; Theory of computation → Sample complexity and generalization bounds

Keywords and phrases All-pairs Shortest Paths; Shortest Path Centrality; Sample Complexity

Funding *Alane M. de Lima*: Supported by CAPES and CNPq.

1 Introduction

The All-pairs Shortest Path (APSP) is the problem of computing a path with the minimum length between every pair of vertices in a weighted graph. The APSP problem is very well studied and there has been recent results for a variety of assumptions for the input graph (directed/undirected, integer/real edge weights, etc) [6, 7, 10, 24]. In this paper we assume that the input is an undirected graph G with n vertices and m edges with non-negative weights.

In our scenario, the fastest known exact algorithms are the algorithm proposed by Williams (2014) [24], which runs in $\mathcal{O}\left(\frac{n^3}{2^{c\sqrt{\log n}}}\right)$ time, for some constant $c > 0$, and by Pettie and Ramachandram (2002) [18] for the case of sparse graphs, which runs in $\mathcal{O}(mn \log \alpha(m, n))$ time, where $\alpha(m, n)$ is the Tarjan's inverse-Ackermann function. If no assumption is taken about the sparsity of the graph, it is an open question whether the APSP problem can be

solved in $\mathcal{O}(n^{3-c})$, for any $c > 0$ even when the edge weights are natural numbers. The hypothesis that there is no such algorithm for such task is used in hardness arguments in some works [1, 2].

The three fastest approximation algorithms for the problem depend on the approximation guarantees as well as the sparsity of the input graph. Elkin et al. (2019) [11] proposed an approximation that runs in $\mathcal{O}(n^2)$ time and has multiplicative factor $1 + \epsilon$ and an additive term $\beta(G)$, where $\beta(G)$ depends on the edge weights. Baswana and Kavitha (2010) [4] proposed two algorithms, one that runs in $\tilde{\mathcal{O}}(m\sqrt{n} + n^2)$ time and other that runs in $\tilde{\mathcal{O}}(m^{2/3}n + n^2)$ time, depending on the required approximation factor. There has also been recent development on approximation algorithms when assumptions for the input graph are not the same as ours [5, 25].

In this paper, we deal with a version of the problem that fits neither in the exact nor in the approximate case. For every pair of vertices u, v our algorithm either outputs a shortest path $P_{u,v}$ between u and v of exact size or it does not compute any shortest path between u and v, depending on a certain measure of the "importance" of the shortest path P, which we call the *centrality* of P. The intuition behind this measure is that the centrality of P is higher when a large number of shortest paths in a certain canonical set (in particular, this set contains a shortest path between every pair of vertex of G) has P as a subpath. The precise definition of this centrality measure is given in Section 2.1.

Let $\mathrm{Diam}_V(G)$ be the vertex-diameter of the input graph (i.e., the maximum number of vertices in a shortest path in G). In this paper we present a $\mathcal{O}(\lg \mathrm{Diam}_V(G) \cdot \max(m + n \log n, \mathrm{Diam}_V(G)^2)$ expected running time randomized algorithm for computing the shortest path between every pair of vertices that has shortest path centrality higher than certain fixed constant. This is particularly interesting for sparse graphs with logarithmic vertex diameter (this is the case for many real word graphs). The central idea is to sample shortest paths starting at the roots of a shortest paths trees. The main contribution of this paper is a bound on the sample size that is exponentially smaller in graphs with logarithmic vertex diameter than bounds given by standard Hoeffding and union-bound techniques.

A second contribution of the paper is an analysis relying on the use of Rademacher Averages in a progressive sampling approach to build an algorithm that iteratively increases the sample size until the desired accuracy is achieved, or the sample size reached a certain maximum size. The number of steps in the progressive sampling technique is sensitive to the probability distribution of the input graph (if we assume that the input is sampled according to a certain distribution). However, even if we make no assumption on the input graph, we use the Vapnik-Chervonenkis (VC) theory to give an upper bound for the maximum sample size that scales with $\lg(\mathrm{Diam}_V(G))$ (note that a bound given by Hoeffding and union-bound scales with $\lg(n)$). This upper bound is tighter since VC-dimension theory captures the combinatorial structure of the input graph and this bound for such graph can be computed efficiently. More precisely, we show that sampling $\lceil \frac{c}{\epsilon}(\lfloor 2 \lg \mathrm{Diam}_V(G) + 1 \rfloor \ln(\frac{1}{\epsilon}) + \ln \frac{1}{\delta}) \rceil$ shortest paths (and inspecting its subpaths) are enough for finding with probability $1 - \delta$, all shortest paths with centrality at least ϵ, where c is a constant around $\frac{1}{2}$.

Some of the techniques used in this paper were developed by Riondato and Kornaropoulos (2016) and Riondato and Upfal (2018) [20, 21]. In their work, the authors use VC-dimension theory, the ϵ-sample theorem and Rademacher averages for the estimation of betweenness centrality in a graph. More recently Lima et al. (2020) [8, 9] used some of these tools for the estimation of the percolation centrality using pseudo-dimension theory and Rademacher averages. In fact, we show two different algorithms in this paper. The first algorithm outputs with probability $1 - \delta$ an estimation for the centrality $c(u, v)$ within ϵ of the optimal

value, for any fixed constants $0 < \epsilon, \delta < 1$. The second algorithm outputs with probability $1 - \delta$ a shortest path between u and v if $c(u, v)$ is at least ϵ. Both algorithms run in $\mathcal{O}(\lg \mathrm{Diam}_V(G) \cdot \max(m + n \log n, \mathrm{Diam}_V(G)^2)$ expected running time.

2 Preliminaries

The definitions, notation and results which are the theoretical foundation of our work are presented below.

2.1 Shortest Paths in Graphs

Let $G = (V, E)$ be an undirected graph and a function $w : E \to \mathbb{R}^+$, where $w(e)$ is the non-negative *weight* of the edge e. W.l.o.g. we assume that G is connected, since all results in this paper can be applied to the connected components of a graph, when a graph is disconnected. A *path* is a sequence of vertices $p = (v_1, v_2, \ldots, v_k)$ such that, for $1 \leq i < k$, $v_i \neq v_{i+1}$ and there is $(v_i, v_{i+1}) \in E$. Let E_p be the set of edges of a path p. The length of p, denoted by $l(p)$, corresponds to the sum $\sum_{e \in E_P} w(e)$. For a pair $(u, v) \in V^2$, let P_{uv} be the set of all paths from u to v. A *shortest path* is a path $p_{uv} \in P_{uv}$ where $l(p_{uv}) = \min\{l(p_{u'v'}) : p_{u'v'} \in P_{uv}\}$. The length of a shortest path is called *distance*.

