Differentially Private Counting Queries on Approximate Shortest Paths

The 17th Annual International Conference on Combinatorial Optimization and Applications (COCOA'24)

Jesse Campbell

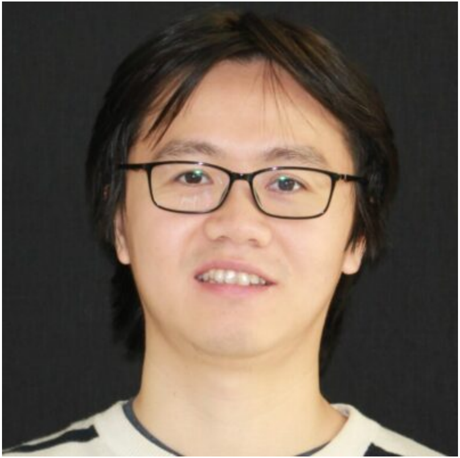
Duke Kunshan University

December 7th, 2024

Contents

Acknowledgments

Joint work with **Dr. Chunjiang Zhu**, Assistant Professor, University of North Carolina at Greensboro.



Acknowledgments

Work supported by **NSF Grant CNS-2349369** and the **Computing Research Association via the NSF**.

Work completed during the **UNCG GraLNA 2024 REU**.

Problem Definition

Given the following information,

Problem Definition

Given the following information,

Public –

1. Weighted graph $G = (V, E, \omega)$
2. System of paths $S \subset 2^E$

Problem Definition

Given the following information,

Public –

1. Weighted graph $G = (V, E, \omega)$
2. System of paths $S \subset 2^E$

Private –

1. Edge attribute $\phi : E \to \mathbb{R}^+$

Problem Definition

Given the following information,

Public –

1. Weighted graph $G = (V, E, \omega)$
2. System of paths $S \subset 2^E$

Private –

1. Edge attribute $\phi : E \to \mathbb{R}^+$

We wish to output the *counting queries* over S with *differential privacy*.

Problem Definition

Given the following information,

Public –

1. Weighted graph $G = (V, E, \omega)$
2. System of paths $S \subset 2^E$

Private –

1. Edge attribute $\phi : E \to \mathbb{R}^+$

We wish to output the *counting queries* over S with *differential privacy*.

Counting Query Definition

A **counting query** over a path $P \subset E$ is the number $\sum_{e \in P} \phi(e)$

Motivation

Range counting – counting the number of points that are contained in a certain geometrically-defined range

Motivation

Range counting – counting the number of points that are contained in a certain geometrically-defined range

Range counting on graphs – ranges are defined as paths

Motivation

Range counting – counting the number of points that are contained in a certain geometrically-defined range

Range counting on graphs – ranges are defined as paths

Application – patient transfer network

Motivation

Range counting – counting the number of points that are contained in a certain geometrically-defined range

Range counting on graphs – ranges are defined as paths

Application – patient transfer network

- *Vertices* represent medical facilities

Motivation

Range counting – counting the number of points that are contained in a certain geometrically-defined range

Range counting on graphs – ranges are defined as paths

Application – patient transfer network

- *Vertices* represent medical facilities
- *Edges* represent paths between facilities

Motivation

Range counting – counting the number of points that are contained in a certain geometrically-defined range

Range counting on graphs – ranges are defined as paths

Application – patient transfer network

- *Vertices* represent medical facilities
- *Edges* represent paths between facilities
- *Edge weights* represent travel times along paths

Motivation

Range counting – counting the number of points that are contained in a certain geometrically-defined range

Range counting on graphs – ranges are defined as paths

Application – patient transfer network

- *Vertices* represent medical facilities
- *Edges* represent paths between facilities
- *Edge weights* represent travel times along paths
- *Private edge attributes* represent number of patients in-transfer along paths

Differential Privacy Preliminaries

Neighboring Graphs

Two isomorphic graphs $G_1, G_2 = (V, E, \omega)$ with edge attribute functions $\phi_1, \phi_2 : E \to \mathbb{R}^+$ are said to be neighboring if

$$\sum_{e \in E} |\phi_1(e) - \phi_2(e)| \leq 1$$

Differential Privacy Preliminaries

Neighboring Graphs

Two isomorphic graphs $G_1, G_2 = (V, E, \omega)$ with edge attribute functions $\phi_1, \phi_2 : E \to \mathbb{R}^+$ are said to be neighboring if

$$\sum_{e \in E} |\phi_1(e) - \phi_2(e)| \leq 1$$

Sensitivity

The l_1 sensitivity of $\mathcal{A} : \mathcal{X} \to \mathbb{R}^D$ is defined as

$$\Delta_1(\mathcal{A}) \coloneqq \max_{X, X'} \|\mathcal{A}(X) - \mathcal{A}(X')\|_1$$

where X, X' are neighboring datasets.

