

Optimal Bounds for Private Minimum Spanning Trees via Input Perturbation

RASMUS PAGH*, BARC, University of Copenhagen, Denmark

LUKAS RETSCHMEIER*, BARC, University of Copenhagen, Denmark

HAO WU*[†], University of Waterloo, Canada

HANWEN ZHANG*, BARC, University of Copenhagen, Denmark

We study the problem of privately releasing an approximate minimum spanning tree (MST). Given a graph $G = (V, E, \mathbf{W})$ where V is a set of n vertices, E is a set of m undirected edges, and $\mathbf{W} \in \mathbb{R}^{|E|}$ is an edge-weight vector, our goal is to publish an approximate MST under *edge-weight* differential privacy, as introduced by Sealfon in PODS 2016, where V and E are considered public and the weight vector is private. Our neighboring relation is ℓ_∞ -distance on weights: for a sensitivity parameter Δ_∞ , graphs $G = (V, E, \mathbf{W})$ and $G' = (V, E, \mathbf{W}')$ are neighboring if $\|\mathbf{W} - \mathbf{W}'\|_\infty \leq \Delta_\infty$.

Existing private MST algorithms face a trade-off, sacrificing either computational efficiency or accuracy. We show that it is possible to get the best of both worlds: With a suitable random perturbation of the input that does *not* suffice to make the weight vector private, the result of *any* non-private MST algorithm will be private and achieves a state-of-the-art error guarantee.

Furthermore, by establishing a connection to *Private Top-k Selection* [Steinke and Ullman, FOCS '17], we give the first privacy-utility trade-off lower bound for MST under approximate differential privacy, demonstrating that the error magnitude, $\tilde{O}(n^{3/2})$, is optimal up to logarithmic factors. That is, our approach matches the time complexity of any non-private MST algorithm and at the same time achieves optimal error. We complement our theoretical treatment with experiments that confirm the practicality of our approach.

CCS Concepts: • **Security and privacy** → **Privacy protections**; • **Theory of computation** → **Graph algorithms analysis**.

Additional Key Words and Phrases: differential privacy, minimum spanning tree

ACM Reference Format:

Rasmus Pagh, Lukas Retschmeier, Hao Wu, and Hanwen Zhang. 2025. Optimal Bounds for Private Minimum Spanning Trees via Input Perturbation. *Proc. ACM Manag. Data* 3, 2 (PODS), Article 103 (May 2025), 26 pages. <https://doi.org/10.1145/3725240>

1 Introduction

Graph algorithms are a cornerstone of data analysis, but directly applying classical graph algorithms to inputs that contain sensitive information risks disclosing too much information about the input. In recent years *differentially private* graph algorithms have emerged as a principled approach to ensuring that such disclosure is limited. We refer to the survey of Li et al. [23] for a general overview

*All authors contributed equally to this research.

[†]This work was partially carried out while the author was still at BARC, University of Copenhagen.

Authors' Contact Information: Rasmus Pagh, BARC, University of Copenhagen, Denmark, pagh@ku.dk; Lukas Retschmeier, BARC, University of Copenhagen, Denmark, lure@di.ku.dk; Hao Wu, University of Waterloo, Canada, hao.wu1@uwaterloo.ca; Hanwen Zhang, BARC, University of Copenhagen, Denmark, hazh@di.ku.dk.



This work is licensed under a Creative Commons Attribution-NoDerivatives 4.0 International License.

© 2025 Copyright held by the owner/author(s).

ACM 2836-6573/2025/5-ART103

<https://doi.org/10.1145/3725240>

Reference	Additive Error	Time	Technique
[37]	$O\left((1/\varepsilon) \cdot n\sqrt{m} \cdot (\log n) \sqrt{\log(1/\delta)}\right)$	MST + $O(m)$	Input privatization
[37]	$\Omega(n)$	–	Lower bound
[31]	$O\left((1/\varepsilon) \cdot n^{3/2} \cdot (\log n) \sqrt{\log(1/\delta)}\right)$	$O(nm)$	In-place noise
[29]	$O\left((1/\varepsilon) \cdot n^{3/2} \cdot (\log n) \sqrt{\log(1/\delta)}\right)$	$O(m + n^{3/2} \log n)$	In-place noise
New	$O\left((1/\varepsilon) \cdot n^{3/2} \cdot (\log n) \sqrt{\log(1/\delta)}\right)$	MST + $O(m)$	Input perturbation
New	$\Omega\left((1/\varepsilon) \cdot n^{3/2} \cdot (\log n)\right)$	–	Lower bound