A *shortest paths tree (SPT)* of a vertex u is a spanning tree T_u of G such that the path from u to every other vertex of T_u is a shortest path in G. Note that there might be many SPTs for a given vertex. In this paper we are interested in fixing one canonical SPT for every vertex of G. More precisely, we fix an (arbitrary) ordering of the vertex set V and let the canonical SPT for a vertex u be the SPT output by Dijkstra's algorithm and denote such tree T_u. We also call T_u the *Dijkstra tree* of u. A shortest path that starts at the root of a Dijsktra tree is called a *branch* of G. More formally, given T_u, for every $v \neq u$, the shortest path from u to v is a branch, denoted by $\mathcal{B}_u(v)$. In addition, every subpath of $\mathcal{B}_u(v) = p_{uv}$ is also a shortest path in G, and we denote such set of subpaths (including p_{uv}) as $S(p_{uv})$ or $S(\mathcal{B}_u(v))$ (both notations are used interchangeably, as per convenience). Since G is undirected, the same applies for paths in reverse order, i.e., every subpath of p_{vu} in T_u is also a shortest path. Let $S(p_{vu})$ be such set of shortest paths.

Note that there are exactly n Dijkstra trees for G since Dijkstra's algorithm is deterministic and we have a fixed ordering for V. The set of n Dijkstra trees of G is denoted by $\mathcal{T}$. Let $S(T_u) = \bigcup_{v \in V \setminus \{u\}} (S(p_{uv}) \cup S(p_{vu}))$. The canonical set of shortest paths of G (w.r.t. the ordering) is $S(G) = \bigcup_{u \in V} S(T_u)$. For the sake of convenience in Definition 1 we present the length of a shortest path (distance) between a pair $(u, v) \in V^2$ in terms of Dijkstra trees.

▶ **Definition 1** (Distance). *Given a graph $G = (V, E)$, a function $w : E \to \mathbb{R}^+$, a pair $(u, v) \in V^2$, a vertex x and a Dijkstra tree T_x, the* distance *from u to v is defined as*

$$d(p_{uv}) = \sum_{e \in E_{T_x}} w(e)$$

where $p_{uv} \in S(T_x)$ and E_{T_x} is the set of edges of p_{uv} in T_x.

We define below the *shortest path centrality* of a pair of vertices (u, v) as the proportion of branches that contains p_{uv} as subpath.

▶ **Definition 2** (Shortest Path Centrality). *Given an undirected weighted graph $G = (V, E)$ with $n = |V|$, a pair $(u, v) \in V^2$ and the Dijkstra tree T_a for each $a \in V$, let $p_{ab} = (a, \ldots, b)$*

and $p_{uv} = (u, \ldots, v)$ *be shortest paths from* a *to* b *and* u *to* v*, respectively, such that* $p_{ab}, p_{uv} \in S(G)$*. The* shortest path centrality *of a pair* $(u, v) \in V^2$ *is defined as*

$$c(u, v) = \frac{t_{uv}}{n(n-1)}$$

where $t_{uv} = \sum_{a,b \in V^2 : a \neq b} \mathbb{1}_{\tau_{uv}}(\mathcal{B}_a(b))$ *and* $\tau_{uv} = \{\mathcal{B}_c(d) \in \bigcup_{a \in V} \bigcup_{b \in V : b \neq a} \mathcal{B}_a(b) : p_{uv} \in S(\mathcal{B}_c(d))\}$.

The function $\mathbb{1}_{\tau_{uv}}(\mathcal{B}_a(b))$ returns 1 if there is some shortest path from u to v as subpath of the branch $\mathcal{B}_a(b) \in S(T_a)$ (and 0 otherwise). Clearly the centrality measure depends on the fixed ordering, however, since we are dealing random sampling note that this ordering is not relevant.

2.2 Sample Complexity and VC-dimension

In sampling algorithms, typically the aim is the estimation of a certain quantity according to given parameters of quality and confidence using a random sample of size as small as possible. A central concept in sample complexity theory is the Vapnik-Chervonenkis Theory (VC-Dimension), in particular, the idea of finding an upper bound for the VC-dimension of a class of binary functions related to the sampling problem at hand. In our context, for instance, we may consider a binary function that takes a branch and outputs 1 if such branch contains a shortest path for a given set. Generally speaking, from the upper bound for the given class of binary functions we can derive an upper bound to the sample size for the sampling algorithm.

We present in this section the main definitions and results from sample complexity theory used in this paper. An in-depth exposition of the Vapnik-Chervonenkis Theory (VC-Dimension), the ϵ-sample and the ϵ-net theorems can be found in the books of Shalev-Schwartz and Ben-David (2014) [23], Mitzenmacher and Upfal (2017) [15], Anthony and Bartlett (2009) [3], and Mohri et al. (2012) [16].

▶ **Definition 3** (Range Space)**.** *A* range space *is a pair* $\mathcal{R} = (U, \mathcal{I})$*, where* U *is a domain (finite or infinite) and* $\mathcal{I}$ *is a collection of subsets of* U*, called* ranges*.*

For a given $S \subseteq U$, the *projection* of $\mathcal{I}$ on S is the set $\mathcal{I}_S = \{S \cap I : I \in \mathcal{I}\}$. If $|\mathcal{I}_S| = 2^{|S|}$ then we say S is *shattered* by $\mathcal{I}$. The VC-dimension of a range space is the size of the largest subset S that can be shattered by $\mathcal{I}$, i.e.,

▶ **Definition 4** (VC-dimension)**.** *The VC-dimension of a range space* $\mathcal{R} = (U, \mathcal{I})$*, denoted by* $VCDim(\mathcal{R})$*, is* $VCDim(\mathcal{R}) = \max\{k : \exists S \subseteq U \text{ such that } |S| = k \text{ and } |\mathcal{I}_S| = 2^k\}$.

The following combinatorial object, called ϵ-net, is useful when one wants to find a sample $S \subseteq U$ that intersects every range in $\mathcal{I}$ of a sufficient size.

▶ **Definition 5** (ϵ-net)**.** *Given* $0 < \epsilon < 1$*, a set* S *is called* ϵ*-net w.r.t. a range space* $\mathcal{R} = (U, \mathcal{I})$ *and a probability distribution* π *on* U *if*

$$\forall I \in \mathcal{I}, \quad \Pr_{\pi}(I) \geq \epsilon \Rightarrow |I \cap S| \geq 1.$$

The definition of ϵ-sample is a stronger notion as it not only intersects ranges of a sufficient size but it also guarantees the right relative frequency of each range in $\mathcal{I}$ within the sample S.