Differential Privacy

Differential Privacy (DP) Definition

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

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

Differential Privacy

Differential Privacy (DP) Definition

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

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

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

Results Overview

Citation	**Path System**	ε-**DP**	(ε, δ)-**DP**
Deng et al. (2023)	Shortest	$\widetilde{O}(n^{1/3})$	$\widetilde{O}(n^{1/4})$
Bodwin et al. (2024)	Shortest	–	$\widetilde{\Omega}(n^{1/4})$
Our results (2024)	$(k \log\log(n))$-approximate	$\widetilde{O}(kn^{1/k})$	$\widetilde{O}(\sqrt{k}n^{1/(2k)})$

Results Overview

Citation	**Path System**	ε-**DP**	(ε, δ)-**DP**
Deng et al. (2023)	Shortest	$\widetilde{O}(n^{1/3})$	$\widetilde{O}(n^{1/4})$
Bodwin et al. (2024)	Shortest	–	$\widetilde{\Omega}(n^{1/4})$
Our results (2024)	$(k \log \log (n))$-approximate	$\widetilde{O}(kn^{1/k})$	$\widetilde{O}(\sqrt{k} n^{1/(2k)})$

In general, let $\mathbf{T}$ be a t-collective tree spanner of G such that $|\mathbf{T}| = \eta_t$. There is a ε-DP algorithm for releasing the counting queries that is $\widetilde{O}(\eta_t)$-accurate and a (ε, δ)-DP algorithm that is $\widetilde{O}(\sqrt{\eta_t})$-accurate with probability $1 - \gamma$.

Differential Privacy Technology

Basic Composition

Let $\varepsilon, \delta \in [0, 1]$ and $k \in \mathbb{N}$. If we run k mechanisms where each mechanism is $(\varepsilon/k, \delta/k)$-DP, then the entire algorithm is (ε, δ)-DP.

Differential Privacy Technology

Basic Composition

Let $\varepsilon, \delta \in [0, 1]$ and $k \in \mathbb{N}$. If we run k mechanisms where each mechanism is $(\varepsilon/k, \delta/k)$-DP, then the entire algorithm is (ε, δ)-DP.

Laplace Mechanism

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

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

The Laplace mechanism is ε-differentially private.

Collective Tree Spanners

Collective Tree Spanner Definition

A collection of spanning trees $\mathbf{T}$ of G is said to be an α-collective tree spanner of G if for every $u, v \in V$, there is a tree $\mathcal{T} \in \mathbf{T}$ such that $d_{\mathcal{T}}(u, v) \leq \alpha \cdot d_G(u, v)$.

Collective Tree Spanners

Collective Tree Spanner Definition

A collection of spanning trees $\mathbf{T}$ of G is said to be an α-collective tree spanner of G if for every $u, v \in V$, there is a tree $\mathcal{T} \in \mathbf{T}$ such that $d_{\mathcal{T}}(u, v) \leq \alpha \cdot d_G(u, v)$.

Abraham et al. (2020). There is a polynomial time deterministic algorithm that finds a $(k \log \log (n))$-collective tree spanner with size $k \cdot n^{1/k}$.

A Recursive Tree Algorithm

Lemma 5. Let $\mathcal{T} = (V, E, \omega)$ be a tree rooted at $z \in V$ with $\varepsilon \in (0, 1]$ and $\gamma \in (0, 0.5]$. Then there is an ε-DP algorithm for releasing counting queries from the root to all other vertices on $\mathcal{T}$ that is $O(\log^{1.5}(n) \cdot \log(n/\gamma)/\varepsilon)$-accurate with probability $1 - \gamma$.