Table 1. Results on (ε, δ) -DP MST with ℓ_∞ **neighboring** relationship on edge weights, assuming sensitivity $\Delta_\infty = 1$. Sealfon [37] and Pinot [31] originally provided upper bounds for pure DP, but these can be adapted for approximate DP. Sealfon considered the ℓ_1 neighboring relationship, which leads to the same lower bound and a weaker upper bound for ℓ_∞ . The unpublished manuscript [29] was written by a subset of the authors. The “MST” in the **Time** column refers to the running time of any non-private MST algorithm.

of private graph algorithms and to the tutorial of Brito et al. [3] for a data management specific overview.

The *minimum spanning tree* (MST) problem is a classical graph optimization problem with many applications. With differential privacy constraints, it has been studied in the context of clustering algorithms [2, 18, 22, 32] and as a subroutine for computing graphical models [24].

We consider the problem of privately releasing an MST for a graph G where both the set of vertices $V = \{1, \dots, n\}$ and edges $E = \{1, \dots, m\}$ are public, but we keep the weight vector $\mathbf{W} = (w_1, \dots, w_m) \in \mathbb{R}^{|E|}$ private. We consider the ℓ_∞ neighborhood relation, where all weights can differ by at most Δ_∞ and want to protect the information encoded on the weights, which is known as *edge-weight differential privacy* [37].

An application example is to model passenger data on a public city traffic network. Another setting comes from graphical modeling: Consider a dataset $D = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(d)})$ of size d where each vector $\mathbf{x}^{(i)} \in \{0, 1\}^n$ represents a list of sensitive binary attributes. The *Chow-Liu Tree* [6] is the minimum spanning tree derived from the negated *mutual information matrix* encoding all pairwise mutual information of the attributes. Changing one $\mathbf{x}$ in D could simultaneously alter *all* weights.

Currently, there are two different algorithmic approaches: *input-privatization* [37] and *in-place* [24, 32]. The former method adds noise (e.g., Laplace or Gaussian) to each edge weight and publishes the entire noisy weight vector. This allows anyone to compute the MST while privacy is ensured by the *post-processing* property of differential privacy. While this approach allows the design of an (expected) linear-time algorithm, it provides only a worst-case additive error of $\tilde{O}(n^2)$.

The latter approach, which adds noise during the execution of an MST algorithm, achieves a better error bound of $\tilde{O}(n^{3/2})$, but its best-known running time is $\tilde{O}(n^{3/2} + m)$ for a fixed privacy parameter [29]. Thus, there is a gap, and whether it is possible to achieve the “best of both worlds” has remained unknown. We answer these two open questions in the affirmative:

Question 1: Can we design an (expected) linear-time private MST algorithm that matches the error guarantee of $\tilde{O}(n^{3/2})$ of the *in-place* approach?

Question 2: Is the error asymptotically optimal?

1.1 Our Contribution

We introduce the *first* algorithmic framework that reduces the private MST problem (under the ℓ_∞ neighboring relationship) to the non-private one, achieving an additive error of $\tilde{O}(n^{3/2})$. Compatible with any non-private MST algorithm, it enables the *first* (expected) linear-time private MST solution.