▶ **Definition 6** (ϵ-sample). *Given $0 < \epsilon < 1$, a set S is called ϵ-sample w.r.t. a range space $\mathcal{R} = (U, \mathcal{I})$ and a probability distribution π on U if*

$$\forall I \in \mathcal{I}, \quad \left| \Pr_{\pi}(I) - \frac{|S \cap I|}{|S|} \right| \leq \epsilon.$$

A more general definition of ϵ-sample (called ϵ-representative) is given below for the context where for a given a domain U and a set of values of interest $\mathcal{H}$, there is a family of functions $\mathcal{F}$ from U to $\mathbb{R}^*$ such that there is one $f_h \in \mathcal{F}$ for each $h \in \mathcal{H}$. Let S be a collection of r elements from U sampled with respect to a probability distribution π.

▶ **Definition 7.** *For each $f_h \in \mathcal{F}$, such that $h \in \mathcal{H}$, we define the expectation of f_h and its empirical average as L_U and L_S, respectively, i.e.,*

$$L_U(f_h) = \mathbb{E}_{u \in U}[f_h(u)] \quad \textit{and} \quad L_S(f_h) = \frac{1}{r} \sum_{s \in S} f_h(s).$$

▶ **Definition 8.** *Given $0 < \epsilon, \delta < 1$, a set S is called* ϵ-representative *w.r.t. some domain U, a set $\mathcal{H}$, a family of functions $\mathcal{F}$ and a probability distribution π if $|L_S(f_h) - L_U(f_h)| \leq \epsilon$, $\forall f_h \in \mathcal{F}$.*

The expectation of the empirical average $L_S(f_h)$ corresponds to $L_U(f_h)$, by linearity of expectation, therefore $|L_S(f_h) - L_U(f_h)| = |L_S(f_h) - \mathbb{E}_{f_h \in \mathcal{F}}[L_S(f_h)]|$. By the *law of large numbers*, $L_S(f_h)$ converges to its true expectation as r goes to infinity, since $L_S(f_h)$ is the empirical average of r random variables sampled independently and identically w.r.t. π. Since this law provides no information about the value $|L_S(f_h) - L_U(f_h)|$ for any sample size, we use results from VC-dimension theory, which provide bounds on the size of the sample that guarantees that the maximum deviation of $|L_S(f_h) - L_U(f_h)|$ is within ϵ with probability at least $1 - \delta$, for given $0 < \epsilon, \delta < 1$.

An upper bound to the VC-dimension of a range space allows to build an ϵ-net and an ϵ-representative sample, as stated in Theorem 9.

▶ **Theorem 9** (see [12], Theorem 2.12). *Given $0 < \epsilon, \delta < 1$, let $\mathcal{R} = (U, \mathcal{I})$ be a range space with $VCDim(\mathcal{R}) \leq k$, a probability distribution π on the domain U and let c be a universal positive constant.*

1. *A collection of elements $S \subseteq U$ sampled w.r.t. π with $|S| = \frac{c}{\epsilon^2}\left(k + \ln \frac{1}{\delta}\right)$ is ϵ-representative with probability at least $1 - \delta$.*
2. *A collection of elements $S \subseteq U$ sampled w.r.t. π with $|S| = \frac{c}{\epsilon}\left(k \ln \frac{1}{\epsilon} + \ln \frac{1}{\delta}\right)$ is an ϵ-net with probability at least $1 - \delta$.*

As pointed by Löffler and Phillips (2009) [14], c is around $\frac{1}{2}$, but in this paper we leave c as an unspecified constant.

2.3 Progressive Sampling and Rademacher Averages

Finding a bound to the sample size that is tight may be a complicated task depending on the problem. Hence, making use of progressive sampling, in which the process starts with a small sample size which progressively increases until the accuracy improves, becomes an alternative to this issue [19]. The combination of an appropriate scheduling for the sample increase with an efficient-to-evaluate stopping condition (i.e., knowing when the sample is large enough) leads to a greater improvement in time for the estimation of the value of interest [21]. A key idea is that the stopping condition takes into consideration the input distribution, which can

be extracted by the use of Rademacher Averages ([15], chapter 14). This theory lies in the core of statistical learning theory, although their applications extend beyond the context of learning algorithms [22].

Consider the computation of the maximum deviation of $L_S(f_h)$ from the true expectation of f_h, for all $f_h \in \mathcal{F}$, that is, $\sup_{f_h \in \mathcal{F}} |L_S(f_h) - L_U(f_h)|$. The *empirical Rademacher average* of $\mathcal{F}$ is defined as follows.

▶ **Definition 10.** *Consider a sample $S = \{z_1, \ldots, z_r\}$ and a distribution of r Rademacher random variables $\sigma = (\sigma_1, \ldots, \sigma_r)$, i.e., $\Pr(\sigma_i = 1) = \Pr(\sigma_i = -1) = 1/2$ for $1 \leq i \leq r$. The empirical Rademacher average of a family of functions $\mathcal{F}$ w.r.t. to S is defined as*

$$\tilde{R}_r(\mathcal{F}, S) = \mathbb{E}_\sigma \left[\sup_{f_h \in \mathcal{F}} \frac{1}{r} \sum_{i=1}^{r} \sigma_i f_h(z_i) \right].$$

In this work, we use the bound previously introduced by Riondato and Upfal [21] for the connection of the empirical Rademacher average with the value of $\sup_{f_h \in \mathcal{F}} |L_S(f_h) - L_U(f_h)|$, which extended the bound of Oneto et al. [17] to the supremum of its absolute value to functions with codomain in $[0, 1]$.

▶ **Theorem 11.** *With probability at least $1 - \delta$,*

$$\sup_{f_h \in \mathcal{F}} |L_S(f_h) - L_U(f_h)| \leq 2\tilde{R}_r(\mathcal{F}, S) + \frac{\ln \frac{3}{\delta} + \sqrt{(\ln \frac{3}{\delta} + 4r\tilde{R}_r(\mathcal{F}, S)) \ln \frac{3}{\delta}}}{r} + \sqrt{\frac{\frac{3}{\delta}}{2r}}.$$

The exact computation of $\tilde{R}_r(\mathcal{F}, S)$ depends on an extreme value, i.e., the supremum of deviations for all functions in $\mathcal{F}$, which can be expensive over a large (or infinite) set of functions [15]. Even a Monte Carlo simulation to estimating $\tilde{R}_r(\mathcal{F}, S)$ is expensive to be extracted in this case; hence, we use the bound given by Theorem 12, which is a variant of the Massart's Lemma (see Theorem 14.22, [15]) that is convex, continuous in $\mathbb{R}^+$ and can be efficiently minimized by standard convex optimization methods.

