

A Simple, Nearly-Optimal Algorithm for Differentially Private All-Pairs Shortest Distances

Jesse Campbell*

Chunjiang Zhu†

Abstract

The all-pairs shortest distances (APSD) with differential privacy (DP) problem takes as input an undirected, weighted graph $G = (V, E, \mathbf{w})$ and outputs a private estimate of the shortest distances in G between all pairs of vertices. In this paper, we present a simple $\tilde{O}(n^{1/3}/\varepsilon)$ -accurate algorithm to solve APSD with ε -DP, which reduces to $\tilde{O}(n^{1/4}/\varepsilon)$ in the (ε, δ) -DP setting, where $n = |V|$. Our algorithm greatly improves upon the error of prior algorithms [CGK⁺23, FLL22], namely $\tilde{O}(n^{2/3}/\varepsilon)$ and $\tilde{O}(\sqrt{n}/\varepsilon)$ in the two respective settings, and is the first to be optimal up to a polylogarithmic factor, based on a lower bound of $\tilde{\Omega}(n^{1/4})$ proven by [BDG⁺24].

In the case where a multiplicative approximation is allowed, we give two different constructions of algorithms with reduced additive error. Our first construction allows a multiplicative approximation of $O(k \log \log n)$ and has additive error $\tilde{O}(k \cdot n^{1/k}/\varepsilon)$ in the ε -DP case and $\tilde{O}(\sqrt{k} \cdot n^{1/(2k)}/\varepsilon)$ in the (ε, δ) -DP case. Our second construction allows multiplicative approximation $2k - 1$ and has the same asymptotic additive error as the first construction. Both constructions significantly improve upon the currently best-known additive error of, $\tilde{O}(k \cdot n^{1/2+1/(4k+2)}/\varepsilon)$ and $\tilde{O}(k \cdot n^{1/3+2/(9k+3)}/\varepsilon)$, respectively from [CGK⁺23]. Our algorithms are straightforward and work by decomposing a graph into a set of spanning trees, and applying a key observation by [Sea16] that we can privately release APSD in trees with $O(\text{polylog}(n))$ error.

1 Introduction

1.1 Differential Privacy and APSD

Differential privacy (DP) is a mathematical framework which quantifies the leakage of sensitive user information by an algorithm. Differential privacy has seen applications in industry and government agencies, with algorithms satisfying differential privacy being adopted by organizations such as Google [ABC⁺20, BDD⁺20, BBD⁺21], Microsoft [DKY17], and the US Census Bureau [GAP18, FMM19]. The definition of differential privacy requires that the probability of a certain outcome being attained from a dataset without the contribution of any single individual is very close to the probability the same outcome is attained from a dataset with the individual's contribution restored, with the notion of *closeness* depending on chosen privacy parameters ε and δ . In the development of differentially private algorithms, a key trade-off occurs between privacy and accuracy, where privacy is, loosely speaking, the amount of leakage of sensitive information, and accuracy is how close the private output is to its true value.

The *all-pairs shortest distances* (APSD) problem is a cornerstone of the field of graphs algorithms and has been studied extensively. Given a connected graph $G = (V, E, \mathbf{w})$ with $n = |V|$, the goal is to release the shortest distances in G between every vertex pair in $V \times V$. Algorithms such as the notable Floyd-Warshall algorithm solve APSD in $O(n^3)$ time. It is conjectured that APSD cannot be solved in $n^{3-\Omega(1)}$ time, being one among a list of problems which are either all solvable in subcubic time, or none of them are [WW10].

In 2016, Adam Sealfon [Sea16] was the first to study the application of differential privacy to the all-pairs shortest distances problem. We follow their DP framework where two graphs are neighboring if they share the same underlying unweighted graphs and their edge weight \mathbf{w} differs by at most 1 in the l_1 -distance. This

*Duke Kunshan University (jesse.campbell@duke.edu)

†University of North Carolina at Greensboro (chunjiang.zhu@uncg.edu)

definition is suitable for problems where the graph topology $G = (V, E)$ is public and the edge weights \mathbf{w} need to be kept private. For example, the topology of a road map in a navigation system is public knowledge while the weight of an edge depends on the number of vehicles on it, which is typically estimated based on clearly private GPS locations of vehicles. In particular, [Sea16] showed an algorithm based on adding Laplace noise to every edge weight which solves the problem with $\tilde{O}(n/\epsilon)$ error (where $\tilde{O}(\cdot)$ hides a polylogarithmic term), and posed the question about whether it can be solved with error sublinear in n . In addition, they considered special cases and presented ϵ -DP algorithms to release APSD on trees with $O(\log^{2.5}(n)/\epsilon)$ error, and release APSD on graphs of edge weights bounded by $M > 0$ with error $\tilde{O}((nM)^{2/3}/\epsilon^{1/3})$ ($\tilde{O}(\sqrt{nM}/\epsilon)$ in the (ϵ, δ) -DP setting). A key observation we make is that the polylogarithmic error in trees is significantly smaller than the polynomial error in general graphs, inspiring the development of the techniques in this work.

Later, Fan et al. [FLL22] and Chen et al. [CGK⁺23] both positively answered Sealfon’s question by proving algorithms for general graphs with error $\tilde{O}(n^{2/3}/\epsilon)$ in the ϵ -DP case and $\tilde{O}(\sqrt{n}/\epsilon)$ in the (ϵ, δ) -DP case. In addition, Chen et al. gave algorithms with improved accuracy for graphs with bounded weights, namely $O(n^{0.5616})$ in the ϵ -DP setting and $O(n^{0.4143})$ in the (ϵ, δ) -DP setting. They also presented an algorithm based on the celebrated distance oracle of Thorup and Zwick [TZ05] which reduces the additive error for APSD on general graphs by introducing a multiplicative error term. Moreover, they showed an additive error lower bound of $\tilde{\Omega}(n^{1/6})$ for the problem on general graphs. This lower bound was recently improved to $\tilde{\Omega}(n^{1/4})$ by Bodwin et al. [BDG⁺24].

1.2 Our Results

In this paper, we improve upon the accuracy of the currently best-known algorithms for APSD with DP under both additive and mixed additive and multiplicative approximations. In particular, we show the additive approximation problem to be $\tilde{\Theta}(n^{1/4})$ -accurate by giving a nearly optimal algorithm which matches the known lower bound of $\tilde{\Omega}(n^{1/4})$ up to a polylogarithmic factor.

Additive Approximation

Section 3 gives algorithms to release estimates of all-pairs distances which incur additive error terms. Table 1 gives a summary of the size of the additive error term for each released distance for known algorithms compared to the improved error in the algorithms in this paper. We greatly improve the additive error from $\tilde{O}(n^{2/3}/\epsilon)$ to $\tilde{O}(n^{1/3}/\epsilon)$ in the ϵ -DP case and from $\tilde{O}(\sqrt{n}/\epsilon)$ to $\tilde{O}(n^{1/4}/\epsilon)$ in the (ϵ, δ) -DP case. The latter matches the lower bound $\tilde{\Omega}(n^{1/4})$ of Bodwin et al. [BDG⁺24] (up to a polylog factor) for any (ϵ, δ) -DP algorithm. The new upper bounds are even smaller than the previously best results $O(n^{0.5616})$ and $O(n^{0.4143})$ for graphs of bounded weights [CGK⁺23] while we don’t need the assumption on edge weights.

Citation	ϵ -DP	(ϵ, δ) -DP
Sealfon (2016) [Sea16]	$\tilde{O}(n/\epsilon)$	-
Chen et al.; Fan et al. (2022) [CGK ⁺ 23, FLL22]	$\tilde{O}(n^{2/3}/\epsilon)$	$\tilde{O}(\sqrt{n}/\epsilon)$
This paper	$\tilde{O}(n^{1/3}/\epsilon)$	$\tilde{O}(n^{1/4}/\epsilon)$

Table 1: Known results in additive error upper bounds for APSD with DP.

Mixed Additive and Multiplicative Approximations

When a multiplicative approximation is allowed, it is theoretically possible to break the lower bound of $\tilde{\Omega}(n^{1/4})$ for general graphs, however no such algorithm has existed until now. In **Section 4**, we give two constructions to release APSD with differential privacy that break the lower bound for additive error in general graphs. If an approximation $\tilde{f}(\mathbf{x})$ of $f(\mathbf{x})$ is (α, β) -accurate, we mean that $f(\mathbf{x}) - \beta \leq \tilde{f}(\mathbf{x}) \leq \alpha \cdot f(\mathbf{x}) + \beta$. Table 2 gives a summary of the error bounds for known algorithms compared to the improved bounds in the algorithms in this paper. The additive term gets significantly improved compared to previous results

in [CGK⁺23], i.e., from $\tilde{O}(k \cdot n^{1/2+1/(4k+2)}/\varepsilon)$ to $\tilde{O}(k \cdot n^{1/k}/\varepsilon)$ in the ε -DP case and from $\tilde{O}(k \cdot n^{1/3+2/(9k+3)}/\varepsilon)$ to $\tilde{O}(\sqrt{k} \cdot n^{1/(2k)}/\varepsilon)$ in the (ε, δ) -DP case. While our first result slightly increases the multiplicative error from $2k - 1$ to $O(k \log \log n)$, its simplicity makes it a good alternative and *warm-up* for the subsequent construction.

Citation	ε -DP	(ε, δ) -DP
Chen et al. (2023) [CGK ⁺ 23]	$(2k - 1, \tilde{O}(k \cdot n^{1/2+1/(4k+2)}/\varepsilon))$	$(2k - 1, \tilde{O}(k \cdot n^{1/3+2/(9k+3)}/\varepsilon))$
This paper	$(O(k \log \log n), \tilde{O}(k \cdot n^{1/k}/\varepsilon))$	$(O(k \log \log n), \tilde{O}(\sqrt{k} \cdot n^{1/(2k)}/\varepsilon))$
This paper	$(2k - 1, \tilde{O}(k \cdot n^{1/k}/\varepsilon))$	$(2k - 1, \tilde{O}(\sqrt{k} \cdot n^{1/(2k)}/\varepsilon))$

Table 2: Known results in error bounds for APSD with DP where $k \in \mathbb{Z}^+$ is a parameter given as input.

1.3 Our Techniques

Algorithm with Additive Approximation. We give a simple algorithm which releases a private approximation of the shortest path distances between all-pairs of vertices while only incurring an additive approximation. Given a pair of vertices, their private shortest distance is computed in one of two ways depending on the number of edges in a shortest path between them (ties broken arbitrarily). For *short* paths, or those whose hop lengths are bounded above by some constant t , we can directly apply the Laplace mechanism to release their shortest path distance. Namely, we perturb the edge weights in G by adding noise sampled from the Laplace distribution, and compute the shortest path distance on the noisy graph. By a concentration inequality for Laplace random variables (**Lemma 2.4**), we can achieve additive error $\tilde{O}(\sqrt{t}/\varepsilon)$ for distances computed via this method. Note that previous constructions [Sea16, CGK⁺23] also add Laplace noise to edge weights, but their analysis simply sums up the error accumulated from each edge in the path, leading to an error $\tilde{O}(t/\varepsilon)$.

For *long* shortest paths between vertices, we utilize a result from Sealfon [Sea16] to release the distances from the root vertex of a rooted tree to all other vertices with $O(\text{polylog}(n)/\varepsilon)$ error. In particular, we construct a *hitting set* of vertices of size s from which to grow shortest path trees. Then, we privately release the distance between the root vertex in the hitting set and each other vertex in the graph with low error. If at least one of the vertices in the hitting set intersects the shortest path between two vertices, then the shortest path is completely contained within a shortest path tree, and the shortest path distance is the sum of two distances to the root vertex. Moreover, if we assume that the number of edges in the shortest path is bounded below by some constant, there is a high probability that the hitting set will intersect the path. We can divide our privacy budget (ε) among all of the s shortest path trees, such that by basic composition releasing their distances is ε -DP, giving an additive error of $\tilde{O}(s/\varepsilon)$.