We also establish the first $\tilde{\Omega}(n^{3/2})$ error lower bound, proving the asymptotic optimality of our framework. Formally, we show the following upper and lower bounds:

THEOREM 1.1 (UPPER BOUND). *Let $G = (V, E, \mathbf{W})$ be a graph with n vertices and m edges, and let $\epsilon, \delta > 0$. Consider an arbitrary (non-private) algorithm that computes an MST of G within time $t(n, m)$, independent of the weight vector. Then there exists an (ϵ, δ) -differentially private mechanism $\mathcal{M}$ that releases a random, approximately minimum spanning tree $\mathcal{M}(G)$, such that, if T^* is a minimum spanning tree:*

- *the weight difference satisfies $\mathbb{E} \left[\sum_{e \in \mathcal{M}(G)} w_e \right] - \sum_{e \in T^*} w_e \in O\left(\frac{1}{\epsilon} \cdot n^{3/2} \cdot (\log n) \cdot \sqrt{\log(1/\delta)}\right)$, and*
- *the running time is $t(n, m) + O(m)$.*

For example, the randomized linear time algorithm by Karger et al. [19] or the best-known near-linear time deterministic algorithm due to Chazelle [5] can be used in Theorem 1.1. Our approach matches all known upper bounds, meanwhile it is very simple to implement. See Table 1 for a comprehensive summary. Moreover, the result can be easily extended to finding a maximum weight independent set in any matroid with rank n and m elements, where $t(n, m)$ is the running time for any algorithm that computes the maximum weight independent set in such a matroid.

The key technique is a novel approach (formally introduced in Section 3.3) to iteratively sample k items from a larger candidate set with probabilities proportional to their weights. After each selection, the chosen item and a subset of the remaining candidates—determined by the sequence of previously selected items—are removed before the next selection. Our approach generates noisy weights for all items at the beginning of the sampling process, which are reused throughout the iterative selection procedure, thereby eliminating the need to regenerate fresh randomness at each step. As discussed in Section 5, this technique extends a substantial body of research on top- k sampling [7, 28, 36, 42].

Lower Bound. Our lower bound partially resolves an open question of Hladík and Tětek [16], establishing a tight bound for worst-case instances under approximate differential privacy:

THEOREM 1.2 (LOWER BOUND). *Let $\epsilon \leq 1$ and $\delta \in O(1/\sqrt{n})$, and let $\mathcal{M}$ be an (ϵ, δ) -DP MST algorithm. There exists an input graph $G = (V, E, \mathbf{W})$ with n vertices such that, if T^* is the minimum spanning tree of G , the (random) spanning tree $\mathcal{M}(G)$ released by $\mathcal{M}$ satisfies:*

$$\mathbb{E}_{\mathcal{M}} \left[\sum_{e \in \mathcal{M}(G)} w_e \right] - \sum_{e \in T^*} w_e \in \Omega\left(\frac{1}{\epsilon} \cdot n^{3/2} \cdot \ln n\right). \quad (1)$$

The proof technique is based on a lower bound technique for top- k selection (reporting k elements of approximately maximum value) in [38]. First, observe that maximization and minimization are equivalent for spanning trees by simply flipping the sign of all edge weights. A maximum spanning tree can be seen as a top- $(n - 1)$ selection problem (selecting $n - 1$ edges of the approximately largest value) with an additional constraint that the edge set is acyclic. Thus, it is natural to guess that the hard instances of Steinke and Ullman [38], embedded into the weights of a complete graph, yield a hard instance for the maximum spanning tree as well. By carefully adapting the proof in [38], we show that this is indeed the case. In particular, leveraging a result from the Erdős–Rényi random graph model [12], we show the existence of a spanning tree with weight similar to the top- $(n - 1)$ edge weights.

Organization. The rest of the paper is organized as follows. Section 2 introduces the problem formally and reviews the necessary preliminaries. Section 3 describes the reduction from normal non-private MST algorithm to private MST algorithm that achieve the desired utility and running time. Section 4 establishes the lower bound for the problem. Section 5 presents a detailed comparison of

our techniques with previous ones and explores the broader background of the problem. Experiments in Section 6 show the practicality of our approach.