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$

Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

1: Let z^* be the vertex in $\mathcal{T}$ 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

2: Let $z_1, z_2, ..., z_\alpha$ be the children of z^*

3: Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$

4: Sample $X \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$

5: Let $\omega(z, \mathcal{T}) = 0$

6: Sample $(X_1, X_2, ..., X_\alpha) \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$

7: Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$

8: For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$

Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

1: Let z^* be the vertex in $\mathcal{T}$ 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
2: Let $z_1, z_2, ..., z_\alpha$ be the children of z^*
3: Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$
4: Sample $X \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$
5: Let $\omega(z, \mathcal{T}) = 0$
6: Sample $(X_1, X_2, ..., X_\alpha) \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$
7: Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$
8: For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$

Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

Let z^* be the vertex in $\mathcal{T}$ 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 $z_1, z_2, ..., z_\alpha$ be the children of z^*

Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$

Sample $X \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$

Let $\omega(z, \mathcal{T}) = 0$

Sample $(X_1, X_2, ..., X_\alpha) \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$

Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$

For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$

Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

Let z^* be the vertex in $\mathcal{T}$ 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 $z_1, z_2, ..., z_\alpha$ be the children of z^*

Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$

Sample $X \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$

Let $\omega(z, \mathcal{T}) = 0$

Sample $(X_1, X_2, ..., X_\alpha) \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$

Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$

For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$
Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

1: Let z^* be the vertex in $\mathcal{T}$ 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
2: Let $z_1, z_2, ..., z_\alpha$ be the children of z^*
3: Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$
4: Sample $X \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$
5: Let $\omega(z, \mathcal{T}) = 0$
6: Sample $(X_1, X_2, ..., X_\alpha) \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$
7: Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$
8: For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$

Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

Let z^* be the vertex in $\mathcal{T}$ 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 $z_1, z_2, ..., z_\alpha$ be the children of z^*

Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$

Sample $X \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$

Let $\omega(z, \mathcal{T}) = 0$

Sample $(X_1, X_2, ..., X_\alpha) \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$

Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$

For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$

Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

1: Let z^* be the vertex in $\mathcal{T}$ 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

2: Let $z_1, z_2, ..., z_\alpha$ be the children of z^*

3: Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$

4: Sample $X \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$

5: Let $\omega(z, \mathcal{T}) = 0$

6: Sample $(X_1, X_2, ..., X_\alpha) \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$

7: Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$

8: For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$

Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

Let z^* be the vertex in $\mathcal{T}$ 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 $z_1, z_2, ..., z_\alpha$ be the children of z^*

Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$

Sample $X \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$

Let $\omega(z, \mathcal{T}) = 0$

Sample $(X_1, X_2, ..., X_\alpha) \sim \mathrm{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$

Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$

For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$

Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

1: Let z^* be the vertex in $\mathcal{T}$ 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

2: Let $z_1, z_2, ..., z_\alpha$ be the children of z^*

3: Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$

4: Sample $X \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$

5: Let $\omega(z, \mathcal{T}) = 0$

6: Sample $(X_1, X_2, ..., X_\alpha) \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$

7: Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$

8: For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

A Recursive Tree Algorithm

Algorithm 1 DPCQ on Rooted Trees

Input: Tree $\mathcal{T} = (V, E, \mathbf{w})$ rooted at $z \in V$ with edge attribute ϕ; parameter $\varepsilon \in (0, 1)$

Output: ε-DP approximate counting queries in $\mathcal{T}$, $\{\widetilde{\omega}(u, v)\}_{u,v \in V}$

1: Let z^* be the vertex in $\mathcal{T}$ 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

2: Let $z_1, z_2, ..., z_\alpha$ be the children of z^*

3: Let $\mathcal{T}_i$ be the subtree rooted at z_i for $i \in [\alpha]$, and $\mathcal{T}_0 = \mathcal{T} - \{\mathcal{T}_1, ..., \mathcal{T}_\alpha\}$

4: Sample $X \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z^*, \mathcal{T}) = \omega(z, z^*) + X$

5: Let $\omega(z, \mathcal{T}) = 0$

6: Sample $(X_1, X_2, ..., X_\alpha) \sim \text{Lap}(\log(n)/\varepsilon)$ and let $\omega(z_i, \mathcal{T}) = \omega(z^*, \mathcal{T}) + \phi(z^*, z_i) + X_i$

7: Recursively compute counting queries in each subtree $\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_\alpha$

8: For each vertex $w \in V$, if $w \in \mathcal{T}_i$, let $\widetilde{\omega}(z, w) = \omega(z_i, \mathcal{T}_z) + \omega(w, \mathcal{T}_i)$

Tree Algorithm Analysis

Remark 1. Every released counting query is the sum of at most $\log n$ estimates.