In theory, letting $s := \tilde{O}(n/t)$ is an appropriate size to ensure that the hitting set intersects every shortest path that traverses at least t edges with high probability. In the ε -DP case, we choose $s := \tilde{O}(n^{1/3})$ to minimize the error from the two methods. The (ε, δ) -DP case is similar, except we use advanced composition to release the distance estimates computed in each shortest path tree, making the error from this method $\tilde{O}(\sqrt{s}/\varepsilon)$, hence the final error is $\tilde{O}(n^{1/4}/\varepsilon)$.

Algorithms with Additive and Multiplicative Approximations. At a high level, our algorithms work by constructing a collection of spanning trees \mathbf{T} , the shortest path distances in-which can be used to estimate the shortest path distances in G for every pair of vertices with a multiplicative error. Then, in the same vein as the previous section, we can release APSD for each tree in \mathbf{T} with additive error $\tilde{O}(|\mathbf{T}|/\varepsilon)$. Releasing APSD in this way yields a combination of multiplicative and additive errors.

In the first construction, the collection of spanning trees \mathbf{T} is simply the *tree-padding spanner* as constructed by Abraham et al. [ACE⁺20]. In particular, they showed that, for each vertex $v \in V$, there is a tree $\mathcal{T}_v \in \mathbf{T}$ such that $d_{\mathcal{T}_v}(v, u) \leq O(k \log \log n) \cdot d_G(v, u)$ for all $u \in V$, where $k \in \mathbb{Z}^+$ is a parameter. The number of trees in \mathbf{T} is $kn^{1/k}$, hence the total algorithm is $(O(k \log \log n), \tilde{O}(kn^{1/k}/\varepsilon))$ -accurate. The (ε, δ) -DP case is very similar, except we use advanced composition release the distance estimates computed within each tree, exactly as in the previous section, reducing the additive error to $O(\sqrt{k}n^{1/(2k)})$. Although

the formation of \mathbf{T} in this construction is simple, we include it as a *warm-up* for the next construction, which uses the same mechanism for differentially private release, but requires a more in-depth approach to construct \mathbf{T} .

Our next construction of \mathbf{T} is based on the celebrated distance oracle of Thorup and Zwick [TZ05], which allows us to reduce the multiplicative approximation from the previous construction to $2k - 1$. The original distance oracle constructs sets of connected vertices called *clusters*, which are computed based on the sets $V = A_0 \supseteq A_1 \supseteq \dots \supseteq A_{k-1}$ which decrease in size as the index increases. The original oracle constructs a shortest-path tree which spans the subgraph induced by each cluster, and computes distances estimates based on the distances in these trees. If we were to directly apply Sealfon's result to these cluster-spanning trees, the additive error would be $\tilde{O}(|A_1|/\varepsilon) = \tilde{O}(n^{1-1/k}/\varepsilon)$, which is higher than the error in our additive approximation.

Instead, we *pack* multiple clusters into a single spanning tree of G such that we can reduce the number of trees in \mathbf{T} . In the original distance oracle, the total number of edges in the combined spanning trees of the clusters is $O(kn^{1+1/k})$, so ideally we should be able to *fit* all of the clusters into $n^{1/k}$ trees at each level $1 \leq i \leq k - 1$, resulting in a final additive error of $\tilde{O}(kn^{1/k}/\varepsilon)$. The main obstacle is that, in the original distance oracle, the clusters are not necessarily disjoint, and consequently adding the edges from their spanning trees to a single graph may contain cycles. To overcome this, we divide each set A_i into $\tilde{O}(n^{1/k})$ (not necessarily disjoint) sets A_i^r and modify the definition of clusters by ensuring that any vertex in a cluster centered at $w \in A_i^r$ is closer to w than any other center in A_i^r . In this way, the clusters with centers in A_i^r are guaranteed to be disjoint. Hence, we can add all the edges from the trees spanning each subgraph induced by a cluster with center in A_i^r into exactly one spanning tree of G . Consequently, $|\mathbf{T}| = \tilde{O}(kn^{1/k})$ as desired.

For the analysis, we show that our sets A_{i+1} are large enough relative to the size of A_i^r such that every vertex is closer to A_{i+1} than A_i^r with high probability. Going back to the original distance oracle, this implies that our modified clusters are at least as large as the original clusters, and hence any distance we can compute from a shortest path tree spanning the original cluster we can also compute from a shortest path tree spanning our modified cluster. This observation implies that the original stretch analysis from [TZ05] also applies in our setting. In particular, this gives a multiplicative approximation of $2k - 1$. Hence, our total algorithm is $(2k - 1, \tilde{O}(kn^{1/k}))$ -accurate in the ε -DP case and $(2k - 1, \tilde{O}(\sqrt{k}n^{1/(2k)}))$ -accurate in the (ε, δ) -DP case.

1.4 Related Work

In 2009, Hay et al. [HLMJ09] were the first to study differentially private graph algorithms by giving a mechanism to release a private estimate of the degree distribution in a graph. In their work, they introduce the notions of node- and edge-differential privacy, which defines neighboring graphs based on their underlying graph topology. Algorithms with node- and edge-differential privacy have been studied considerably in recent literature [LLXL21, JWC23, LML20, ZNF23, SU21, HCYH24, ELRS23].

By contrast, Sealfon [Sea16] introduced the all-pairs shortest distances problem with differential privacy, defining neighboring graphs based on differing edge weights assigned to a fixed, public graph topology. In particular, Sealfon constructed an ε -DP algorithm that releases APSD in trees with additive error $O(\log^{2.5}(n) \cdot \log(1/\gamma)/\varepsilon)$ with probability $1 - \gamma$, which is the basis of our algorithms in this paper. Furthermore, for graphs with edge weights bounded by $M > 0$, they proved an upper bound of $O((n \cdot M)^{2/3} \cdot \log(n \cdot M \cdot \varepsilon/\gamma)/\varepsilon^{1/3})$ in the pure-DP case and $O(\sqrt{n \cdot M} \cdot \log(1/\delta)/\varepsilon \cdot \log(n \cdot M \cdot \varepsilon/\gamma))$ in the approximate-DP case. For general graphs, Sealfon used a simple application of the Laplace mechanism to achieve additive error $\tilde{O}(n)$ in the ε -DP case, but posed the question about whether an algorithm for general graphs with error sublinear in n is possible.

Later, Chen et al. [CGK⁺23] positively answered Sealfon's question by giving algorithms to release all-pairs shortest distances with error $O(n^{2/3} \cdot \log^{4/3}(n)/\varepsilon)$ in the pure-DP case and $O(\sqrt{n} \cdot \log(n) \cdot \sqrt[4]{\log(1/\delta)}/\varepsilon)$ in the approximate-DP case. Moreover, for graphs with bounded edge weights, they gave an algorithm which is $O(n^{(\sqrt{17}-3)/2+o(1)}/\varepsilon)$ -accurate in the ε -DP case and $O(n^{\sqrt{2}-1+o(1)}/\varepsilon)$ -accurate in the (ε, δ) -DP case. In the case where multiplicative error is allowed, Chen et al. gave an $(2k - 1, O(k \cdot n^{(k+1)/(2k+1)} \cdot \log^2(n)/\varepsilon))$ -accurate

algorithm for APSD with ε -DP and a $(2k - 1, O(n^{(k+1)/(3k+1)} \cdot \log(n)^{(5k+2)/(3k+1)} \cdot \log(1/\delta)^{k/(3k+1)}/\varepsilon))$ -accurate algorithm for APSD with (ε, δ) -DP. Similar to our work, their algorithm is based on the oracle of Thorup and Zwick [TZ05], which they adapted to the DP setting by using the exponential mechanism to select *bunches* of vertices based on their distance to a fixed vertex.

Other notable results are those of Fan et al. [FLL22], who gave an alternative algorithm to Chen et al. for solving APSD with DP with error sublinear in n . Furthermore, Fan and Li [FL22] gave improved algorithms for trees with bounded depth and grid-graphs. Ebrahimi et al. gave improved error bounds for graphs with bounded tree-width [EMK23]. Moreover, related graphs problems such as shortest paths publishing and range query on shortest paths with differential privacy have seen a recent increase in the literature [CSC⁺24, DGUW23].

Another contribution of Chen et al. was proving a lower bound for the additive error of $\Omega(n^{1/6})$ in any algorithm on general graphs, for a sufficiently small $\varepsilon, \delta > 0$, by exploiting a reduction from the *linear queries* problem to APSD, and a hereditary discrepancy lower bounded provided by Chazelle and Lvov [CL01]. Chen et al. noted that the greatest lower bound achievable by this method is $\Omega(n^{1/4})$. On this note, Bodwin et al. [BDG⁺24] proved a lower bound of $\Omega(n^{1/4}/\sqrt{\log n})$ as a result of a better discrepancy lower bound.

2 Preliminaries

Given a mechanism that takes as input a dataset and releases some aggregate statistical information about the dataset, the goal of differential privacy is to protect any one individual's privacy whose data is included in the dataset. For this reason, it is necessary to define the notion of what it means for a dataset to differ in the contribution of one individual. In our case, $G = (V, E, \mathbf{w})$ and $G' = (V, E, \tilde{\mathbf{w}})$ are *neighbors* if their weight vectors differ in the l_1 norm by at most 1.

Definition 2.1. Two graphs $G = (V, E, \mathbf{w})$ and $G' = (V, E, \tilde{\mathbf{w}})$ with equivalent topology are said to be *neighboring* if $\|\mathbf{w} - \tilde{\mathbf{w}}\|_1 \leq 1$.

Given a notion of *neighboring* datasets X and X' , we can define the notion of the sensitivity of a function that takes a dataset as input as the most the function can differ over neighboring datasets.

Definition 2.2. The l_1 sensitivity of $f : \mathcal{X} \rightarrow \mathbb{R}^D$ is defined as $\Delta_1(f) := \max_{X, X'} \|f(X) - f(X')\|_1$, where X, X' are neighboring datasets.

We now present the formal definition of *differential privacy*. Intuitively, a differentially private system limits the amount of information we can gain about any individual in a dataset based solely on the outcome of a mechanism which takes the dataset as input. The *strictness* of this limitation is controlled by privacy parameters $\varepsilon, \delta \geq 0$.

Definition 2.3. An algorithm $\mathcal{M} : \mathcal{X} \rightarrow \mathbb{R}^D$ is said to be (ε, δ) -differentially private if, for all outcomes $S \subseteq \mathbb{R}^D$ and neighboring datasets X, X' ,

$$\mathbb{P}[\mathcal{M}(X) \in S] \leq e^\varepsilon \cdot \mathbb{P}[\mathcal{M}(X') \in S] + \delta$$

We call the case where $\delta = 0$ *pure differential privacy* and the case where $\delta > 0$ *approximate differential privacy*.

One of the strongest aspects of differentially private mechanisms are their composition properties. Namely, the aggregation of outputs from multiple differentially private mechanisms is itself differentially private, albeit with gradually degrading privacy parameters. We present here two main results about the composition of differentially private systems which we will use several times throughout this paper.

Lemma 2.1 (Basic Composition, [DMNS06, DL09]). *Let $\varepsilon, \delta \in [0, 1]$ and $k \in \mathbb{N}$. If we run k mechanisms where each mechanism is $(\varepsilon/k, \delta/k)$ -differentially private, then the entire algorithm is (ε, δ) -differentially private.*

Lemma 2.2 (Advanced Composition, [DMNS06, DR14]). *Let $\varepsilon, \delta \in (0, 1]$ and $k \in \mathbb{N}$. If we run k mechanisms where each mechanism is $\left(\frac{\varepsilon}{2\sqrt{2k \log(2/\delta)}}, \frac{\delta}{2k}\right)$ -DP, then the entire algorithm is (ε, δ) -DP.*

Regarding **Lemma 2.2**, we note that running k mechanisms where each mechanism is $\frac{\varepsilon}{2\sqrt{2k \log(2/\delta)}}$ -DP is at least (ε, δ) -DP, as each individual mechanism we run has an even stronger privacy guarantee.