Consider the vector $v_{f_h} = (f_h(z_1), \ldots, f_h(z_m))$ for a given sample of m elements, denoted by $S = \{z_1, \ldots, z_m\}$, and let $\mathcal{V}_S = \{v_{f_h}, f_h \in \mathcal{F}\}$.

▶ **Theorem 12.** *(Riondato and Upfal [21]) Let $w : \mathbb{R}^+ \to \mathbb{R}^+$ be the function*

$$w(s) = \frac{1}{s} \ln \sum_{v_{f_h} \in \mathcal{V}_S} \exp\left(\frac{s^2 ||v_{f_h}||_2^2}{2m^2} \right).$$

Then $\tilde{R}_r(\mathcal{F}, S) \leq \min_{s \in \mathbb{R}^+} w(s)$.

3 Estimation for the Shortest Path Centrality and the All-pairs Shortest Path Problem

We first define the problem in terms of a range space, and then we give an outline of the two algorithms that we present in this paper. Both algorithms take as input an undirected graph $G = (V, E)$ with n vertices and m edges with non-negative edge weights, a sample schedule $(|S_i|)_{i \geq 1}$ and the quality and confidence parameters $0 < \epsilon, \delta < 1$, assumed to be constants.

Let $n = |V|$ and $\mathcal{T}$ be the set of n Dijkstra trees of G. The set $\mathcal{H}$ from Section 2.2 is defined to be V^2 and the universe U is the set of all branches, i.e., $U = \bigcup_{a \in V} \bigcup_{b \in V : b \neq a} \mathcal{B}_a(b)$. For each pair $(u, v) \in V^2$, let p_{uv} be a shortest path from u to v. Let $\tau_{uv} = \{\mathcal{B}_a(b) \in U :$

$p_{uv} \in S(\mathcal{B}_a(b))\}$, and let $\mathcal{I} = \{\tau_{uv} : (u,v) \in V^2\}$. Note that $\mathcal{R} = (U, \mathcal{I})$ is a range space. For $\mathcal{B}_a(b) \in U$, let $f_{uv} : U \to \{0,1\}$ be the function $f_{uv}(\mathcal{B}_a(b)) = \mathbb{1}_{\tau_{uv}}(\mathcal{B}_a(b))$. The indicator function $\mathbb{1}_{\tau_{uv}}(\mathcal{B}_a(b))$ returns 1 if there is some shortest path from u to v as subpath of $\mathcal{B}_a(b)$ (and 0 otherwise). We define $\mathcal{F} = \{f_{uv} : (u,v) \in V^2\}$.

Each $T_a \in \mathcal{T}$ is sampled according to the function $\pi(T_a) = \frac{1}{n}$ and each $\mathcal{B}_a(b) \in T_a$ is sampled with probability $\frac{1}{n-1}$, leading to the function $\pi(\mathcal{B}_a(b)) = \frac{1}{n(n-1)}$ (which is a valid probability distribution), and $\mathbb{E}[f_{uv}(\mathcal{B}_a(b))] = c(u,v)$ for all $(u,v) \in V^2$, as proved in Theorem 13.

▶ **Theorem 13.** *For $f_{uv} \in \mathcal{F}$, for each $T_a \in \mathcal{T}$ and for each branch $\mathcal{B}_a(b) \in T_a$, such that each $\mathcal{B}_a(b)$ is sampled according to the probability function $\pi(\mathcal{B}_a(b))$, $\mathbb{E}[f_{uv}(\mathcal{B}_a(b))] = c(u,v)$.*

Proof. Given an undirected weighted graph $G = (V, E)$, for all $(u,v) \in V^2$ we have from Definition 7

$$\begin{aligned} L_U(f_{uv}) = L_{\mathcal{T}}(f_{uv}) = \mathbb{E}_{(a,b)\in V^2}[f_{uv}(\mathcal{B}_a(b))] &= \sum_{T_a \in \mathcal{T}} \sum_{\mathcal{B}_a(b) \in T_a} \pi(\mathcal{B}_a(b)) f_{uv}(\mathcal{B}_a(b)) \\ &= \frac{1}{n(n-1)} \sum_{T_a \in \mathcal{T}} \sum_{\mathcal{B}_a(b) \in T_a} \mathbb{1}_{\tau_{uv}}(\mathcal{B}_a(b)) \\ &= \frac{1}{n(n-1)} \sum_{a \in V} \sum_{b \in V : b \neq a} \mathbb{1}_{\tau_{uv}}(\mathcal{B}_a(b)) \\ &= \frac{t_{uv}}{n(n-1)} = c(u,v) \end{aligned}$$

◀

Let $S = \{\mathcal{B}_{a_i}(b_i), 1 \leq i \leq r\}$ be a set of i branches sampled independently and identically from $U = \cup_{a \in V} \cup_{b \in V : b \neq a} \mathcal{B}_a(b)$. Next, we define $\tilde{c}(u,v)$, the estimation to be computed by the algorithm, as the empirical average from Definition 7:

$$\tilde{c}(u,v) = L_S(f_{uv}) = \frac{1}{r} \sum_{\mathcal{B}_{a_i}(b_i) \in S} f_{uv}(\mathcal{B}_{a_i}(b_i)) = \frac{1}{r} \sum_{\mathcal{B}_{a_i}(b_i)) \in S} \mathbb{1}_{\tau_{uv}}(\mathcal{B}_{a_i}(b_i)).$$

For each $(u,v) \in V^2$, the value $\tilde{c}(u,v)$ can be defined as $||\mathbf{v}_{uv}||_1/r$, where

$$\mathbf{v}_{uv} = (f_{uv}(\mathcal{B}_{a_1}(b_1)), \ldots, f_{uv}(\mathcal{B}_{a_i}(b_i))).$$

Each function f_{uv}, however, is a binary function such that $||\mathbf{v}_{uv}|| = t_{uv}$. Hence, we denote $\mathcal{V}$ as the set of such values, i.e., $\mathcal{V} = \{t_{uv}, (u,v) \in V^2\}$. Note that $|\mathcal{V}| \leq (|V^2| - n)/2$, since G is undirected – and then for a pair (u,v), $t_{uv} = t_{vu}$ – and there may be different pairs of vertices (u_k, v_k) and (u_l, v_l) with $t_{u_k v_k} = t_{u_l v_l}$.