2 Preliminaries

We consider a graph $G = (V, E, \mathbf{W})$ where the set of vertices $V = \{1, \dots, n\}$ and edges $E = \{1, \dots, m\}$ are public, and the weight vector $\mathbf{W} = (w_1, \dots, w_m) \in \mathbb{R}^m$ is private. For each edge $e \in E$, let w_e denote its weight. The cost of a subset $T \subseteq E$ is defined as $w(T) := \sum_{e \in T} w_e$. A *spanning tree* is an acyclic subset $T \subseteq E$ that makes the graph connected. Denote $\mathcal{T}(G)$ the collection of all possible spanning trees in G . A *minimum (cost) spanning tree* (MST) is a spanning tree T^* which minimizes $w(T^*)$. We sometimes hide logarithmic factors using tilde notation for $\tilde{O}$, $\tilde{\Theta}$ and $\tilde{\Omega}$.

2.1 Differential Privacy

Given an input graph $G = (V, E, \mathbf{W})$, the private MST problem aims to find a spanning tree T while preserving the privacy of the weight vector $\mathbf{W}$.

Privacy Guarantee. To achieve this, a private MST algorithm must produce similar output distributions for input graphs with similar weights. In what follows, we first formally define similar inputs and then similar output distributions.

DEFINITION 2.1 (ℓ_∞ -NEIGHBORING INPUTS). *Given $\Delta_\infty > 0$, two weight vectors $\mathbf{W}, \mathbf{W}' \subseteq \mathbb{R}^E$ are neighboring, denoted $\mathbf{W} \sim \mathbf{W}'$, if and only if $\|\mathbf{W} - \mathbf{W}'\|_\infty = \max_{e \in E} |w_e - w'_e| \leq \Delta_\infty$. Two graphs $G = (V, E, \mathbf{W})$ and $G' = (V', E', \mathbf{W}')$ are neighboring, denoted $G \sim G'$, if and only if $V = V'$, $E = E'$, and $\mathbf{W} \sim \mathbf{W}'$.*

For simplicity, we assume throughout the paper that $\Delta_\infty = 1$. If $\Delta_\infty \neq 1$, all bounds presented can be generalized by scaling a factor of Δ_∞ .

DEFINITION 2.2 (DWORK ET AL. [9] (ϵ, δ)-PRIVATE ALGORITHM). *Let $\epsilon, \delta > 0$, $G = (V, E)$ be a graph, and let $\mathcal{T}(G)$ denote the set of all possible spanning trees in G . An MST algorithm $\mathcal{M}$ is called (ϵ, δ) -differentially private (DP), if for every $G = (V, E, \mathbf{W})$, $G' = (V', E', \mathbf{W}')$ such that $G \sim G'$, and all $Z \subseteq \mathcal{T}(G)$,*

$$\Pr[\mathcal{M}(G) \in Z] \leq e^\epsilon \cdot \Pr[\mathcal{M}(G') \in Z] + \delta. \quad (2)$$

The above definitions follow the notion of edge-weight differential privacy introduced by Sealfon [37]. Other privacy notions for graphs include *edge-level privacy* [15] and *node-level privacy* [20].

Further, although we present the definition in the context of private MST algorithms, it applies more generally to any randomized algorithms $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the input space, which is associated with a symmetric relation $\sim$ that defines neighboring inputs.

Composition. We also explore an alternative formulation of differential privacy, which can offer a tighter analysis of the overall privacy guarantee of the composition of a sequence of private algorithms.

DEFINITION 2.3 ([41] RÉNYI DIVERGENCE). *Let $(\Omega, \mathcal{F}, \mu)$ be a measure space where μ is σ -finite. Let P and Q be probability measures that are absolutely continuous with respect to μ , and let p and q denote their Radon-Nikodym derivatives with respect to μ . For $\alpha \in (1, \infty)$, the Rényi divergence of order α of P from Q is defined as*

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \ln \int_\Omega p^\alpha q^{1-\alpha} d\mu. \quad (3)$$

DEFINITION 2.4 ([4] ρ -ZERO-CONCENTRATED DIFFERENTIAL PRIVACY). *Let $\rho > 0$. An algorithm $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ satisfies ρ -zCDP, if for all $\alpha > 1$ and all pairs of neighboring inputs $X, X' \in \mathcal{X}$, s.t.,*

$X \sim X'$, it holds that

$$D_\alpha(\mathcal{M}(X) || \mathcal{M}(X')) \leq \rho\alpha, \quad (4)$$

where $D_\alpha(\mathcal{M}(X) || \mathcal{M}(X'))$ denotes the α -Rényi divergence between two output distributions of $\mathcal{M}(X)$ and $\mathcal{M}(X')$.