Next, we present the Laplace mechanism, which is one of the primary tools in the theory of differential privacy. In specific, the Laplace mechanism describes how much one needs to perturb a released statistic in order to guarantee it is ε -DP.

Lemma 2.3 ([DMNS06]). *Given any function $f : \mathcal{X} \rightarrow \mathbb{R}^k$, the Laplace mechanism on input $X \in \mathcal{X}$ independently samples Y_1, \dots, Y_k according to $\text{Lap}(\Delta_1(f)/\varepsilon)$ and outputs,*

$$\mathcal{M}_{f,\varepsilon}(X) = f(X) + (Y_1, \dots, Y_k)$$

The Laplace mechanism is ε -differentially private.

We also present a result about the concentration of Laplace random variables.

Lemma 2.4 ([CSS11]). *Let X_1, \dots, X_t be independent random variables distributed according to $\text{Lap}(b)$, and let $X = X_1 + \dots + X_t$. Then for all $\gamma \in (0, 1)$, with probability at least $1 - \gamma$ we have,*

$$|X| < O(b\sqrt{t} \log(1/\gamma))$$

Next, we state a Chernoff inequality which bounds the tail probability of a binomial random variable.

Lemma 2.5 ([CL06]). *Let $X \sim \text{Binomial}(n, p)$ be a random variable, then,*

$$\mathbb{P}[X \leq \mathbb{E}[X] - \lambda] \leq e^{-\lambda^2/2\mathbb{E}[X]}$$

3 A Nearly-Optimal Algorithm for General Graphs

In this section, we present a differentially private algorithm for APSD on general graphs which is $\tilde{O}(n^{1/3}/\varepsilon)$ -accurate in the ε -DP case and $\tilde{O}(n^{1/4}/\varepsilon)$ -accurate in the (ε, δ) -DP case. Our algorithm is based on a key observation that we can release shortest path distances in trees with $\text{polylog}(n)$ error.

We denote by $d_G(u, v)$ the shortest distance between $u, v \in V$ in the graph G . When it is clear in which graph we are computing the distance, G is not specified. Furthermore, we denote by $h(u, v)$ the *hop length* between u and v , namely, $h(u, v) = d_{G'}(u, v)$ where $G' = (V, E, \{1\}^{|E|})$. We let $d_G^{(t)}(u, v)$ to be the shortest t -hop path between u and v in the graph G , or the shortest path between u and v under the constraint of traversing at most t edges. In particular, if $h(u, v) > t$, then $d_G^{(t)}(u, v) = \infty$.

This section is structured as follows: we will first prove a result which bounds the error of releasing distances via the Laplace mechanism for paths with bounded hop length. Then, we will present the result by Sealfon which gives an accurate algorithm for computing shortest path distances on rooted trees. Finally, we will show how to combine these two mechanisms to release APSD with the claimed error bounds.

The *input perturbation algorithm* takes as input a graph G , adds noise to each edge weight according to $\text{Lap}(1/\varepsilon)$, and releases the shortest distances computed on the noisy graph. It is a simple application of the Laplace mechanism (**Lemma 2.3**) to the function $f : \mathbf{w} \rightarrow \mathbb{R}^{n^2}$ which takes as input the edge weights of a graph and outputs all-pairs shortest distances. The result was proven by [Sea16] and later by [CGK⁺23]. By a more careful analysis, we establish a lower error for this method.

Lemma 3.1. *Let $t \in \mathbb{N}$, $\varepsilon \in (0, 1]$, and $\gamma \in (0, 0.5]$. There is an ε -DP algorithm for computing $\widetilde{d}^{(t)}(u, v)$ for every $u, v \in V$ such that with probability $1 - \gamma$ we have,*

$$\max_{u, v \in V} |\widetilde{d}^{(t)}(u, v) - d_G^{(t)}(u, v)| \leq O(\sqrt{t} \cdot \log(n/\gamma)/\varepsilon)$$

Algorithm 1 Nearly-optimal error for private all-pairs shortest distances release

Input: Graph $G = (V, E, \mathbf{w})$, parameters $\varepsilon \in (0, 1)$ and $\gamma \in (0, 0.5]$

Output: ε -DP distance estimates $(\mathbf{d}(u, v))_{u, v \in V}$

- 1: Randomly sample $S \subseteq V$ uniformly of size $n^{1/3}/\log^{2/3}(n)$
 - 2: **for** $z \in S$ **do**
 - 3: Compute the shortest path tree rooted at z , call it \mathcal{T}_z
 - 4: Let z^* be the vertex in \mathcal{T}_z such that the subtree rooted at z^* has more than $n/2$ vertices, but the subtree rooted at each of z^* 's children has at most $n/2$ vertices.
 - 5: Let $z_1, z_2, \dots, z_\alpha$ be the children of z^*
 - 6: Let \mathcal{T}_i be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T}_z - \{\mathcal{T}_1, \dots, \mathcal{T}_\alpha\}$
 - 7: Sample $X \sim \text{Lap}(2|S| \log(n)/\varepsilon)$ and let $d(z^*, \mathcal{T}_z) = d(z, z^*) + X$
 - 8: Let $d(z, \mathcal{T}_z) = 0$
 - 9: Let $(X_1, X_2, \dots, X_\alpha) \sim \text{Lap}(2|S| \log(n)/\varepsilon)$ and $d(z_i, \mathcal{T}_z) = d(z^*, \mathcal{T}_z) + \text{wt}(z^*, z_i) + X_i$
 - 10: Recursively compute distances in each subtree $\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_\alpha$
 - 11: For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\tilde{d}(z, w) = d(z_i, \mathcal{T}_z) + d(w, \mathcal{T}_i)$
 - 12: **end for**
 - 13: Sample $(Y_1, Y_2, \dots, Y_{|E|}) \sim \text{Lap}(2/\varepsilon)$ and let $\tilde{\mathbf{w}} = \mathbf{w} + (Y_1, Y_2, \dots, Y_{|E|})$, $\tilde{G} = (V, E, \tilde{\mathbf{w}})$
 - 14: Compute $d_{\tilde{G}}^{(t)}(u, v)$ for all $u, v \in V$ where $t := \lceil 10 \cdot (n/s) \log(n) \rceil$
 - 15: Let $\mathbf{d}(u, v) = \min \{d_{\tilde{G}}^{(t)}(u, v), \min_{z \in S} \{\tilde{d}(z, u) + \tilde{d}(z, v)\}\}$ for all $u, v \in V$
 - 16: **return** $(\mathbf{d}(u, v))_{u, v \in V}$;
-

Proof. The algorithm works as follows: given a weighted, undirected graph, $G = (V, E, \mathbf{w})$, we construct a noisy graph $\tilde{G} = (V, E, \tilde{\mathbf{w}})$ by adding Laplace noise to each component of \mathbf{w} , independently sampled from $\text{Lap}(1/\varepsilon)$. Then, we return $\tilde{d}^{(t)}(u, v) \leftarrow d_{\tilde{G}}^{(t)}(u, v)$ for each $u, v \in V$.

Since a shortest path cannot traverse any single edge more than once, the l_1 sensitivity of a function that takes as input \mathbf{w} and returns the shortest path distance between a pair of vertices is 1. Consequently, by **Lemma 2.3**, releasing the graph \tilde{G} is ε -differentially private. Since the shortest t -hop distances are simply a post-processing of the edge weights of \tilde{G} , we can also release them with ε -differential privacy.

To achieve our upper bound for the accuracy, we note that each distance $d_{\tilde{G}}^{(t)}(u, v)$ is the sum of a maximum of t edge weights, hence the total error in the estimate $\tilde{d}^{(t)}(u, v)$ is the accumulation of at most t Laplace random variables. By **Lemma 2.4**, with probability at least $1 - \gamma/n^2$ we have,

$$|\tilde{d}^{(t)}(u, v) - d_G^{(t)}(u, v)| \leq O(\sqrt{t} \cdot \log(n^2/\gamma)/\varepsilon) = O(\sqrt{t} \cdot \log(n/\gamma)/\varepsilon)$$

by **Proposition E.2**. By a union bound, with probability at least $1 - \gamma$, this upper bound holds for each of the n^2 released distances. \square

Next, we will present a result from [Sea16] which gives a recursive algorithm for computing the shortest path distances between the root and all other vertices in a rooted tree, such that the maximum error in any single released distance is $O(\text{polylog}(n))$. We state the result here but defer its proof to **Appendix A**.

Lemma 3.2 (Theorem 4.1 in [Sea16]). *Let $\mathcal{T}_z = (V, E, \mathbf{w})$ be a tree with root vertex $z \in V$, $\varepsilon \in (0, 1]$, and $\gamma \in (0, 0.5]$. Then there is an ε -DP algorithm for releasing the distance between z and every $u \in V$ that is $O(\log^{1.5}(n) \cdot \log(n/\gamma)/\varepsilon)$ -accurate with probability $1 - \gamma$.*

Next, we show how to combine **Lemma 3.1** and **Lemma 3.2** to release all-pairs distances with a nearly-optimal additive error. Our proof borrows some ideas from the proof of **Lemma 3.1** in [CGK⁺23].

Theorem 3.1. *For $\gamma \in (0, 0.5]$ and $\varepsilon \in (0, 1]$, there is an ε -DP algorithm for computing all-pairs shortest distances that is $O(n^{1/3} \cdot \log(n)^{11/6} \cdot \log(1/\gamma)/\varepsilon)$ -accurate with probability $1 - \gamma - 1/n^2$.*

Proof. The algorithm is given in **Algorithm 1** and works by first sampling a subset of vertices $S \subseteq V$ of size s . Then, for each vertex $z \in S$, we compute the shortest path tree rooted at z (by Dijkstra, for instance), call it \mathcal{T}_z . We note that since each tree is constructed by removing edges from G , the sensitivity of the shortest path distance function over \mathcal{T}_z is still 1. For each tree, we call the $\varepsilon/(2s)$ -DP algorithm given by **Lemma**

3.2 to release the shortest path distances between z and $u \in V$, denoted by $\tilde{d}(z, u)$. Furthermore, we use the $\varepsilon/2$ -DP input perturbation algorithm (**Lemma 3.1**) to compute the shortest t -hop distances in G for every pair of vertices $u, v \in V$, where $t := \lceil 10 \cdot (n/s) \log(n) \rceil$, denoted by $d_G^{(t)}(u, v)$. Then, we let,

$$\mathbf{d}(u, v) = \min \{d_G^{(t)}(u, v), \min_{z \in S} \{\tilde{d}(z, u) + \tilde{d}(z, v)\}\}$$

be the final distance approximation. By basic composition (**Lemma 2.1**), running the $s, \varepsilon/(2s)$ -DP mechanisms and the single $\varepsilon/2$ -DP mechanism is ε -DP. In particular, we note that we do not release any shortest path *trees* themselves, hence no shortest path information is revealed. For the accuracy analysis, we will first show that for pair of vertices $(u, v) \in V \times V$, either,

- **Case A.** There is a vertex $z \in S$ that intersects the shortest path in G from u to v , or,
- **Case B.** The shortest path between u and v in G is the t -hop shortest path between u and v .