The VC-dimension of the range space $\mathcal{R} = (\mathcal{T}, \mathcal{I})$, which is an upper bound to the fixed sample size that guarantees that $|\tilde{c}(u,v) - c(u,v)| \leq \epsilon$ with probability at least $1 - \delta$, for each $(u,v) \in V^2$ and for $0 < \epsilon, \delta < 1$, is stated below. In remainder of this paper let $\mathrm{Diam}_V(G)$ be the vertex-diameter of G, i.e., the maximum number of vertices in a shortest path of G.

▶ **Theorem 14.** *The VC-dimension of the range space $\mathcal{R} = (U, \mathcal{I})$ is*

$$VCDim(\mathcal{R}) \leq \lfloor 2 \lg Diam_V(G) + 1 \rfloor.$$

Proof. Let $\text{VCDim}(\mathcal{R}) = k$, where $k \in \mathbb{N}$. Then, there is $S \subseteq U$ such that $|S| = k$ and S is shattered by $\mathcal{I}$. By the definition of shattering, each $\mathcal{B}_{a_i}(b_i)) \in S$ must appear in 2^{k-1} different ranges in $\mathcal{I}$. On the other hand, $\mathcal{B}_{a_i}(b_i)$ has length at most $\text{Diam}_V(G)$. So, considering that $\mathcal{B}_{a_i}(b_i) = \text{Diam}_V(G)$, then $|S(\mathcal{B}_{a_i}(b_i))| = \text{Diam}_V(G) \cdot (\text{Diam}_V(G) - 1)$. Hence, $2^{k-1} \leq \text{Diam}_V(G) \cdot (\text{Diam}_V(G) - 1)$, and $k - 1 \leq \lg(\text{Diam}_V(G) \cdot (\text{Diam}_V(G) - 1)) \leq 2 \cdot \text{Diam}_V(G)$. Since k must be integer, $k \leq \lfloor 2 \lg \text{Diam}_V(G) + 1 \rfloor \leq 2 \lg \text{Diam}_V(G) + 1$. Finally, $\text{VCDim}(\mathcal{R}) = k \leq \lfloor 2 \lg \text{Diam}_V(G) + 1 \rfloor$. ◀

Note that for a sample of size r, by Hoeffding bound we have

$$\Pr(|\tilde{c}(u, v) - c(u, v)| \geq \epsilon) \leq 2exp(-2r\epsilon^2).$$

Applying the union bound for all $(u, v) \in V^2$, the value of r must be $2exp(-2r^2)n^2 \geq \delta$, which leads to $r \geq \frac{1}{2\epsilon^2}(\ln 2 + 2 \ln n + \ln(1/\delta))$. Even though $\text{Diam}_V(G)$ might be as large as n, we note that our bound given in Theorem 14 is tighter since it depends on the combinatorial structure of G. In particular, since a bound on $\text{Diam}_V(G)$ can be computed efficiently, in our algorithm we compute a sample size tailored for the input graph in question. In particular, our bound is exponentially smaller for graphs with logarithmic vertex diameter, which are common in practice.

The results above can be improved if we consider a progressive sampling approach instead of running the sampling algorithm directly in a sample of fixed size. We define the progressive sampling schedule of this work as follows: let S_1 be the initial sample size and $\delta_1 = \delta/2$. At this point, the only information available about the empirical Rademacher complexity of S_1 is that $\tilde{R}_r(\mathcal{F}, S_1) \geq 0$. Plugging this with the r.h.s. of the bound in Theorem 11, which has to be at most ϵ, we have

$$\frac{\ln(3/(\delta/2)) + \sqrt{\ln(3/(\delta/2))\ln(3/(\delta/2))}}{|S_1|} + \sqrt{\frac{\ln(3/(\delta/2))}{2|S_1|}} \leq \epsilon$$

$$\frac{2\ln(6/\delta)}{|S_1|} + \sqrt{\frac{\ln(6/\delta)}{2|S_1|}} \leq \epsilon$$

$$\frac{4\ln^2(6/\delta)}{|S_1|^2} + \frac{\ln(6/\delta)}{2|S_1|} \leq \epsilon^2$$

which leads to the quadratic inequality

$$2|S_1|^2\epsilon^2 - |S_1|\ln(6/\delta) - 8\ln^2(6/\delta) \geq 0.$$

with solution

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

There is no fixed strategy for scheduling. In our algorithm we follow the results of Provost et al. [19] as well as Riondato and Upfal [21] where a geometric sampling schedule is proposed as a strategy (the authors conjecture that such strategy is optimal, but we do not need such assumption), i.e., the one that $S_i = c^i S_1$, for each $i \geq 1$ and for a constant $c > 1$.

Given $0 < \epsilon, \delta < 1$, let $(|S_i|)_{i \geq 1}$ be a geometric sample schedule with starting sample size defined in Equation (1). We present the outline of the algorithms for estimating the shortest path centrality and for the computation of shortest paths with such centrality at least ϵ. Both algorithms return the correct output with probability $1 - \delta$. For instance, the first algorithm outputs a table $\tilde{c}$ with the centrality estimation, while the second outputs a table d

with the distances between the pair of vertices and it stops sampling if the sample size reach the bound in Theorem 9 (ii). Since both algorithms are similar, we outline them in parallel.

Consider the table for the estimation of canonical shortest paths tree $\tilde{t}$ and the set $\mathcal{V}$ that contains the values in $\tilde{t}$ without repetition. The following steps are repeated for each $i \geq 1$. For the sake of clarity, $S_0 = \emptyset$.

step 1. Create a sample of $k = |S_i| - |S_{i-1}|$ elements of U chosen uniformly and independently (with replacement) at random;

step 2. Sample a vertex a, compute a canonical shortest path tree T_a and an array of distances *dist* of size $n-1$ from a to each $y \in V$, $a \neq y$. Sample a vertex $b \in \{V \setminus \{a\}\}$ and get the corresponding branch $\mathcal{B}_a(b) \in S(T_a)$. For every shortest path $p_{uv} \in S(\mathcal{B}_a(b))$, increase the value $\tilde{t}(u,v)$ by 1. Repeat this step k times;

step 3. Compute the bound to $\tilde{R}_r(\mathcal{F}, S_i)$ by minimizing the function defined in Theorem 12. If it satisfies the stopping condition defined in Theorem 11 or if the sample size corresponds to the bound in Theorem 9 (ii), then return the set $\{d(u,v), (u,v) \in V^2 \text{ and } d(u,v) > 0\}$ (and also $\{\tilde{c}(u,v) = \tilde{t}_{uv}/|S_i|, (u,v) \in V^2 \text{ and } \tilde{t}_{uv} > 0\}$ if the estimation $\tilde{c}$ is being computed). Otherwise, increase the size of S_i until it has size $|S_{i+1}|$, increase i and return to step 1.