Its (partial) relationship with (ϵ, δ) -DP and its composition property are outlined below.

FACT 2.5 ([4] CONVERSION). *If $\mathcal{M}$ satisfies ρ -zCDP, then $\mathcal{M}$ is $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$ -DP for all $\delta > 0$. Conversely, if $\mathcal{M}$ satisfies ϵ -DP, then $\mathcal{M}$ satisfies ρ -zCDP for $\rho \leq \frac{1}{2}\epsilon^2$.*

FACT 2.6 ([4] COMPOSITION). *If $\mathcal{M}_1$ and $\mathcal{M}_2$ satisfy ρ_1 -zCDP and ρ_2 -zCDP, respectively, then $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2)$ satisfies $(\rho_1 + \rho_2)$ -zCDP.*

Group Privacy. Group privacy studies the similarity between the output distributions of a private algorithm for two inputs that are " r -hops away", where $r \in \mathbb{N}^+$.

FACT 2.7 ([40] GROUP PRIVACY). *Let $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{Y}$ be an (ϵ, δ) -differentially private mechanism. Given $r \in \mathbb{N}^+$ and $X, X' \in \mathcal{X}$, if there exist $X^{(1)}, \dots, X^{(r-1)}$ such that $X^{(i-1)} \sim X^{(i)}$ for each $i \in [r]$ (where we define $X^{(0)} = X$ and $X^{(r)} = X'$), then for each (measurable) subset $S \subseteq \mathcal{Y}$, we have:*

$$\Pr[\mathcal{M}(X) \in S] \leq e^{r\epsilon} \cdot \Pr[\mathcal{M}(X') \in S] + \frac{e^{r\epsilon} - 1}{e^\epsilon - 1} \cdot \delta. \quad (5)$$

Exponential Mechanism. The exponential mechanism [26] can be used to differentially private release discrete outputs. It operates over an input space $\mathcal{X}$ (with neighboring datasets defined by a relation $\sim$) and a finite output space $\mathcal{Y}$. The mechanism $\mathcal{M}_{\text{EXP}} : \mathcal{X} \rightarrow \mathcal{Y}$ assigns the following probabilities for a given dataset $x \in \mathcal{X}$:

$$\Pr[\mathcal{M}_{\text{EXP}}(x) = y] \propto \exp(-\epsilon \cdot \mathcal{E}(x, y) / (2 \cdot \Delta_{\text{EXP}})), \quad \forall y \in \mathcal{Y} \quad (6)$$

where $\mathcal{E} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ is the *loss function* quantifying the cost of selecting y given input x , and Δ_{EXP} is the *sensitivity* of $\mathcal{E}$, i.e., the maximum change in the loss function for neighboring datasets:

$$\Delta_{\text{EXP}} \doteq \max_{x \sim x', y \in \mathcal{Y}} |\mathcal{E}(x, y) - \mathcal{E}(x', y)|.$$

FACT 2.8 ([26] PROPERTIES OF $\mathcal{M}_{\text{EXP}}$). *The exponential mechanism $\mathcal{M}_{\text{EXP}}$ satisfies ϵ -differential privacy. Moreover, for each $\beta \in (0, 1)$, and $\tau \doteq \frac{2 \cdot \Delta_{\text{EXP}}}{\epsilon} \cdot \ln \frac{|\mathcal{Y}|}{\beta}$, it holds that:*

$$\Pr[\mathcal{E}(x, \mathcal{M}_{\text{EXP}}(x)) \geq \min_{y \in \mathcal{Y}} \mathcal{E}(x, y) + \tau] \leq \beta, \quad \forall x \in \mathcal{X}. \quad (7)$$

2.2 Probabilities

We discuss several probability distributions applied in this paper.