In particular, assume that (u, v) is not in **Case B**. Let $u = p_1, p_2, \dots, p_l = v$ be a shortest path between u and v in G , with ties broken arbitrarily. Because the vertices in S are sampled uniformly, each vertex in V has probability s/n of being included in S . Consequently, the probability that none of the vertices p_j for $j \in [l]$ are included in S is $(1 - s/n)^l \leq (1 - s/n)^t \leq 1/n^4$ by **Proposition E.1**. It follows that with probability $1 - 1/n^4$, (u, v) is in **Case A**. We condition on this holding true for all n^2 pairs of vertices with probability at least $1 - 1/n^2$, by a union bound. (*)

We begin our accuracy analysis with **Case A**. Suppose $p_i \in S$ for some $i \in [l]$, then $d_G(u, v) = d_{\mathcal{T}_{p_i}}(u, p_i) + d_{\mathcal{T}_{p_i}}(p_i, v)$. We note that these distances are estimated in **Line 11**. Moreover, **Lemma 3.2** ensures that, with probability $1 - \gamma/(2s)$,

$$|d_G(u, v) - (\tilde{d}(u, p_i) + \tilde{d}(p_i, v))| \leq O(s \cdot \log^{1.5}(n) \cdot \log(2sn/\gamma)/\varepsilon). \quad (1)$$

We condition on this upper bound for all s trees with probability at least $1 - \gamma/2$, by a union bound.

We move on to the analysis of **Case B**. Since we are assuming that the true shortest path between u and v has at most t edges, **Lemma 3.1** ensures that with probability $1 - \gamma/2$,

$$|d(u, v) - d_G^{(t)}(u, v)| \leq O(\sqrt{n/s} \cdot \log^{1.5}(n) \cdot \log(1/\gamma)/\varepsilon) \quad (2)$$

by **Proposition E.2** to simplify the argument of a logarithmic term. By a union bound, the event (*) and the upper bounds (1) and (2) hold with probability $1 - \gamma - 1/n^2$. Furthermore, choosing $s = n^{1/3}/\log^{2/3}(n)$ by balancing the two errors, the upper bounds (1) and (2) ensure,

$$\begin{aligned} \max_{u, v \in V} |d_G(u, v) - \mathbf{d}(u, v)| &\leq O(n^{1/3} \cdot \log^{5/6}(n) \cdot \log(2n^{4/3}/(\gamma \cdot \log^{2/3}(n)))/\varepsilon) \\ &= O(n^{1/3} \cdot \log^{11/6}(n) \cdot \log(1/\gamma)/\varepsilon) \end{aligned}$$

by **Proposition E.2**. □

Next, we present a similar result in the (ε, δ) -DP setting, which is optimal up to a polylogarithmic factor. The proof is analogous and thus deferred to **Appendix B**.

Theorem 3.2. *For $\gamma \in (0, 0.5]$ and $\varepsilon, \delta \in (0, 1]$, there is an (ε, δ) -DP algorithm for computing all-pairs shortest distances that is $O(n^{1/4} \cdot \log^2(n) \cdot \sqrt[4]{\log(2/\delta)} \cdot \log(1/\gamma)/\varepsilon)$ -accurate with probability $1 - \gamma - 1/n^2$.*

4 Improved Algorithms Allowing Multiplicative Error

Chen et al. [CGK⁺23] noted that the known lower bound of $\tilde{\Omega}(n^{1/4})$ for the purely additive error of an algorithm that privately releases all-pairs distances does not apply in the setting where a multiplicative approximation is allowed. In this section, we adapt two existing mechanisms which output shortest distance estimates to the DP setting, which are the first of their kind to have additive error below the lower bound $\tilde{\Omega}(n^{1/4})$ in the purely additive setting, as the current best additive term [CGK⁺23] is $\tilde{O}(k \cdot n^{1/3+2/(9k+3)}/\varepsilon)$. In specific, both of our methods decompose a graph into a collection \mathbf{T} of spanning trees such that for

any pair of vertices $u, v \in V$, there is a tree $\mathcal{T} \in \mathbf{T}$ that satisfies $d_{\mathcal{T}}(u, v) \leq \alpha \cdot d_G(u, v)$ where α is the multiplicative stretch factor. We then turn to an algorithm of privately releasing all-pairs distances in trees with polylogarithmic additive error. Hence, this leads to a mixed multiplicative and additive error in the shortest distance estimates.

The motivation for decomposing a graph into trees comes from a corollary of **Theorem 3.2** originally proven by [Sea16] which extends the algorithm for releasing distances between the root of a tree and every other vertex to all-pairs distances in a tree. We state the result here and defer the proof to **Appendix A**.

Corollary 4.1 (Theorem 4.2 in [Sea16]). *Let $\mathcal{T} = (V, E, \mathbf{w})$ be a tree with $\varepsilon \in (0, 1]$ and $\gamma \in (0, 0.5]$. Then there is an ε -DP algorithm for releasing all-pairs distances on \mathcal{T} that is $O(\log^{1.5}(n) \cdot \log(n/\gamma)/\varepsilon)$ -accurate with probability $1 - \gamma$.*

In what follows we will give two different constructions of algorithms to release all-pairs shortest distances in general graphs with a multiplicative error term, making heavy use of **Corollary 4.1**. **Construction 1** can be considered a *warm-up* for **Construction 2**, as the result from **Construction 2** is stronger, but requires a more in-depth analysis.

4.1 Construction 1: Tree-Padding Spanner

In this section, we give a simple construction of an algorithm with multiplicative error $O(k \log \log n)$ which releases all-pairs distances with differential privacy. This section also serves as a *warm-up* for the next construction.

Abraham et al. [ACE⁺20] gave the first construction of a *tree-padding spanner*, which, given a graph $G = (V, E, \mathbf{w})$, returns a collection of spanning trees \mathbf{T} such that for each vertex $v \in V$, there exists a tree $\mathcal{T} \in \mathbf{T}$ that is a sourcewise $O(k \log \log n)$ -spanner for the source set $\{v\}$, or equivalently, $d_{\mathcal{T}}(v, u) \leq O(k \log \log n) \cdot d_G(v, u)$ for all $u \in V$. We state the result here, but refer the interested reader to [ACE⁺20] for a detailed proof.

Lemma 4.1 (Theorem 2 in [ACE⁺20]). *For a parameter $k \in \mathbb{Z}^+$, there exists an algorithm that finds a collection \mathbf{T} of $k \cdot n^{1/k}$ spanning trees of G such that for every $u, v \in V$, there exists a tree $\mathcal{T} \in \mathbf{T}$ such that $d_{\mathcal{T}}(u, v) \leq O(k \log \log n) \cdot d_G(u, v)$.*

By applying **Corollary 4.1** to each tree in \mathbf{T} , we can release all-pairs distances in the tree-padding spanner, and use these distances to compute an estimate for each pair of vertices in V .

Theorem 4.1. *For $\gamma \in (0, 0.5]$, $k \in \mathbb{Z}^+$, and $\varepsilon \in (0, 1]$, there is an ε -DP algorithm to release all-pairs shortest distances that is $(O(k \log \log n), O(k \cdot n^{1/k} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon))$ -accurate with probability $1 - \gamma$.*

Proof. Let \mathbf{T} be the tree-padding spanner constructed in **Lemma 4.1**. For each $\mathcal{T} \in \mathbf{T}$ and $u, v \in V$, we run the $\varepsilon/(kn^{1/k})$ -DP algorithm given by **Corollary 4.1** to release a private estimate of the distance $d_{\mathcal{T}}(u, v)$, denoted by $\tilde{d}_{\mathcal{T}}(u, v)$. By basic composition (**Lemma 2.1**), releasing all the distances is ε -DP. Moreover, for each estimate $\tilde{d}_{\mathcal{T}}(u, v)$, the result of **Corollary 4.1** ensures, with probability $1 - \gamma$ by a union bound over all $kn^{1/k}$ trees,

$$\begin{aligned} \max_{u, v \in V} |d_{\mathcal{T}}(u, v) - \tilde{d}_{\mathcal{T}}(u, v)| &\leq O(k \cdot n^{1/k} \cdot \log^{1.5}(n) \cdot \log(n/(\gamma/kn^{1/k}))/\varepsilon) \\ &= O(k \cdot n^{1/k} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon) \end{aligned}$$

where the equality follows from **Proposition E.2**. For each $u, v \in V$, by letting $\tilde{d}(u, v) \leftarrow \min_{\mathcal{T} \in \mathbf{T}} \tilde{d}_{\mathcal{T}}(u, v)$, the result of **Lemma 4.1** guarantees that the stretch of the estimate is at most $O(k \log \log n)$. Hence, the total algorithm is $(O(k \log \log n), O(k \cdot n^{1/k} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon))$ -accurate. \square

We also provide a similar (ε, δ) -DP algorithm, and defer the proof to **Appendix C**.

Theorem 4.2. *For any $\gamma \in (0, 0.5]$, $k \in \mathbb{Z}^+$ and $\varepsilon, \delta \in (0, 1]$, there is an (ε, δ) -DP algorithm to release all-pairs shortest distances that is $(O(k \log \log n), O(\sqrt{k} \cdot n^{1/(2k)} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \sqrt{\log(2/\delta)} \cdot \log(1/\gamma)/\varepsilon))$ -accurate with probability $1 - \gamma$.*

4.2 Construction 2: A Cluster-Packing Distance Oracle

In this section, we give an algorithm that modifies the oracle of Thorup and Zwick [TZ05] in order to apply **Corollary 4.1** effectively. The original oracle constructs a total of $O(n^{1-1/k})$ sets of vertices called *clusters*, and for each cluster computes a shortest path tree spanning it. The shortest path distances within each cluster are then used to output the distance approximations. Applying **Corollary 4.1** directly to all $O(n^{1-1/k})$ trees would result in $\tilde{O}(n^{1-1/k}/\varepsilon)$ additive error, which is clearly worse than our result from the previous section. Instead, we modify the sampling process to *pack* multiple clusters into a single spanning tree of G . Namely, we add the edges from the shortest path tree spanning each cluster to a single graph to form a spanning tree of G . This allows us to bound the number of times we need to call the algorithm from **Corollary 4.1** to be $\tilde{O}(k \cdot n^{1/k})$. In the original distance oracle, this is not possible as the clusters can share common vertices with one another, so adding their shortest path spanning trees to a single graph may create cycles.

Throughout this section, for a subset of vertices $S \subseteq V$ and $v \in V$, we let $d(v, S) = \min_{u \in S} d(v, u)$. Furthermore, for an integer parameter $i \in \mathbb{Z}^+$, we let $[i] := \{1, 2, \dots, i\}$. Lastly, for a graph $H = (V_H, E_H)$, we let $E(H) = E_H$.

Our algorithm (**Algorithm 2**) begins by constructing the sets $V = A_0 \supseteq A_1 \supseteq \dots \supseteq A_{k-1}$, but starting from A_{k-1} to A_1 in a backward manner. In particular, the algorithm runs for $t := \lceil 100 \cdot n^{1/k} \log(n) \rceil$ iterations, and we sample a portion of the vertices in each set A_i in each iteration. For each $r \in [t]$, we begin by letting A_{k-1}^r be a single vertex sampled at random from V . Then, for each $i \in [k-2]$, we let A_i^r be A_{i+1}^r , and sample $n^{1-(i+1)/k} - n^{1-(i+2)/k}$ vertices randomly from the set $V - A_{i+1}^r$ to add to the set A_i^r . Lastly, we let $A_i = \cup_r A_i^r$ for each $i \in [k-1]$.

Moreover, for every $w \in A_i^r$, we let $C^r(w) = \{u \in V \mid d(w, u) < d(u, A_i^r - \{w\})\}$ be the cluster *centered* at w . We define clusters in this way in order to ensure that the vertices in each cluster with a center in A_i^r are disjoint, so that we can *fit* them into a single spanning tree.