Step 1 is trivial and step 2 can be performed by running Dijkstra's Algorithm in time $\mathcal{O}(m + n \log n)$ in the input graph G for each sampled vertex a. The update of tables $\tilde{t}$ and d is performed by a procedure running on the branch $\mathcal{B}_a(b)$ described in Algorithm 1. Since G is undirected, for each subpath found in $\mathcal{B}_a(b)$, we can update both $d(u,v)$ and $d(v,u)$ (and also $\tilde{c}(u,v)$ and $\tilde{c}(v,u)$). Note that since $S(\mathcal{B}_a(b))$ might have $\mathcal{O}(\mathrm{Diam}_V(G)^2)$ shortest paths, this algorithm performs $\mathcal{O}(\mathrm{Diam}_V(G)^2)$ updates in the output table. The dimension of the tables d, $\tilde{c}$ and $\tilde{t}$ are $\min(\lg \mathrm{Diam}_V(G) \cdot \mathrm{Diam}_V(G)^2, n^2)$, since in the worst case $\mathrm{Diam}_V(G) = n$ and a value is not reinserted in the table during the traversing of the algorithm.

Next we give the algorithm that updates tables $\tilde{t}$ and d and the main algorithm (Algorithms 1 and 2, respectively) in detail. For the storage of each value in $\mathcal{V}$ in a sparse way and without repetition, Algorithm 1 keeps an array *count* of size n, such that each value in *count*$[p]$ contains the amount of pairs of vertices having p branches from canonical shortest path trees, for $1 \leq p \leq n$. For the sake of clarity, we present in Algorithm 2 the computation of table d with the option to compute table $\tilde{c}$ if the indicator variable for the shortest path centrality *spc* is equal to one.

▶ **Theorem 15.** *Consider a sample $S_r = \{T_1, \ldots, T_r\}$ of size r and let η_i be the value obtained in line 15 on the i-th iteration. Then $\eta_r = 2w_s + \frac{\ln \frac{3}{\delta_r} + \sqrt{(\ln \frac{3}{\delta_r} + 4|S_r|w_s) \ln \frac{3}{\delta_r}}}{|S_r|} + \sqrt{\frac{\ln \frac{3}{\delta_r}}{2|S_r|}}$, where $\delta_r = \delta/2^r$, is the value where r is the minimal $i \geq 1$ such that $\eta_r \leq \epsilon$ for the input graph $G = (V, E)$ and for fixed constants $0 < \epsilon, \delta < 1$. Algorithm 2 returns with probability at least $1 - \delta$ the exact distance $d(u,v)$, for each $(u,v) \in V^2$ such that $d(u,v) > 0$, and the corresponding shortest path between the vertices u and v whenever p_{uv} has centrality at least ϵ. Additionally, the value of $\tilde{c}(u,v)$ is within ϵ error to the value of $c(u,v)$ with probability $1 - \delta$.*

Proof. Let $i \geq 1$ be an iteration of the loop in 5–16 and let E_i be the event where $\sup_{(u,v) \in V^2} |\tilde{c}(u,v) - c(u,v)| > \eta_i$ in this iteration. We need the event E_i occurring with probability at most δ for some iteration i. That is, we need

Algorithm 1 updateShortestPaths($dist$, $\mathcal{B}_a(b)$, d, $\tilde{t}$, $\mathcal{V}$, $count$)

Data: Array of distances $dist$ generated by Dijkstra algorithm, branch $\mathcal{B}_a(b)$, exact distances table d, number of canonical shortest path trees table $\tilde{t}$, set $\mathcal{V}$, counter array $count$.

Result: Update of the distances array d and the values in $\tilde{t}$ using paths from $\mathcal{B}_a(b)$.

1 **for** $j \leftarrow 1$ *to* $|\mathcal{B}_a(b)|$ **do**
2 **for** $i \leftarrow j+1$ *to* $|\mathcal{B}_a(b)|$ **do**
3 $b_i \leftarrow \mathcal{B}_a(b)[i]$
4 $b_j \leftarrow \mathcal{B}_a(b)[j]$
5 $d[b_j][b_i] \leftarrow d[b_i][b_j] \leftarrow dist[b_i] - dist[b_j]$
6 **if** *first time insertion of* b_j *and* b_i *in* **then**
7 $\tilde{t}[b_j][b_i] \leftarrow \tilde{t}[b_i][b_j] \leftarrow 0$
8 $\tilde{t}[b_j][b_i] \leftarrow \tilde{t}[b_i][b_j] \leftarrow \tilde{t}[b_i][b_j] + 1$
9 $count[\tilde{t}[b_j][b_i]] \leftarrow count[\tilde{t}[b_j][b_i]] + 1$
10 **if** $(\tilde{t}[b_j][b_i]$ *or* $\tilde{t}[b_i][b_j]) \notin \mathcal{V}$ **then**
11 $\mathcal{V}.add(\tilde{t}[b_j][b_i])$
12 **if** $count[\tilde{t}[b_j][b_i] - 1] \geq 1$ **then**
13 $count[\tilde{t}[b_j][b_i] - 1] \leftarrow count[\tilde{t}[b_j][b_i] - 1] - 1$
14 **if** $count[\tilde{t}[b_j][b_i] - 1] = 0$ **then**
15 $\mathcal{V}.remove(\tilde{t}[b_j][b_i])$

$$\Pr(\exists i \geq 1 \text{ s.t. } E_i \text{ occurs}) \leq \sum_{i=1}^{\infty} \Pr(E_i) \leq \delta,$$

where the inequality comes from union bound. Setting $\Pr(E_i) = \delta/2^i$, we have

$$\sum_{i=1}^{\infty} \Pr(E_i) = \delta \sum_{i=1}^{\infty} \frac{1}{2^i} = \delta.$$

Let $S_r = \{T_1, \ldots, T_r\}$ be the final sample obtained after the iteration r in the loop 5–16 where the stopping condition is satisfied, i.e., $\eta_r \leq \epsilon$. For each iteration i in 5–16, where $1 \leq i \leq r$, consider that for each $a \in V$, there is one Dijkstra tree $T_a \in \mathcal{T}$, and hence, $|\mathcal{T}| = n$. A vertex $a \in V$ is sampled with probability $1/n$ and a vertex b is sampled with probability $1/n-1$ in lines 8 and 10, respectively; therefore, $\mathcal{B}_a(b)$ is sampled with probability $1/n(n-1)$ (line 11). The branch $\mathcal{B}_a(b)$ is traversed by Algorithm 1, and the distances of every shortest path in $S(\mathcal{B}_a(b))$ are clearly correctly and exactly computed.