Tree Algorithm Analysis

Remark 1. Every released counting query is the sum of at most $\log n$ estimates.

Proof. Every subtree has at most $n/2$ vertices, therefore the recursion depth is bounded above by $\log n$. $\square$

Tree Algorithm Analysis

Remark 1. Every released counting query is the sum of at most $\log n$ estimates.

Proof. Every subtree has at most $n/2$ vertices, therefore the recursion depth is bounded above by $\log n$. □

Accuracy analysis. We apply the following lemma,

Laplace R.V. Concentration Bound

Let $X_1, \ldots, X_t$ be independent random variables distributed according to $\mathrm{Lap}(b)$, and let $X = X_1 + \ldots + X_t$. Then for all $\gamma \in (0, 1)$, with probability at least $1 - \gamma$ we have,

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

Tree Algorithm Analysis

The error in each estimate $\widetilde{\omega}(u, v)$ is the sum of at most $\log n$ i.i.d. Laplace R.V.s distributed according to $\mathrm{Lap}(\log(n)/\varepsilon)$, therefore by the previous bound,

Tree Algorithm Analysis

The error in each estimate $\widetilde{\omega}(u, v)$ is the sum of at most $\log n$ i.i.d. Laplace R.V.s distributed according to $\mathrm{Lap}(\log(n)/\varepsilon)$, therefore by the previous bound,

$$|\widetilde{\omega}(u, v) - \omega_T(u, v)| \leq O(\log^{1.5}(n) \cdot \log(1/\gamma)/\varepsilon)$$

with probability at least $1 - \gamma$.

Tree Algorithm Analysis

The error in each estimate $\widetilde{\omega}(u, v)$ is the sum of at most $\log n$ i.i.d. Laplace R.V.s distributed according to $\mathrm{Lap}(\log(n)/\varepsilon)$, therefore by the previous bound,

$$|\widetilde{\omega}(u, v) - \omega_T(u, v)| \leq O(\log^{1.5}(n) \cdot \log(1/\gamma)/\varepsilon)$$

with probability at least $1 - \gamma$.

Privacy analysis. The estimates computed in each recursion are $(\varepsilon/\log(n))$-DP by the **Laplace Mechanism**.

Tree Algorithm Analysis

The error in each estimate $\widetilde{\omega}(u, v)$ is the sum of at most $\log n$ i.i.d. Laplace R.V.s distributed according to $\mathrm{Lap}(\log(n)/\varepsilon)$, therefore by the previous bound,

$$|\widetilde{\omega}(u, v) - \omega_T(u, v)| \leq O(\log^{1.5}(n) \cdot \log(1/\gamma)/\varepsilon)$$

with probability at least $1 - \gamma$.

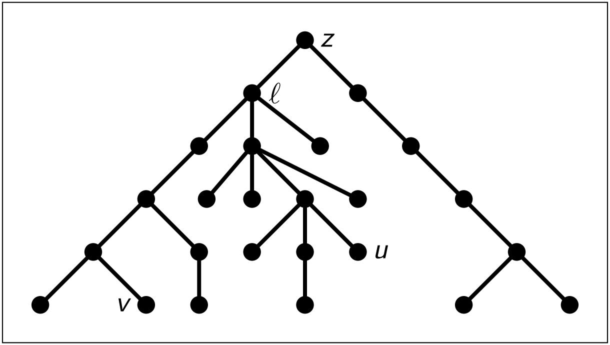
Privacy analysis. The estimates computed in each recursion are $(\varepsilon/\log(n))$-DP by the **Laplace Mechanism**. By **Basic Composition**, the entire algorithm is ε-DP.

Extension to General Trees

Algorithm 1 suffices to compute DPCQ for any tree.