The next step is to define the *pivots* and *bunches* for every vertex $u \in V$, which we do in the same way as [TZ05], except for our modified sets A_i . Namely, for each $i \in [k-1]$, we let $p_i(u) \in A_i$ be a vertex such that $d_G(u, p_i(u)) = d_G(u, A_i)$, where ties are broken arbitrarily. In particular, if $d(u, A_i) = d(u, A_{i+1})$, then $p_i(u) \leftarrow p_{i+1}(u)$. Furthermore, we define the bunches as $B(u) = \{w \in V \mid d(w, u) < d(u, p_{i+1}(u))\}$.

We make the following observations about our construction which will be critical to our later analysis. Throughout our analysis, we assume that $2 \leq k \leq \lfloor \log(n)/\log(2) \rfloor$ such that $n^{1/k} \geq 2$ for simplicity. At this end of this section, we will justify this assumption.

Our first observation establishes that each cluster is disjoint by construction.

Lemma 4.2. *For all $r \in [t]$, $i \in [k-1]$, and $w_1, w_2 \in A_i^r$ with $w_1 \neq w_2$, $C^r(w_1) \cap C^r(w_2) = \emptyset$.*

Proof. By way of contradiction, suppose $C^r(w_1) \cap C^r(w_2) \neq \emptyset$. Then, there exists a vertex $u \in C^r(w_1) \cap C^r(w_2)$. Let a_1 be the closest vertex to u in $A_i^r - \{w_1\}$ and a_2 be the closest vertex to u in $A_i^r - \{w_2\}$. Then, by our definition of clusters,

$$d(w_1, u) < d(u, a_1) \leq d(u, w_2) < d(u, a_2) \leq d(u, w_1)$$

Hence no such vertex u can exist. \square

Next, we make a key observation that no one vertex has a higher probability of being included in the sets A_i^r or A_i than any other vertex.

Lemma 4.3. *Each vertex $u \in V$ has a uniform probability of being included in A_i^r and A_i for $i \in [k-1]$ and $r \in [t]$.*

Proof. By **Line 15**, the set A_i is simply the union of the sets A_i^r over all values of $r \in [t]$. Since the sampling procedure in each of the t iterations is independent with respect to every other iteration, if each vertex has an equal probability of being included in A_i^r , then each vertex has an equal probability of being included in A_i .

Algorithm 2 Differentially private distance oracle

Input: Graph $G = (V, E, \mathbf{w})$, parameter $k \in \mathbb{Z}^+ - \{1\}$, $\varepsilon \in (0, 1]$, and $\gamma \in (0, 0.5]$

Output: ε -DP distance estimates $(\tilde{d}(u, v))_{u, v \in V}$

```

1: Set  $t := \lceil 100 \cdot n^{1/k} \log(n) \rceil$  and  $\mathbf{T} = \emptyset$ 
2: for  $r \leftarrow 1, 2, \dots, t$  do
3:   Sample  $v \in V$  randomly and let  $A_{k-1}^r = \{v\}$  and  $C^r(v) = V$ . Let  $\mathcal{T}_{k-1}^r$  be the shortest path tree grown from  $v$ .
4:   for  $i \leftarrow k-2, \dots, 1$  do
5:      $A_i^r \leftarrow A_{i+1}^r$ 
6:     Initialize  $\mathcal{T}_i^r = (V, \emptyset)$ 
7:     Sample  $n^{1-(i+1)/k} - n^{1-(i+2)/k}$  vertices randomly from  $V - A_{i+1}^r$  and add them to  $A_i^r$ 
8:     for  $w \in A_i^r$  do
9:       Let  $C^r(w) = \{u \in V \mid d(w, u) < d(u, A_i^r - \{w\})\}$ 
10:    end for
11:    Compute the shortest path tree from  $w$  spanning  $C^r(w)$  and add the edges to  $\mathcal{T}_i^r$ .
12:  end for
13:  For each  $i \in [k-1]$  and  $r \in [t]$ , connect the components of  $\mathcal{T}_i^r$  into a single spanning tree using the modified version of Kruskal's algorithm. Then, let  $\mathbf{T} \leftarrow \mathbf{T} \cup \mathcal{T}_i^r$ .
14: end for
15: Let  $A_i = \cup_r A_i^r$  for  $i \in [k-1]$ , and  $A_k = \emptyset$ 
16: Let  $p_i(u) \in A_i$  be the nearest to  $v$  for every  $v \in V$ ,  $i \in [k-1]$ . If  $d(u, p_{i+1}(u)) = d(u, A_i)$ , then  $p_i(u) \leftarrow p_{i+1}(u)$ 
17: Let  $B(v) = \cup_{i \in [k-1]} \{w \in A_i - A_{i+1} \mid d(w, v) < d(A_{i+1}, v)\}$  for every  $v \in V$ 
18: For each  $\mathcal{T} \in \mathbf{T}$  and  $u, v \in V$ , use the algorithm from Corollary 4.1 to compute a private estimate of  $d_{\mathcal{T}}(u, v)$  with  $\varepsilon/(t \cdot k)$  privacy, denoted by  $\tilde{d}_{\mathcal{T}}(u, v)$ 
19: for  $u, v \in V$  do
20:   Let  $\tilde{d}(u, v) \leftarrow \min_{\mathcal{T} \in \mathbf{T}} \tilde{d}_{\mathcal{T}}(u, v)$ 
21: end for
22: return  $(\tilde{d}(u, v))_{u, v \in V}$ 

```

Therefore, it suffices to show that $\mathbb{P}[u \in A_i^r] = |A_i^r|/n = n^{-(i+1)/k}$. We prove the claim by induction on i . For the base case of $i = k-1$, by **Line 3**, A_{k-1}^r consists of a single vertex picked at random, so each vertex has probability $1/n$ of being picked. Now, suppose that $1 \leq i < k-1$ and the claim holds for index $i+1$. By **Lines 5** and **7**, $\mathbb{P}[u \in A_i^r] = \mathbb{P}[u \in A_{i+1}^r] + \mathbb{P}[u \notin A_{i+1}^r] \cdot \mathbb{P}[u \in A_i^r - A_{i+1}^r]$. By the induction hypothesis, $\mathbb{P}[u \in A_{i+1}^r] = n^{-(i+2)/k}$, hence,

$$\mathbb{P}[u \in A_i^r] = n^{-(i+2)/k} + (1 - n^{-(i+2)/k}) \cdot \frac{n^{1-(i+1)/k} - n^{1-(i+2)/k}}{n - n^{1-(i+2)/k}} = n^{-(i+1)/k}$$

□

In our sampling procedure, the sets A_i^r are sampled without replacement, however, a single vertex can be sampled at the i^{th} level in more than one iteration $r \in [t]$. Consequently, $|A_i| \leq t \cdot |A_i^r|$, so for our analysis we must give a lower bound on the size of A_i , which we do next.

Lemma 4.4. $|A_i| \geq (t/4) \cdot n^{1-(i+1)/k}$ for all $i \in [k-1]$ with probability at least $1 - 1/(2n^2)$.

Proof. By **Line 15**, for $i \in [k-1]$, A_i is the union of the sets A_i^r over $r \in [t]$, each of which is sampled uniformly and independently from V without replacement by **Lemma 4.3** and has size $n^{1-(i+1)/k}$. In particular, if we fix $u \in V$, then $\mathbb{P}[u \in A_i^r] = |A_i^r|/n$. Hence, the probability that u is included in A_i after t iterations is $1 - (1 - |A_i^r|/n)^t = p_i$. Let $X_i \sim \text{Binomial}(n, p_i)$ be the number of vertices in A_i . It follows that,

$$\begin{aligned}
\mathbb{E}[X_i] &= n \cdot (1 - (1 - |A_i^r|/n)^t) \\
&\geq n \cdot \left(1 - \frac{1}{1 + |A_i^r|t/n}\right) \\
&= \frac{t}{1 + |A_i^r|t/n} \cdot |A_i^r| \\
&\geq (t/2) \cdot |A_i^r| = (t/2) \cdot n^{1-(i+1)/k}
\end{aligned}$$

where the first inequality follows from Bernoulli's inequality, and the last inequality follows from $|A_i^r| \cdot t \leq n$.

Furthermore, by a Chernoff bound (**Lemma 2.5** with $\lambda = \mathbb{E}[X_i]/2$), we have that $\mathbb{P}[X_i \leq \mathbb{E}[X_i]/2] \leq e^{-\mathbb{E}[X_i]/8}$. The expectation of X_i , $\mathbb{E}[X_i]$ is minimized when $i = k - 1$, as we sample the least amount of vertices in A_{k-1} . So, $\mathbb{E}[X_i] \geq \mathbb{E}[X_{k-1}] \geq t/2 = 50 \cdot n^{1/k} \log n$. Hence, we have that

$$\begin{aligned} \mathbb{P}[X_i \leq \mathbb{E}[X_i]/2] &\leq e^{-\mathbb{E}[X_i]/8} \\ &\leq \exp(-(50/8) \cdot n^{1/k} \log n) \\ &= n^{-(50/8)n^{1/k}} \leq 1/(2n^3) \end{aligned}$$

Since the assumption $k \leq \lfloor \log(n)/\log(2) \rfloor$ implies $k \leq n$, by a union bound we get that the event $|A_i| \geq (t/4) \cdot n^{1-(i+1)/k}$ for all $1 \leq i \leq k-1$ occurs with probability at least $1 - 1/(2n^2)$. \square

Finally, we make the key observation that our sets A_{i+1} are large enough relative to the size of A_i^r such that every vertex in a cluster $C^r(w)$ closer to the set A_{i+1} than $A_i^r - \{w\}$ for every $i \in [k-2]$, $r \in [t]$.

Lemma 4.5. *For all $i \in [k-2]$, $r \in [t]$, $u \in V$, if $w \in A_i^r$ is the closest to u in G , then $d(u, p_{i+1}(u)) \leq d(u, A_i^r - \{w\})$ with probability at least $1 - 1/n^2$.*

Proof. Let $\mathcal{O}_u : V \rightarrow [n]$ be the ordering of vertices V by sorting vertices based on their distance to vertex u in G , with ties broken arbitrarily. For example, the j^{th} -closest vertex to u in G maps to j . We note $d(u, p_{i+1}(u)) \leq d(u, A_i^r - \{w\})$ if at least one vertex in A_{i+1} comes before all the vertices in $A_i^r - \{w\}$ in this ordering. Suppose that for a vertex $v \in A_{i+1}$ we have $\mathcal{O}_u(v) = j$. By **Lemma 4.3**, since each vertex has an equal probability of being included in the set A_i^r , the probability that all the vertices in $A_i^r - \{w\}$ come after v , or $\mathcal{O}_u(y) > \mathcal{O}_u(v)$ for all $y \in A_i^r - \{w\}$, is $(1 - j/n)^{|A_i^r|-1}$. Also by **Lemma 4.3**, the probability that $\mathcal{O}_u(v) = j$ is $1/n$ for every $j \in [n]$. Since the events $\mathcal{O}_u(v) = j$ and $\mathcal{O}_u(v) = m$ are clearly disjoint when $j \neq m$, we have that,

$$\begin{aligned} \mathbb{P}[d(u, v) \leq d(u, A_i^r - \{w\})] &= \mathbb{P}\left[\bigcup_{j=1}^n (\mathcal{O}_u(v) = j) \cap (\mathcal{O}_u(y) > j \text{ for all } y \in A_i^r - \{w\})\right] \\ &= \sum_{j=1}^n (1 - j/n)^{|A_i^r|-1} (1/n) \\ &\geq \int_{1/n}^1 (1 - x)^{|A_i^r|-1} dx \\ &= \frac{(1 - 1/n)^{|A_i^r|}}{|A_i^r|} \\ &\geq \frac{1 - (|A_i^r|/n)}{|A_i^r|} \geq \frac{1}{2|A_i^r|} \end{aligned}$$

where the last inequality follows from $|A_i^r|/n \leq n^{-2/k} \leq n^{-1/k} \leq \frac{1}{2}$. We note that $|A_i^r| = n^{1-(i+1)/k}$ and by **Lemma 4.4** we condition on the event that $|A_{i+1}| \geq (t/4) \cdot n^{1-(i+2)/k}$ for all $i \in [k-2]$ with probability at least $1 - 1/(2n^2)$. Hence, the probability that at least one of the vertices in A_{i+1} comes before $A_i^r - \{w\}$ in the ordering \mathcal{O}_u is at least,

$$\begin{aligned} \mathbb{P}[d(u, p_{i+1}(u)) \leq d(u, A_i^r - \{w\})] &\geq 1 - \left(1 - \frac{1}{2|A_i^r|}\right)^{|A_{i+1}|} \\ &\geq 1 - \left(1 - \frac{1}{2|A_i^r|}\right)^{(t/4) \cdot n^{1-(i+2)/k}} \\ &\geq 1 - n^{-7} \end{aligned}$$

where the last inequality follows from **Proposition E.1**. We condition on the event $d(u, p_{i+1}(u)) \leq d(u, A_i^r - \{w\})$ for all $u \in V$, for all levels 1 through $k-2$, for all t iterations with probability at least $1 - n \cdot t \cdot k/n^7$,