Let $p_{uv} \in S(\mathcal{B}_a(b))$ be a shortest path from u to v in the sampled branch. At this point, $d(u,v)$ has the exact distance and the shortest path from u to v correctly computed. We will show by contraposition that if $c(u,v) \geq \epsilon$, then $|\tau_{uv} \cap S_r| \geq 1$, where $\tau_{uv} = \{\mathcal{B}_c(d) \in \bigcup_{a \in V} \bigcup_{b \in V : b \neq a} \mathcal{B}_a(b) : p_{uv} \in S(\mathcal{B}_c(d))\}$ and $c(u,v) = \mathbb{E}[f_{uv}(\mathcal{B}_a(b))] = \Pr_\pi(\tau_{uv})$.

If $|\tau_{uv} \cap S_r| < 1$, then there is no branch in S_r that contains a shortest path from u to v, and hence $\tilde{c}(u,v) = 0$. Then the value of $c(u,v)$ must be at most ϵ so that $|\tilde{c}(u,v) - c(u,v)| \leq \epsilon$ holds. Therefore, if $c(u,v) \geq \epsilon$, then $|\tau_{uv} \cap S_r| \geq 1$ and S_r with size at most $\lceil \frac{c}{\epsilon} \lfloor 2 \lg \mathrm{Diam}_V(G) + 1 \rfloor \ln \frac{1}{\epsilon} + \ln \frac{1}{\delta} \rceil$ is an ϵ-net with probability at least $1 - \delta$. The probability that $d(u,v)$ (as well as $d(v,u)$) is exactly computed is $\geq 1 - \delta$ (Theorem 9 (ii)).

Now consider the computation of the estimation $\tilde{c}(u,v)$, for a pair $(u,v) \in V^2$. Let $p_{uv} \in S(G)$ be a shortest path from u to v and let $S' \subseteq S_r$ be the set of sampled branches

such that $p_{uv} \in S(\mathcal{B}_{x'_i}(y'_i))$, where $\mathcal{B}_{x'_i}(y'_i) \in S'$. If the branch sampled in lines 8 and 10 of Algorithm 2 is in S', then the value $\tilde{t}(u,v)$ has its value increased by 1 in line 8 of Algorithm 1, so at the end or r-th iteration in loop 5–16, $\tilde{c}(u,v) = \frac{\tilde{t}(u,v)}{|S_r|} = \frac{1}{|S_r|}\sum_{\mathcal{B}_{x'_i}(y'_i)\in S'} 1 = \frac{1}{|S_r|}\sum_{\mathcal{B}_{x_i}(y_i)\in S} \mathbb{1}_{\tau_{uv}}(\mathcal{B}_{x_i}(y_i)) = \frac{1}{|S_r|}\sum_{\mathcal{B}_{x_i}(y_i)\in S} f_{uv}(\mathcal{B}_{x_i}(y_i))$.

Since $\eta_r \leq \epsilon$, $L_S(f_{uv}) = \tilde{c}(u,v)$ and $L_U(f_{uv}) = c(u,v)$ (Theorem 13) for all $(u,v) \in V^2$ and $f_{uv} \in \mathcal{F}$, then $\Pr(|\tilde{c}(u,v) - c(u,v)| \leq \epsilon) \geq 1 - \delta$ (Theorem 11). ◀

Algorithm 2 ProbabilisticAllPairsShortestPaths(G,ϵ,δ,spc)

Data: Weighted graph $G = (V,E)$ with $n = |V|$ and $m = |E|$, accuracy parameter $0 < \epsilon < 1$, confidence parameter $0 < \delta < 1$, sample scheduling $(S_i)_{i\geq 1}$
Result: Probabilistic Shortest Path Distance $d[u][v]$, for each $(u,v) \in V^2$ with $d[u][v] > 0$.

1 $count[i] \leftarrow 0, \forall i \in 1,\ldots,n$
2 $|S_0| \leftarrow 0$
3 $\mathcal{V} \leftarrow \emptyset$
4 $i \leftarrow 0, j \leftarrow 1$
5 **do**
6 $i \leftarrow i + 1$
7 **for** $l \leftarrow 1$ **to** $|S_i| - |S_{i-1}|$ **do**
8 sample $a \in V$ with probability $1/n$
9 T_a, $dist \leftarrow$ singleSourceShortestPaths(a)
10 sample $b \in \{V \setminus \{a\}\}$ with probability $1/(n-1)$
11 $\mathcal{B}_a(b) \leftarrow$ shortest path from a to b in T_a
12 updateShortestPath($dist$, $\mathcal{B}_a(b)$, d, $\tilde{t}$, $\mathcal{V}$, $count$)
13 $w_s \leftarrow \min_{s\in\mathbb{R}^+} \frac{1}{s}\ln\sum_{t\in\mathcal{V}} \exp\frac{s^2 t}{2|S_i|^2}$
14 $\delta_i \leftarrow \delta/2^i$
15 $\eta \leftarrow 2w_s + \frac{\ln\frac{3}{\delta_i} + \sqrt{(\ln\frac{3}{\delta_i} + 4|S_i|w_s)\ln\frac{3}{\delta_i}}}{|S_i|} + \sqrt{\frac{\ln\frac{3}{\delta_i}}{2|S_i|}}$
16 **while** $\eta > \epsilon$
17 $\tilde{c}[u][v] \leftarrow \tilde{t}[u][v]/|S_i|$, for each $(u,v) \in V^2$ such that $\tilde{t}[u][v] > 0$
18 **if** $spc = 1$ **then**
19 **return** $d[u][v]$ and $\tilde{c}[u][v]$, for each$(u,v) \in V^2$ such that $\tilde{t}[u][v] > 0$
20 **else**
21 **return** $d[u][v]$, for each $(u,v) \in V^2$ such that $d[u][v] > 0$

▶ **Theorem 16.** *Given an undirected weighted graph $G = (V,E)$ with $n = |V|$ and a sample of size at most $r = \lceil \frac{c}{\epsilon}\lfloor 2\lg Diam_V(G) + 1\rfloor \ln\frac{1}{\epsilon} + \ln\frac{1}{\delta}\rceil$, Algorithm 2 has expected running time $\mathcal{O}(\lg Diam_V(G)\cdot\max(m + n\log n, Diam_V(G)^2)$ for the computation of table d.*