DEFINITION 2.9 ([35] BETA DISTRIBUTION). *The beta distribution $\text{Beta}(\alpha, \beta)$ is a distribution defined on $[0, 1]$ whose density is given by $p(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$, $\forall x \in [0, 1]$, where $\alpha, \beta > 0$ are shape parameters, $B(\alpha, \beta) \doteq \int_0^1 x^{\alpha-1}(1-x)^{\beta-1} dx$ is a normalization constant.*

The cornerstone of our algorithm design is the exponential distribution.

DEFINITION 2.10 (EXPONENTIAL DISTRIBUTION). *Given $\lambda > 0$, a random variable X follows exponential distribution $\text{Exp}(\lambda)$, denoted by $X \sim \text{Exp}(\lambda)$, if it has density $p_X(x) \doteq \lambda e^{-\lambda x}$, $\forall x \geq 0$, and $p_X(x) \doteq 0$, $\forall x < 0$.*

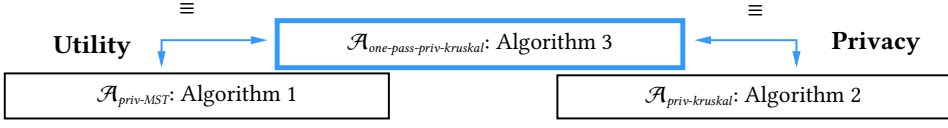


Fig. 1. Roadmap of Section 3. The figure outlines the proof structure: the utility guarantee is established for Algorithm 1, and the privacy guarantee is proven for Algorithm 2. A "bridging" algorithm (Algorithm 3) is introduced to demonstrate the equivalence of Algorithms 1 and 2 by showing they share the same output distribution.

3 Reduction from Private MST to Non-Private MST

In this section, we present a novel algorithmic framework (Algorithm 1) that enables a reduction from the private MST problem to the non-private one, thereby proving Theorem 1.1. A key feature of this framework is its independence from the chosen MST algorithm. Since the graph's topology is public, the most suitable algorithm can always be selected in advance. Notably, this framework facilitates an (expected) linear-time private MST algorithm.

Overview. Figure 1 outlines the roadmap of this section. In Section 3.1, we introduce a simple algorithmic framework that fulfills Theorem 1.1. The framework adds certain noise to each edge weight, then applies an arbitrary non-private MST algorithm to publish only the set of MST edges with respect to the noisy weights. Proving its time complexity and utility is straightforward, but its privacy guarantee requires more effort. Instead of proving privacy directly, we establish it through two equivalent algorithms. Section 3.2 describes a private version of Kruskal's algorithm [21] (Algorithm 2), which iteratively and privately selects the approximate MST edges. Its privacy analysis follows directly from the composition property of DP, though its utility analysis is more intricate. Finally, in Section 3.3, we present a "bridging" algorithm (Algorithm 3) that has the same output distribution as both Algorithm 1 and Algorithm 2, establishing their equivalence. This allows us to transfer privacy and utility guarantees between the algorithms, simplifying the proofs.

A detailed comparison of our algorithmic techniques with prior work, along with a broader discussion of the problem's background, is deferred to Section 5.

3.1 Reduction

The framework is outlined in Algorithm 1. It is straightforward to implement and requires only 3 lines of code using standard libraries. Given a privacy parameter ρ and an input graph $G = (V, E, \mathbf{W})$, it computes for each edge $e \in E$ a noisy weight $\tilde{w}_e \doteq w_e + (2/\varepsilon') \cdot \ln \text{Exp}(1)$, where $\text{Exp}(1)$ denotes a random variable following the exponential distribution (Definition 2.10). Finally, it computes the MST using an existing (non-private) MST algorithm $\mathcal{A}_{\text{MST}}$ on the noisy weights $\tilde{\mathbf{W}} = (\tilde{w}_1, \dots, \tilde{w}_m)$.