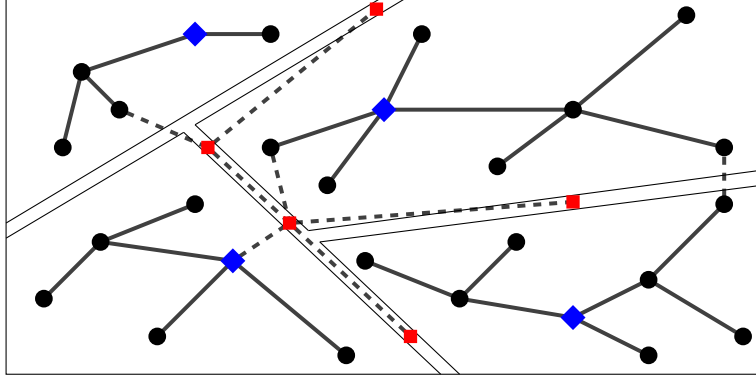


Figure 1: Visualization of a tree \mathcal{T}_i^r . The blue (diamond) vertices denote the centers of clusters, and the solid edges are the minimum spanning trees that span each cluster. The red (square) vertices do not belong to any cluster, and the dashed edges connect the clusters into a single spanning tree.

by a union bound. Furthermore, $t \cdot k \leq \frac{100}{\log(2)} \cdot n^{1/k} \cdot \log^2(n) \leq n^3$ for all n sufficiently large, where the first inequality holds from $k \leq \lfloor \log(n)/\log(2) \rfloor$. Hence, $t \cdot k/n^6 \leq 1/n^3 \leq 1/(2n^2)$, so the probability of success is bounded below by $1 - 1/(2n^2)$. By a union bound with **Lemma 4.4**, $d(u, p_{i+1}(u)) \leq d(u, A_i^r - \{w\})$ occurs with probability at least $1 - 1/n^2$. \square

The next step in our algorithm is to form a collection \mathbf{T} of trees, the distances in which will be used to compute the final distance estimates as in the construction from the previous section. In particular, to construct $\mathcal{T}_i^r \in \mathbf{T}$, we begin by letting $\mathcal{T}_i^r = (V, \emptyset)$. Then, for each center $w \in A_i^r$, we add the edges of the shortest path tree centered at w and spanning $C^r(w)$ to \mathcal{T}_i^r . By **Lemma 4.2**, \mathcal{T}_i^r is a forest. To connect the components of the forest into a single spanning tree of G , we imagine *contracting* each cluster into its center and adding the vertices of the minimum spanning tree of the new graph to \mathcal{T}_i^r .

In fact, we do not utilize the distances *between* clusters in each level, so any arbitrary connection of the components such that no cycles are formed is adequate. To give an explicit algorithm for connecting the clusters into a single spanning tree, we give a simple modification of Kruskal's algorithm [Kru56]. In specific, let E' be the set of edges $E(G) - E(\mathcal{T}_i^r)$. We (1) pick the edge in E' with the smallest weight, with ties broken arbitrarily, and check if it forms a cycle in \mathcal{T}_i^r . If it doesn't, we add the edge to \mathcal{T}_i^r . Then, we remove the edge from E' . We (2) repeat step (1) until \mathcal{T}_i^r has $n - 1$ edges. By the condition in step (1) and **Lemma 4.2**, \mathcal{T}_i^r is acyclic. Furthermore, \mathcal{T}_i^r cannot be disconnected as the graph G is connected and the first edge encountered in E' that connects two disconnected components in \mathcal{T}_i^r must be added to \mathcal{T}_i^r as it won't form a cycle.

Now we let $\mathbf{T} = \cup_i \cup_r \mathcal{T}_i^r$. In this way, we construct exactly one spanning tree for each set A_i^r , hence there are $t \cdot k = 100 \cdot kn^{1/k} \log(n)$ total trees in \mathbf{T} . Next, we analyze the maximum *stretch* of the distance estimates computed in \mathbf{T} , which parallels the analysis in [TZ05]. We begin by showing that the pivot of a vertex is always included in its bunch.

Lemma 4.6. *For all $i \in [k - 1]$ and $u \in V$, $p_i(u) \in B(u)$.*

Proof. The claim follows by induction on the index i . For the base case of $i = k - 1$, since $A_k = \emptyset$, for all $w \in A_{k-1}$, $d(w, A_k) = \infty$. Consequently, $d(w, u) < d(w, A_k)$, so $w \in B(u)$. It follows that $p_{k-1}(u) \in A_{k-1} \subseteq B(u)$. Suppose further that $i < k - 1$ and $p_{i+1}(u) \in B(u)$. There are two cases. In the first case, $d(u, A_i) = d(u, A_{i+1})$, and $p_i(u) \leftarrow p_{i+1}(u) \in B(u)$ by **Line 16**. In the second case, $d(u, A_i) = d(u, p_i(u)) < d(u, A_{i+1})$, and hence $p_i(u) \in B(u)$ by the definition of bunches. \square

For a pair of vertices $u, v \in V$, to compute a distance estimate $\tilde{d}(u, v)$, we first introduce a placeholder variable $w = u$ and initially set $i = 0$. If $w \in B(v)$, then the algorithm stops. Otherwise, we let $i \leftarrow i + 1$, swap u and v , and let $w \leftarrow p_i(u) \in A_i$. We repeat this process of checking if $w \in B(v)$ and swapping until the algorithm stops. We note that the algorithm is guaranteed to stop as $A_{k-1} \subseteq B(v)$ for every vertex

$v \in V$, as the cluster constructed for each vertex in A_{k-1} in **Line 3** of **Algorithm 2** is simply the entire vertex set. Once the swapping algorithm stops, we let $\mathbf{d}(u, v) = d(w, u) + d(w, v)$.

It remains to show that the swapping procedure outlined above indeed releases an estimate with stretch at most $(2k - 1)$, which we will prove next.

Lemma 4.7. $\mathbf{d}(u, v) \leq (2k - 1) \cdot d_G(u, v)$.

Proof. Let $\Delta = d_G(u, v)$. Initially, we set $w = u$, hence $d(w, u) = 0$. We will show that each iteration of the swapping procedure increases $d(w, u)$ by at most Δ . Let u_i, v_i , and w_i be the values of the variables u, v , and w respectively in iteration $i \in [k - 1]$ of the algorithm. If the algorithm does not terminate at iteration $i - 1$, then $w_{i-1} \notin B(v_{i-1})$, so $d(w_{i-1}, v_{i-1}) \geq d(A_i, v_{i-1}) = d(p_i(v_{i-1}), v_{i-1})$. However, since $v_{i-1} = u_i$ and $w_i = p_i(u_i)$, we have that,

$$d(w_i, u_i) = d(p_i(u_i), u_i) = d(p_i(v_{i-1}), v_{i-1}) \leq d(w_{i-1}, v_{i-1}) \leq d(w_{i-1}, u_{i-1}) + \Delta$$

Since $A_{k-1} \subseteq B(v)$, there are at most $k-1$ iterations in the algorithm, hence $d(w, u) \leq (k-1)\Delta$. Furthermore, $d(w, v) \leq d(w, u) + d(u, v) \leq (k-1)\Delta + \Delta = k\Delta$ by the triangle inequality. It follows that $\mathbf{d}(u, v) = d(w, u) + d(w, v) \leq (2k-1)\Delta$. \square

We can now apply **Corollary 4.1** to the set \mathbf{T} of trees as in the previous section.

Theorem 4.3. For $\gamma \in (0, 0.5]$, $\varepsilon \in (0, 1]$, and $k \in \{2, \dots, \lfloor \log(n)/\log(2) \rfloor\}$, there is an ε -DP algorithm for computing all-pairs shortest distances that is $(2k - 1, O(k \cdot n^{1/k} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon))$ -accurate with probability $1 - \gamma - 1/n^2$.

Proof. For each $\mathcal{T} \in \mathbf{T}$ and $u, v \in V$, we run the $\varepsilon/(t \cdot k)$ -DP algorithm given by **Corollary 4.1** to release the private distance estimates $\tilde{d}_{\mathcal{T}}(u, v)$ and let $\tilde{d}(u, v) \leftarrow \min_{\mathcal{T} \in \mathbf{T}} \tilde{d}_{\mathcal{T}}(u, v)$. By basic composition (**Lemma 2.1**), releasing all the private estimates is ε -DP. Moreover, for each $\tilde{d}_{\mathcal{T}}(u, v)$, the result of **Corollary 4.1** guarantees that, with probability $1 - \gamma$ by a union bound over all $t \cdot k$ trees,

$$\begin{aligned} \max_{u, v \in V} |d_{\mathcal{T}}(u, v) - \tilde{d}_{\mathcal{T}}(u, v)| &\leq O(k \cdot n^{1/k} \cdot \log^{1.5}(n) \cdot \log(n \cdot t \cdot k/\gamma)/\varepsilon) \\ &= O(k \cdot n^{1/k} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon) \end{aligned} \quad (3)$$

by **Proposition E.2**. Moreover, we note that for our distance estimate $\mathbf{d}(u, v) = d(w, u) + d(w, v)$, $w \in A_l^m$ for some $m \in [t]$ and $l \in [k - 1]$. Furthermore, $w \in B(v)$ as it is a condition for stopping, and $w \in B(u)$ by **Lemma 4.6**. $w \in B(u)$ implies $d(w, u) < d(u, p_{i+1}(u)) \leq d(u, A_i^r - \{w\})$ with probability $1 - 1/n^2$ by **Lemma 4.5**, and consequently $u \in C^m(w)$. By equivalent logic, $v \in C^m(w)$. It follows that, with probability $1 - 1/n^2$, for any pair of vertices $u, v \in V$ there is some tree $\mathcal{T}^* \in \mathbf{T}$ such that $\mathbf{d}(u, v) = d(w, u) + d(w, v) = d_{\mathcal{T}^*}(w, u) + d_{\mathcal{T}^*}(w, v)$. Hence, by **Lemma 4.7**,

$$\min_{\mathcal{T} \in \mathbf{T}} d_{\mathcal{T}}(u, v) \leq (2k - 1) \cdot d_G(u, v) \quad (4)$$

By a union bound, the upper bounds (3) and (4) hold with probability $1 - \gamma - 1/n^2$, and ensure that,

$$\begin{aligned} d_G(u, v) - O(k \cdot n^{1/k} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon) &\leq \tilde{d}(u, v) \\ &\leq (2k - 1) \cdot d_G(u, v) + O(k \cdot n^{1/k} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon) \end{aligned}$$

Consequently, the algorithm is $(2k - 1, O(k \cdot n^{1/k} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon))$ -accurate, as claimed. \square

The proof of the following theorem is analogous to the proof of **Theorem 4.3**, and hence is deferred to **Appendix D**.