Extension to General Trees

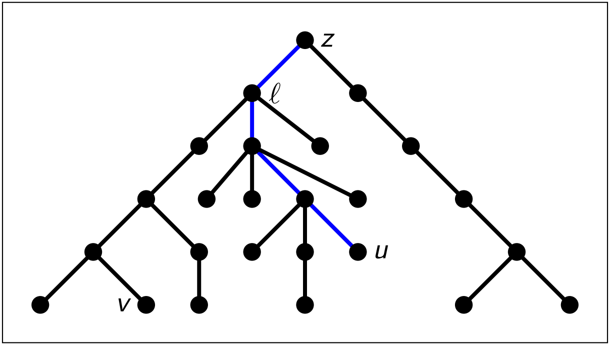
Algorithm 1 suffices to compute DPCQ for any tree. Arbitrarily pick a root z and let ℓ be the least common ancestor of $u, v \in V$.



$$\widetilde{\omega}(u, v) =$$

Extension to General Trees

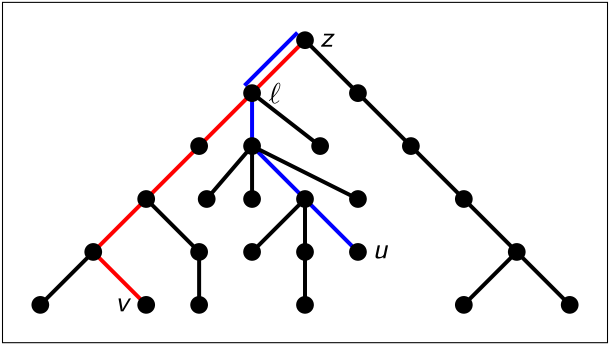
Algorithm 1 suffices to compute DPCQ for any tree. Arbitrarily pick a root z and let ℓ be the least common ancestor of $u, v \in V$.



$$\widetilde{\omega}(u, v) = \widetilde{\omega}(z, u)$$

Extension to General Trees

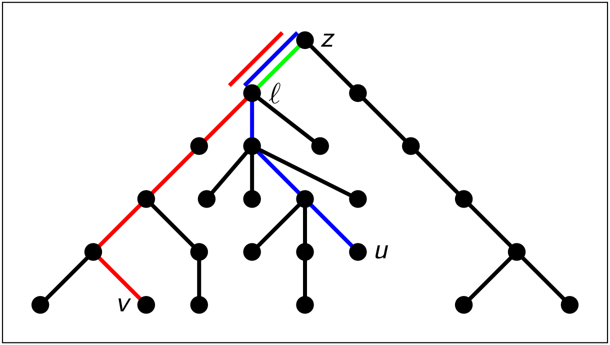
Algorithm 1 suffices to compute DPCQ for any tree. Arbitrarily pick a root z and let ℓ be the least common ancestor of $u, v \in V$.



$$\widetilde{\omega}(u, v) = \color{blue}{\widetilde{\omega}(z, u)} + \color{red}{\widetilde{\omega}(z, v)}$$

Extension to General Trees

Algorithm 1 suffices to compute DPCQ for any tree. Arbitrarily pick a root z and let ℓ be the least common ancestor of $u, v \in V$.



$$\widetilde{\omega}(u, v) = \widetilde{\omega}(z, u) + \widetilde{\omega}(z, v) - 2 \cdot \widetilde{\omega}(z, \ell)$$

A Generalization

Theorem 1. Let $\mathbf{T}$ be a t-collective tree spanner of G such that $|\mathbf{T}| = \eta_t$. For $\gamma \in (0, 0.5]$ and $\varepsilon \in (0, 1]$, there is an ε-DP algorithm for releasing the counting query between $u, v \in V$ on a t-approximate shortest path in G that is $O(\eta_t \cdot \log^{2.5}(n) \cdot \log(1/\gamma)/\varepsilon)$-accurate with probability $1 - \gamma$.

A Generalization

Theorem 1. Let $\mathbf{T}$ be a t-collective tree spanner of G such that $|\mathbf{T}| = \eta_t$. For $\gamma \in (0, 0.5]$ and $\varepsilon \in (0, 1]$, there is an ε-DP algorithm for releasing the counting query between $u, v \in V$ on a t-approximate shortest path in G that is $O(\eta_t \cdot \log^{2.5}(n) \cdot \log(1/\gamma)/\varepsilon)$-accurate with probability $1 - \gamma$.