Algorithm 1 $\mathcal{A}_{\text{priv-MST}}$: Private MST Framework

Input: $G = (V, E, \mathbf{W})$, any MST algorithm $\mathcal{A}_{\text{MST}}$, privacy parameters (ε, δ)

- 1: $\rho \leftarrow (\sqrt{\varepsilon + \log(1/\delta)} - \sqrt{\log(1/\delta)})^2$; $\varepsilon' \leftarrow \sqrt{2 \cdot \rho / (n-1)}$;
 - 2: $\tilde{w}_e \leftarrow w_e + (2/\varepsilon') \cdot \ln(\text{Exp}(1))$ for all $e \in E$
 - 3: **return** $\mathcal{A}_{\text{MST}}(V, E, \tilde{\mathbf{W}})$
-

Extension to finding a maximum weight independent set in matroids: The framework extends to general matroids by perturbing each element's weight with similar noise and computing the maximum weight independent set using any suitable algorithm on the noisy weights, see Appendix A.

The properties of Algorithm 1 are outlined in Theorem 1.1.

Proof of Theorem 1.1. It remains to analyze its running time, privacy and utility guarantees.

Running Time. Adding noise to all edge weights requires only $O(m)$ time. Consequently, the overall running time of Algorithm 1 is given by $O(m + t_{\mathcal{A}_{MST}}(n, m))$, where $t_{\mathcal{A}_{MST}}(n, m)$ denotes the running time of the chosen MST algorithm $\mathcal{A}_{MST}$ for a graph with n vertices and m edges. For example, with the celebrated MST algorithm by Karger et al. [19], Algorithm 1 achieves an expected linear running time, where $t_{\mathcal{A}_{MST}}(n, m) \in O(n + m)$. Our reduction is compatible with any MST algorithm, even in parallel or distributed settings.

Utility Guarantee. We analyze the utility of Algorithm 1 before establishing the privacy guarantee, as the former is more straightforward. We need the following fact, whose proof is in Appendix C.

FACT 3.1. *Let $Z_i \sim \ln \text{Exp}(1)$ for $i \in [m]$ be independent. Then $\mathbb{E} [\max_{i \in [m]} |Z_i|] \in O(\ln m)$.*

Let T^* and T denote the MSTs with respect to the original edge weights $\mathbf{W}$ and the noisy edge weights $\tilde{\mathbf{W}}$, respectively. Note that T is precisely the spanning tree returned by Algorithm 1. We aim to prove that T satisfies the utility guarantee specified in Theorem 1.1.

By Fact 3.1 and noting that both T and T^* contain fewer than n edges, it follows that

$$\mathbb{E} [\max_{e \in E} |\tilde{w}_e - w_e|] \in O(1/\epsilon' \cdot \ln m) \implies \begin{aligned} \mathbb{E} [|w(T) - \tilde{w}(T)|] &\in O(n/\epsilon' \cdot \ln m). \\ \mathbb{E} [|w(T^*) - \tilde{w}(T^*)|] &\in O(n/\epsilon' \cdot \ln m). \end{aligned} \quad (8)$$

As T is the MST on the noisy weights $\tilde{\mathbf{W}}$, we have $\tilde{w}(T) \leq \tilde{w}(T^*)$. Therefore,

$$\begin{aligned} \mathbb{E} [w(T) - w(T^*)] &\leq \mathbb{E} [w(T) - \tilde{w}(T) + \tilde{w}(T^*) - w(T^*)] \\ &\leq \mathbb{E} [|w(T) - \tilde{w}(T)|] + \mathbb{E} [|\tilde{w}(T^*) - w(T^*)|] \in O(n/\epsilon' \cdot \ln m). \end{aligned} \quad (9)$$

Finally, to obtain the utility bound in Theorem 1.1, based on Algorithm 1, Line 1, we observe that

$$\frac{1}{\epsilon'} = \sqrt{\frac{n-1}{2\rho}} \in O\left(\frac{1}{\epsilon} \sqrt{n \log \frac{1}{\delta}}\right), \text{ since } \frac{1}{\sqrt{\rho}} = \frac{1}{\sqrt{\epsilon + \log \frac{1}{\delta} - \log \frac{1}{\delta}}} = \frac{\sqrt{\epsilon + \log \frac{1}{\delta} + \log \frac{1}{\delta}}}{\epsilon} \in O\left(\frac{1}{\epsilon} \sqrt{\log \frac{1}{\delta}}\right). \quad (10)$$

Substituting the expression for $1/\epsilon'$ into Equation (9) completes the proof.