Theorem 4.4. For $\gamma \in (0, 0.5]$, $\varepsilon, \delta \in (0, 1]$, and $k \in \{2, \dots, \lfloor \log(n)/\log(2) \rfloor\}$, there is an (ε, δ) -DP algorithm for computing all-pairs shortest distances that is $(2k - 1, O(\sqrt{k} \cdot n^{1/(2k)} \cdot \log^{2.5}(n) \cdot \sqrt{\log(2/\delta)} \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon))$ -accurate with probability $1 - \gamma - 1/n^2$.

Finally, we give a short justification as to why our assumption that $k \leq \lfloor \log(n)/\log(2) \rfloor$ is reasonable. When $k = \lfloor \log(n)/\log(2) \rfloor$, both multiplicative and additive errors become $O(\text{polylog}(n))$. As such, further increasing the value of k would only increase the multiplicative error.

5 Discussion and Open Problems

In this work, we give an algorithm to release all-pairs shortest distances for general graphs with $\tilde{O}(n^{1/3}/\varepsilon)$ additive error in the ε -DP setting and $\tilde{O}(n^{1/4}/\varepsilon)$ additive error in the (ε, δ) -DP setting, which is optimal up to a polylogarithmic factor based on the lower bound of $\Omega(n^{1/4}/\sqrt{\log n})$ proven by [BDG⁺24]. In the case with the added assumption that the edge weights are in $(0, M]$ for some $M \in \mathbb{R}^+$, [CGK⁺23] showed that there is a constant $c > 0$ such that no (ε, δ) -DP algorithm for APSD can be $o(n^c)$ -accurate, even when $M = O(1)$. That is, one natural open question is whether an algorithm with error $O(n^{1/4-\Omega(1)})$ is possible in the bounded edge weights setting.

Furthermore, we give algorithms on general graphs that have additive error whose polynomial dependence on n beats the known lower bound for purely additive algorithms if one allows a multiplicative approximation. The question remains open that whether an algorithm with an $O(1)$ multiplicative approximation and $O(\text{polylog}(n))$ additive approximation is possible.

6 Acknowledgements

This material is based upon work supported by the National Science Foundation Grant CNS-2349369. This work was completed during the UNCG GraLNA 2024 REU, where Jesse Campbell was working under the mentorship of Chunjiang Zhu.

References

- [ABC⁺20] Ahmet Aktay, Shailesh Bavadekar, Gwen Cossoul, John Davis, Damien Desfontaines, Alex Fabrikant, Evgeniy Gabrilovich, Krishna Gadepalli, Bryant Gipson, Miguel Guevara, Chaitanya Kamath, Mansi Kansal, Ali Lange, Chinmoy Mandayam, Andrew Oplinger, Christopher Plunkte, Thomas Roessler, Arran Schlosberg, Tomer Shekel, Swapnil Vispute, Mia Vu, Gregory Wellenius, Brian Williams, and Royce J Wilson. Google covid-19 community mobility reports: Anonymization process description (version 1.1). *ArXiv preprint*, arXiv:2004.04145, 2020.
- [ACE⁺20] Ittai Abraham, Shiri Chechik, Michael Elkin, Arnold Filtser, and Ofer Neiman. Ramsey spanning trees and their applications. *ACM Trans. Algorithms*, 16(2), 2020.
- [BBD⁺21] Shailesh Bavadekar, Adam Boulanger, John Davis, Damien Desfontaines, Evgeniy Gabrilovich, Krishna Gadepalli, Badih Ghazi, Tague Griffith, Jai Gupta, Chaitanya Kamath, Dennis Kraft, Ravi Kumar, Akim Kumok, Yael Mayer, Pasin Manurangsi, Arti Patankar, Irippuge Milinda Perera, Chris Scott, Tomer Shekel, Benjamin Miller, Karen Smith, Charlotte Stanton, Mimi Sun, Mark Young, and Gregory Wellenius. Google covid-19 vaccination search insights: Anonymization process description. *ArXiv preprint*, arXiv:2107.01179, 2021.
- [BDD⁺20] Shailesh Bavadekar, Andrew Dai, John Davis, Damien Desfontaines, Ilya Eckstein, Katie Everett, Alex Fabrikant, Gerardo Flores, Evgeniy Gabrilovich, Krishna Gadepalli, Shane Glass, Rayman Huang, Chaitanya Kamath, Dennis Kraft, Akim Kumok, Hinali Marfatia, Yael Mayer, Benjamin Miller, Adam Pearce, Irippuge Milinda Perera, Venky Ramachandran, Karthik Ramman, Thomas Roessler, Izhak Shafran, Tomer Shekel, Charlotte Stanton, Jacob Stimes, Mimi Sun, Gregory Wellenius, and Masrour Zoghi. Google covid-19 search trends symptoms dataset: Anonymization process description (version 1.0). *ArXiv preprint*, arXiv:2009.01265, 2020.
- [BDG⁺24] Greg Bodwin, Chengyuan Deng, Jie Gao, Gary Hoppenworth, Jalaj Upadhyay, and Chen Wang. The discrepancy of shortest paths. *ArXiv preprint*, arXiv:2401.15781, 2024.
- [CGK⁺23] Justin Y. Chen, Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Shyam Narayanan, Jelani Nelson, and Yinzhan Xu. Differentially private all-pairs shortest path distances: Improved algorithms and lower bounds. *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5040–5067, 2023.

- [CL01] B. Chazelle and A. Lvov. A trace bound for the hereditary discrepancy. *Discrete & Computational Geometry*, 26(2):221–231, 2001.
- [CL06] Fan Chung and Linyuan Lu. Concentration inequalities and martingale inequalities: a survey. *Internet Mathematics*, 3(1):79 – 127, 2006.
- [CSC⁺24] Bin Cai, Weihong Sheng, Jiajun Chen, Chunqiang Hu, and Jiguo Yu. Shortest paths publishing with differential privacy. *IEEE Transactions on Sustainable Computing*, 9(2):209–221, 2024.
- [CSS11] Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3), 2011.
- [DGUW23] Chengyuan Deng, Jie Gao, Jalaj Upadhyay, and Chen Wang. Differentially private range query on shortest paths. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures - 18th International Symposium, WADS 2023, Proceedings*, pages 340–370, Germany, 2023. Springer Science and Business Media Deutschland GmbH.
- [DKY17] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [DL09] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, STOC '09*, page 371–380, New York, NY, USA, 2009. Association for Computing Machinery.
- [DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.
- [ELRS23] Talya Eden, Quanquan C. Liu, Sofya Raskhodnikova, and Adam Smith. Triangle counting with local edge differential privacy. *ArXiv preprint*, arXiv:2305.02263, 2023.
- [EMK23] Javad B. Ebrahimi, Alireza Tofghi Mohammadi, and Fatemeh Kermani. Differentially private all-pairs shortest distances for low tree-width graphs. In *2023 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, 2023.
- [FL22] Chenglin Fan and Ping Li. Distances release with differential privacy in tree and grid graph. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 2190–2195, 2022.
- [FLL22] Chenglin Fan, Ping Li, and Xiaoyun Li. Breaking the linear error barrier in differentially private graph distance release. *NeurIPS*, 2022.
- [FMM19] Andrew David Foote, Ashwin Machanavajjhala, and Kevin McKinney. Releasing earnings distributions using differential privacy: Disclosure avoidance system for post-secondary employment outcomes (pseo). *Journal of Privacy and Confidentiality*, 9(2), 2019.
- [GAP18] Simson L. Garfinkel, John M. Abowd, and Sarah Powazek. Issues encountered deploying differential privacy. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society, WPES'18*, page 133–137, New York, NY, USA, 2018. Association for Computing Machinery.
- [HCYH24] Calvin Hawkins, Bo Chen, Kasra Yazdani, and Matthew Hale. Node and edge differential privacy for graph laplacian spectra: Mechanisms and scaling laws. *IEEE Transactions on Network Science and Engineering*, 11(2):1690–1701, 2024.

- [HLMJ09] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178, 2009.
- [JWC23] Xun Jian, Yue Wang, and Lei Chen. Publishing graphs under node differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):4164–4177, 2023.
- [Kru56] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.
- [LLXL21] Wenfen Liu, Bixia Liu, Qiang Xu, and Hui Lei. Graph node strength histogram publication method with node differential privacy. *Journal of Physics: Conference Series*, 1757(1):012186, 2021.
- [LML20] Ganghong Liu, Xuebin Ma, and Wuyungerile Li. Publishing node strength distribution with node differential privacy. *IEEE Access*, 8:217642–217650, 2020.
- [Sea16] Adam Sealfon. Shortest paths and distances with differential privacy. PODS ’16, page 29–41, New York, NY, USA, 2016. Association for Computing Machinery.
- [SU21] Adam Sealfon and Jonathan Ullman. Efficiently estimating erdos-renyi graphs with node differential privacy. *Journal of Privacy and Confidentiality*, 11(1), 2021.
- [TZ05] M. Thorup and U. Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, 2005.
- [Wan15] Hua Wang. Centroid, leaf-centroid, and internal-centroid. *Graphs and Combinatorics*, 31(3):783–793, 2015.
- [WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 645–654, 2010.
- [ZNF23] Sen Zhang, Wei-Wei Ni, and Nan Fu. Community-preserving social graph release with node differential privacy. *Journal of Computer Science and Technology*, 38(6):1369–1386, 2023.

A Proof of Lemma 3.2 and Corollary 4.1 on APSD in trees

The proofs presented are slightly modified from [Sea16] and are given for completeness. In particular, in the proof of **Lemma 3.2** instead of showing that, with probability $1 - \gamma$, any *single* released distance is bounded above by some term, we show that with probability $1 - \gamma$, *all* the released distances are bounded above by some term. We do this by letting the probability of failure for releasing any single distance to be γ/n such that, by a union bound, the total failure probability over all n released distances is γ . This gives a term $\log(n/\gamma)$ in the final bound instead of $\log(1/\gamma)$ as in the original proof. We note that this slightly changes our proof of **Corollary 4.1**, however it does not affect the final result.

Lemma 3.2. *Let $\mathcal{T}_z = (V, E, \mathbf{w})$ be a tree with root vertex $z \in V$, $\varepsilon \in (0, 1]$, and $\gamma \in (0, 0.5]$. Then there is an ε -DP algorithm for releasing the distance between z and every $u \in V$ that is $O(\log^{1.5}(n) \cdot \log(n/\gamma)/\varepsilon)$ -accurate with probability $1 - \gamma$.*

Proof. Let \mathcal{T}_z be the tree rooted at vertex $z \in V$. The algorithm works by splitting the tree into subtrees, each with fewer than $n/2$ vertices, computing distances between the root and the roots of the subtrees, and then applying the algorithm recursively on each subtree.

It is a well-known result that every tree has either one or two vertices known as *centroids* such that their removal disconnects the tree into subtrees, each with fewer than $n/2$ vertices (see, for example, [Wan15]). If \mathcal{T}_z has a single centroid, we let z^* be that vertex. If \mathcal{T}_z has two centroids, we let z^* be the centroid that is closer to the root node z . By choosing it in this way, we guarantee that z^* is the unique vertex such that the subtree rooted at z^* has more than $n/2$ vertices, but the subtree rooted at each of z^* ’s children has at most

$n/2$ vertices. Let z_1, \dots, z_t be the children of z^* and $\mathcal{T}_i = (V_i, E_i)$, $i \in \{1, \dots, t\}$, their corresponding subtrees. Additionally, we define \mathcal{T}_0 to be the subtree rooted at z with vertex set $V_0 = V - \{V_1, \dots, V_t\}$.