Proof. We sample the vertex $a, b \in V$ in lines 8 and 10 in linear time. Algorithm 1 (line 12) takes time $\mathcal{O}(\mathrm{Diam}_V(G)^2)$ since in the worst case, the length of a branch $\mathcal{B}_a(b)$ cannot be deeper than the diameter of the graph. Line 13 is executed by an algorithm that is linear in the size of the sample [13]. The loop in lines 5–16 runs at most r times and the Dijkstra algorithm which is executed in line 9 has running time $\mathcal{O}(m + n\log n)$. The operations of insertion, deletion and search in tables $\tilde{t}$ and d take time $\mathcal{O}(1)$ on average. So, the total expected running time of Algorithm 2 is $\mathcal{O}(r\max(m + n\log n, \mathrm{Diam}_V(G)^2)) = \mathcal{O}(\lg\mathrm{Diam}_V(G)\cdot\max(m + n\log n, \mathrm{Diam}_V(G)^2))$. ◀

▶ **Corollary 17.** *Given an undirected weighted graph $G = (V, E)$ with $n = |V|$ and a sample of size $r = \lceil \frac{c}{\epsilon^2} \lfloor 2 \lg Diam_V(G) + 1 \rfloor \ln \frac{1}{\delta} \rceil$, Algorithm 2 has running time $\mathcal{O}(\lg Diam_V(G) \cdot \max(m + n \log n, Diam_V(G)^2))$ for the computation of table $\tilde{c}$.*

4 Concluding remarks

In this paper we presented a $\mathcal{O}(\lg \mathrm{Diam}_V(G) \cdot \max(m+n \log n, \mathrm{Diam}_V(G)^2)$ expected running time algorithm that outputs a shortest path between every pair of vertices (u, v) with probability at least $1 - \delta$ whenever the shortest path centrality of (u, v) is at least ϵ, for fixed constants $0 < \epsilon, \delta < 1$. The algorithm uses the idea of sampling and we show that a sample of shortest paths of size $\lceil \frac{c}{\epsilon} \lfloor 2 \lg \mathrm{Diam}_V(G) + 1 \rfloor \ln(\frac{1}{\epsilon}) + \ln \frac{1}{\delta} \rceil$ is sufficient for achieving the desired result. We note that this is particularly interesting in sparse graphs with logarithmic diameter, which are common in practical applications, since for those graphs the sample size drops exponentially in comparison to a sample size obtained by, for instance, Hoeffding and union bounds. In such cases, the algorithm runs in $O(n \log n \log \log n)$ time. So, in an application where one might be interested only in computing "central" shortest paths the algorithm is rather efficient.

Finally, an open question that we are particularly interested is the connection between ϵ and n for specific input distributions. For the general case, trivially setting $\epsilon = 2/n$, by Theorem 1, we have a guarantee that every shortest path in G is computed with probability $1 - \delta$, but that would increase the algorithm complexity to $\tilde{O}(n^3)$. We wonder if this fact may be related to the assumption that the APSP may not admit a strictly subcubic algorithm. However, if we assume that the graph is sampled from a given probability distribution, a strictly subcubic randomized algorithm for the (original) APSP may be achievable.

References

1. A. Abboud and V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 434–443, Oct 2014. doi:10.1109/FOCS.2014.53.
2. A. Abboud, V. Williams, and H. Yu. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM Journal on Computing*, 47(3):1098–1122, 2018. arXiv:https://doi.org/10.1137/15M1050987, doi:10.1137/15M1050987.
3. M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
4. Surender Baswana and Telikepalli Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM J. Comput.*, 39(7):2865–2896, May 2010. doi:10.1137/080737174.
5. Karl Bringmann, Marvin Künnemann, and Karol Wundefinedgrzycki. Approximating apsp without scaling: Equivalence of approximate min-plus and exact min-max. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 943–954, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316373.
6. Andrej Brodnik and Marko Grgurovič. Solving all-pairs shortest path by single-source computations: Theory and practice. *Discrete applied mathematics*, 231:119–130, 2017.
7. Timothy M. Chan. All-pairs shortest paths for unweighted undirected graphs in o(mn) time. *ACM Trans. Algorithms*, 8(4), October 2012. doi:10.1145/2344422.2344424.
8. Alane M. de Lima, Murilo. V. G. da Silva, and André L. Vignatti. Estimating the percolation centrality of large networks through pseudo-dimension theory. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), San Diego, USA*, 2020.

9 Alane M. de Lima, Murilo. V. G. da Silva, and André L. Vignatti. Percolation centrality via rademacher complexity. *Manuscript*, 2020.

10 Pavlos Eirinakis, Matthew D Williamson, and K Subramani. On the shoshan-zwick algorithm for the all-pairs shortest path problem. *J. Graph Algorithms Appl.*, 21(2):177–181, 2017.

11 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Almost shortest paths and pram distance oracles in weighted graphs. *arXiv preprint: 1907.11422*, 2019.

12 Sariel Har-Peled and Micha Sharir. Relative (p, ε)-approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011.

13 Steven G. Johnson. The nlopt nonlinear-optimization package, 2014. URL: `http://github.com/stevengj/nlopt`.

14 M. Löffler and J. M. Phillips. Shape fitting on point sets with probability distributions. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 313–324. Springer Berlin Heidelberg, 2009.

15 M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2nd edition, 2017.

16 M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

17 Luca Oneto, Alessandro Ghio, Davide Anguita, and Sandro Ridella. An improved analysis of the rademacher data-dependent bound using its self bounding property. *Neural Networks*, 44:107 – 111, 2013. `doi:https://doi.org/10.1016/j.neunet.2013.03.017`.

18 Seth Pettie and Vijaya Ramachandran. Computing shortest paths with comparisons and additions. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 267–276, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=545381.545417`.

19 Foster Provost, David Jensen, and Tim Oates. Efficient progressive sampling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, page 23–32, New York, NY, USA, 1999. Association for Computing Machinery. `doi:10.1145/312129.312188`.

20 M. Riondato and E. M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, 2016.

21 M. Riondato and E. Upfal. Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages. *ACM Trans. Knowl. Discov. Data*, 12(5):61:1–61:38, July 2018.

22 Matteo Riondato and Eli Upfal. Mining frequent itemsets through progressive sampling with rademacher averages. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1005–1014, New York, NY, USA, 2015. Association for Computing Machinery. `doi:10.1145/2783258.2783265`.

23 S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

24 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 664–673, New York, NY, USA, 2014. ACM. `doi:10.1145/2591796.2591811`.

25 Raphael Yuster. Approximate shortest paths in weighted graphs. *Journal of Computer and System Sciences*, 78(2):632 – 637, 2012. Games in Verification. `doi:https://doi.org/10.1016/j.jcss.2011.09.001`.