Proof. We run the (ε/η_t)-DP mechanism given in **Algorithm 1** on each tree in $\mathbf{T}$. By **Basic Composition** and a union bound, we can release the estimates over all of $\mathbf{T}$ with

$$O(\eta_t \cdot \log^{2.5}(n) \cdot \log(1/\gamma)/\varepsilon)$$

error with probability $1 - \gamma$. □

Collective Tree Spanners

Assuming the Erdös girth conjecture, any $(2k-1)$ spanner must have at least $\Omega(n^{1+(1/k)})$ edges, and this bound is tight.

Collective Tree Spanners

Assuming the Erdös girth conjecture, any $(2k-1)$ spanner must have at least $\Omega(n^{1+(1/k)})$ edges, and this bound is tight. So, any $(2k-1)$ collective tree spanner must have at least $\Omega(n^{1/k})$ trees.

Collective Tree Spanners

Assuming the Erdös girth conjecture, any $(2k-1)$ spanner must have at least $\Omega(n^{1+(1/k)})$ edges, and this bound is tight. So, any $(2k-1)$ collective tree spanner must have at least $\Omega(n^{1/k})$ trees.

The best (and possibly only) multiplicative, collective tree spanner in the literature has $k \cdot n^{1/k}$ trees and stretch $(k \log \log (n))$.

Collective Tree Spanners

Assuming the Erdös girth conjecture, any $(2k-1)$ spanner must have at least $\Omega(n^{1+(1/k)})$ edges, and this bound is tight. So, any $(2k-1)$ collective tree spanner must have at least $\Omega(n^{1/k})$ trees.

The best (and possibly only) multiplicative, collective tree spanner in the literature has $k \cdot n^{1/k}$ trees and stretch $(k \log \log (n))$.

Open question 1 – Can we construct a collective tree spanner that has a size/stretch tradeoff that is closer to being optimal?

APSD with Differential Privacy

A *closely related* problem is the all-pairs shortest distances problem with differential privacy.

APSD with Differential Privacy

A *closely related* problem is the all-pairs shortest distances problem with differential privacy.The problem is identical to DP counting queries if the edge weights are kept private.

APSD with Differential Privacy

A *closely related* problem is the all-pairs shortest distances problem with differential privacy.The problem is identical to DP counting queries if the edge weights are kept private.

Chen et al. (2023). There is an algorithm which solves the DP APSD problem with $\widetilde{O}(n^{1/2}/\varepsilon)$-error w.h.p.

APSD with Differential Privacy

A *closely related* problem is the all-pairs shortest distances problem with differential privacy.The problem is identical to DP counting queries if the edge weights are kept private.

Chen et al. (2023). There is an algorithm which solves the DP APSD problem with $\widetilde{O}(n^{1/2}/\varepsilon)$-error w.h.p.

Bodwin et al. (2024). Any algorithm which solves the DP APSD problem must have error at least $\widetilde{\Omega}(n^{1/4})$.

APSD with Differential Privacy

A *closely related* problem is the all-pairs shortest distances problem with differential privacy.The problem is identical to DP counting queries if the edge weights are kept private.

Chen et al. (2023). There is an algorithm which solves the DP APSD problem with $\widetilde{O}(n^{1/2}/\varepsilon)$-error w.h.p.

Bodwin et al. (2024). Any algorithm which solves the DP APSD problem must have error at least $\widetilde{\Omega}(n^{1/4})$.

Open question 2 – Close the gap between these upper- and lower-bounds.

References

Chengyuan, Deng et al. (2022). Differentially Private Range Query on Shortest Paths. arXiv preprint. https://arxiv.org/abs/2212.07997

Bodwin, Greg et al. (2024). The Discrepancy of Shortest Paths. arXiv preprint. https://arxiv.org/abs/2401.15781

Abraham, Ittai et al. (2020). Ramsey Spanning Trees and Their Applications. ACM Trans. Algorithms 16.2. issn: 1549-6325.

Chen, Justin et al. (2023). Differentially Private All-Pairs Shortest Path Distances: Improved Algorithms and Lower Bounds, Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 5040-5067, 10.1137/1.9781611977554.ch184.