Next, we release the distances between z and z^* , as well as between z^* and each of its children by adding Laplace noise from the distribution $\text{Lap}(\log(n)/\varepsilon)$. In particular, we note that since each tree is disjoint by construction, the function which releases these distances has sensitivity 1. Hence, by **Lemma 2.3**, releasing each subtree is $(\varepsilon/\log(n))$ -DP. Then, we recursively call this algorithm for each subtree until each subtree consists only of a single vertex.

Since each tree has at most $n/2$ vertices, the maximum recursion depth is bounded by $\log(n)$. Hence, by basic composition (**Lemma 2.1**), since the released distances in our algorithm are a composition of $\log(n)$, $(\varepsilon/\log(n))$ -DP mechanisms, the total algorithm is ε -DP.

It remains to be shown that the error in each released distance is bounded above by $O(\text{polylog}(n))$. We note that for every $u \in V$, the distance $d(z, u)$ is the composition of released distances from root nodes to centroids, and from centroids to their children. By bounding the number of released distances in the composition, we can bound the total error incurred from each noisy random variable.

We will prove by induction on the number of vertices that the number of released distances used to calculate $d(z, u)$ is bounded above by $2 \log(n)$. Firstly, the base case of $n = 1$ is vacuously true. Suppose $n > 1$. In the first iteration of the algorithm, the vertex sets V_0, V_1, \dots, V_t are a partition of V . Hence, $u \in V_i$ for some $i \in [t]$. By the induction hypothesis, we can release the distance from z_i to u by the composition of at most $2 \log(n/2) = 2 \log(n) - 2$ noisy distances. Then, we have that $d(z, z_i) = d(z, z^*) + d(z^*, z_i)$, giving a total of $2 \log(n)$ noisy distances.

It follows that the error in each released distance is the concatenation of $2 \log(n)$ random variables distributed according to $\text{Lap}(\log(n)/\varepsilon)$. By **Lemma 2.4**, with probability at most γ/n , we have that the error exceeds $O(\log^{1.5}(n) \cdot \log(n/\gamma)/\varepsilon)$. Hence, by a union bound, with probability at least $1 - \gamma$, the error for any distance from z to a vertex $u \in V$ is bounded above by $O(\log^{1.5}(n) \cdot \log(n/\gamma)/\varepsilon)$, as desired. \square

Corollary 4.1. *Let $\mathcal{T} = (V, E, \mathbf{w})$ be a tree with $\varepsilon \in (0, 1]$ and $\gamma \in (0, 0.5]$. Then there is an ε -DP algorithm for releasing all-pairs distances on \mathcal{T} that is $O(\log^{1.5}(n) \cdot \log(n/\gamma)/\varepsilon)$ -accurate with probability $1 - \gamma$.*

Proof. Choose a vertex $z \in V$ arbitrarily as a root vertex of \mathcal{T} , and call the rooted tree \mathcal{T}_z . Then, using **Lemma 3.2**, we can privately release the distance from z to every other vertex in \mathcal{T}_z . We will show that these distances suffice to compute the distance for any pair $x, y \in V$ of vertices in V .

Let $v_{x,y} \in V$ be the lowest common ancestor of x and y in the rooted tree \mathcal{T}_z . Then, $d_{\mathcal{T}}(x, y) = d_{\mathcal{T}}(x, v_{x,y}) + d_{\mathcal{T}}(v_{x,y}, y) = d_{\mathcal{T}_z}(z, x) + d_{\mathcal{T}_z}(z, y) - 2d_{\mathcal{T}_z}(z, v_{x,y})$. By the result in **Lemma 3.2**, we condition on the event that for all $w \in V$, $d(z, w)$ is $O(\log^{1.5}(n) \cdot \log(n/\gamma)/\varepsilon)$ -accurate with probability $1 - \gamma$. Since our distance estimate is the sum of four distances from z , the total error is at most four times this, which is still $O(\log^{1.5}(n) \cdot \log(n/\gamma)/\varepsilon)$. \square

B Proof of Theorem 3.2

Theorem 3.2. *For $\gamma \in (0, 0.5]$ and $\varepsilon, \delta \in (0, 1]$, there is an (ε, δ) -DP algorithm for computing all-pairs shortest distances that is $O(n^{1/4} \cdot \log^2(n) \cdot \sqrt[4]{\log(2/\delta)} \cdot \log(1/\gamma)/\varepsilon)$ -accurate with probability $1 - \gamma - 1/n^2$.*

Proof. The algorithm is the same as **Algorithm 1**, except we run the $\varepsilon/(4\sqrt{2s \log(2/\delta)})$ -DP algorithm from **Lemma 3.2** to release the shortest path distances in each tree. Namely, in **Lines 7** and **9** we sample the Laplacian noise from $\text{Lap}(4\sqrt{2|S| \log(2/\delta)} \cdot \log(n)/\varepsilon)$. By advanced composition (**Lemma 2.2**), running the s , $\varepsilon/(4\sqrt{2s \log(2/\delta)})$ -DP mechanisms is $(\varepsilon/2, \delta)$ -DP. By basic composition (**Lemma 2.1**), running the $(\varepsilon/2, \delta)$ -DP mechanism and the $\varepsilon/2$ -DP input perturbation algorithm is (ε, δ) -DP.

We note that the upper bound for the additive error from distance estimates computed via the input perturbation algorithm is the same as in **Theorem 3.1**. Furthermore, by **Lemma 3.2**, with probability $1 - \gamma/2$,

$$\max_{z \in S, u \in V} |d_G(z, u) - \tilde{d}(z, u)| \leq O(\sqrt{s} \cdot \log^{1.5}(n) \cdot \sqrt{\log(2/\delta)} \cdot \log(2sn/\gamma)/\varepsilon) \quad (5)$$

We again condition on the event (*) and the bounds (5) and (2) with probability $1 - \gamma - 1/n^2$. Moreover, in **Line 1** we choose $s = \sqrt{n}/(\log(n) \cdot \sqrt{\log(2/\delta)})$, and hence the upper bounds (5) and (2) ensure,

$$\max_{u,v \in V} |d_G(u,v) - \mathbf{d}(u,v)| \leq O(n^{1/4} \cdot \log^2(n) \cdot \sqrt[4]{\log(2/\delta)} \cdot \log(1/\gamma)/\varepsilon)$$

by **Proposition E.2**. □

C Proof of Theorem 4.2

Theorem 4.2. *For any $\gamma \in (0, 0.5]$, $k \in \mathbb{Z}^+$ and $\varepsilon, \delta \in (0, 1]$, there is an (ε, δ) -DP algorithm to release all-pairs shortest distances that is $(O(k \log \log n), O(\sqrt{k} \cdot n^{1/(2k)} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \sqrt{\log(2/\delta)} \cdot \log(1/\gamma)/\varepsilon))$ -accurate with probability $1 - \gamma$.*

Proof. The proof is identical to the proof of **Theorem 4.1**, except we run the $\varepsilon/(2\sqrt{2(kn^{1/k}) \log(2/\delta)})$ -DP algorithm from **Corollary 4.1** to release APSD for each tree $\mathcal{T} \in \mathbf{T}$. By advanced composition (**Lemma 2.2**), releasing all the distances is (ε, δ) -DP. Moreover, the result of **Corollary 4.1** ensures that with probability $1 - \gamma$,

$$\max_{u,v \in V} |d_{\mathcal{T}}(u,v) - \tilde{d}_{\mathcal{T}}(u,v)| \leq O(\sqrt{k} \cdot n^{1/(2k)} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \sqrt{\log(2/\delta)} \cdot \log(1/\gamma)/\varepsilon)$$

by **Proposition E.2**. By letting $\tilde{d}(u,v) \leftarrow \min_{\mathcal{T} \in \mathbf{T}} \tilde{d}_{\mathcal{T}}(u,v)$, the result in **Lemma 4.1** guarantees the total algorithm is $(O(k \log \log n), O(\sqrt{k} \cdot n^{1/(2k)} \cdot \log^{2.5}(n) \cdot \log(k) \cdot \sqrt{\log(2/\delta)} \cdot \log(1/\gamma)/\varepsilon))$ -accurate, as claimed. □

D Proof of Theorem 4.4

Theorem 4.4. *For $\gamma \in (0, 0.5]$, $\varepsilon, \delta \in (0, 1]$, and $k \in \{2, \dots, \lfloor \log(n)/\log(2) \rfloor\}$, there is an (ε, δ) -DP algorithm for computing all-pairs shortest distances that is $(2k - 1, O(\sqrt{k} \cdot n^{1/(2k)} \cdot \log^{2.5}(n) \cdot \sqrt{\log(2/\delta)} \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon))$ -accurate with probability $1 - \gamma - 1/n^2$.*

Proof. The proof is the same as **Theorem 4.3**, except we run the $\varepsilon/(2\sqrt{2(t \cdot k) \log(2/\delta)})$ -DP algorithm given by **Corollary 4.1** to release private APSD $\tilde{d}_{\mathcal{T}}(u,v)$ for each $\mathcal{T} \in \mathbf{T}$. By advanced composition (**Lemma 2.2**), releasing the distances is (ε, δ) -DP. Moreover, by a union bound over all $t \cdot k$ trees, with probability $1 - \gamma$,

$$\max_{u,v \in V} |d_{\mathcal{T}}(u,v) - \tilde{d}_{\mathcal{T}}(u,v)| \leq O(\sqrt{k} \cdot n^{1/(2k)} \cdot \log^{2.5}(n) \cdot \sqrt{\log(2/\delta)} \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon)$$

by **Proposition E.2**. The stretch analysis is the same as in **Theorem E.2**. Consequently, with probability $1 - \gamma - 1/n^2$, the algorithm is $(2k - 1, O(\sqrt{k} \cdot n^{1/(2k)} \cdot \log^{2.5}(n) \cdot \sqrt{\log(2/\delta)} \cdot \log(k) \cdot \log(1/\gamma)/\varepsilon))$ -accurate. □

E Other Propositions and Proofs

We will make use of the following result several times, and so we prove it here for convenience.

Proposition E.1. *For any $x \in (0, 1)$, $n > 0$ and $m \in \mathbb{Z}^+$, we have that,*

$$(1 - x)^{mx^{-1} \log(n)} \leq n^{-m}$$

Proof. First, note that $(1 - x)^{mx^{-1} \log(n)} = n^{mx^{-1} \log(1-x)}$. Hence, it suffices to show that $mx^{-1} \log(1 - x) \leq -m$. To see this, we let $f(x) = mx^{-1} \log(1 - x)$ and note that $\lim_{x \rightarrow 0^+} f(x) = -m$, and $f'(x) < 0$ for all $x \in (0, 1)$. □

Finally, we state some elementary results regarding big- O notation which we will use several times throughout this paper in order to simplify the arguments of our asymptotic upper-bounds.

Proposition E.2. *Let $\alpha \geq 2$, $\gamma \in (0, 0.5]$, and $\lambda > 0$, then,*

1. $\log(\alpha^{1+\lambda}/\gamma) = O(\log(\alpha/\gamma))$, and,

2. $\log(\alpha/\gamma) = O(\log(\alpha) \log(1/\gamma))$.

Proof of 1. $\log(\alpha^{1+\lambda}/\gamma) = \log((\alpha/\gamma^{1/(1+\lambda)})^{1+\lambda}) = (1 + \lambda) \cdot \log(\alpha/\gamma^{1/(1+\lambda)}) \leq (1 + \lambda) \cdot \log(\alpha/\gamma) = O(\log(\alpha/\gamma))$.

Proof of 2. Suppose the base of our logarithm is $b = \{2, e\}$. Then,

$$\log(\alpha/\gamma) = \log(\alpha) + \log(1/\gamma) = O(\max\{\log(\alpha), \log(1/\gamma)\}) = O(\log(\alpha) \log(1/\gamma))$$

as $\alpha, 1/\gamma \geq 2$. □