Privacy Guarantee. As outlined in the *roadmap* and illustrated in Figure 1, directly proving the privacy guarantees is challenging. To address this, we introduce Algorithm 2 in Section 3.2, whose privacy guarantee can be easily established, and Algorithm 3 in Section 3.3, which shares the same output distribution as Algorithms 1 and 2, transferring the privacy guarantee to the former. $\square$

3.2 Private Kruskal Algorithm

Algorithm 2 presents a privatized version of Kruskal's algorithm. Compared to the non-private one [21], at each iteration (Line 3-6), instead of selecting the edge with minimum weight, it selects an approximately minimum-weight edge by the exponential mechanism [26] which samples an edge e with probability proportional to $e^{-(\epsilon'/2) \cdot w_e}$.

THEOREM 3.2 (PROPERTIES OF ALGORITHM 2). *Given an input graph $G = (V, E, \mathbf{W})$ with n vertices and m edges, Algorithm 2 runs in $O(m + n \log n)$ time, is (ϵ, δ) -DP, and returns an approximately minimum spanning tree $\mathcal{M}(G)$ such that, if T^* is a minimum spanning tree:*

$$\mathbb{E} [\sum_{e \in \mathcal{M}(G)} w_e] - \sum_{e \in T^*} w_e \in O\left(\frac{1}{\epsilon} \cdot n^{3/2} \cdot (\log n) \cdot \sqrt{\log(1/\delta)}\right). \quad (11)$$

Proof of Theorem 3.2. We analyze the running time, privacy and utility guarantees of Algorithm 2.

Algorithm 2 $\mathcal{A}_{\text{private-Kruskal}}$: Private Kruskal's Algorithm

Input: Graph $G = (V, E, \mathbf{W})$, privacy parameter (ε, δ)

- 1: $\rho \leftarrow (\sqrt{\varepsilon + \log(1/\delta)} - \sqrt{\log(1/\delta)})^2$; $\varepsilon' \leftarrow \sqrt{2 \cdot \rho / (n-1)}$;
 - 2: Initialize $T \leftarrow \emptyset, F \leftarrow E$
 - 3: **for** $i \leftarrow 1$ to $|V| - 1$ **do**
 - 4: Sample $e \in F$ with probability $\propto e^{-\frac{\varepsilon'}{2} \cdot w_e}$
 - 5: $T \leftarrow T \cup \{e\}$
 - 6: $F \leftarrow F \setminus \{e' \in F : \{e'\} \cup T \text{ contains a cycle}\}$
 - 7: **end for**
 - 8: **return** T
-

Running Time. Algorithm 2 can be implemented in $O(m + n \log n)$ time using a designated data structure. As it is not our primary focus, we refer for the details to Appendix E. It remains to analyze the privacy-utility trade-offs of Algorithm 2.

Privacy Guarantee. We leverage zCDP for its tighter composition bounds, so the proof involves conversion from DP to zCDP and back. We need to show that Algorithm 2 guarantees (ε, δ) -DP.

- Based on the privacy guarantee of the exponential mechanism (Fact 2.8), each round of Algorithm 2 is ε' -DP. Using the conversion from DP to zCDP (Fact 2.5) and the initialization of ε' (Line 1), this corresponds to ρ' -zCDP, where $\rho' = (\varepsilon')^2/2 = \rho/(n-1)$.
- Since the algorithm runs $n-1$ rounds, by the composition property (Fact 2.6), Algorithm 2 satisfies $(n-1) \cdot \rho' = \rho$ -zCDP.
- Applying Fact 2.5 in reverse (from zCDP to DP), we conclude that the algorithm satisfies $(\rho + 2 \cdot \sqrt{\rho \log(1/\delta)}, \delta)$ -DP. Substituting ρ with $(\sqrt{\varepsilon + \log(1/\delta)} - \sqrt{\log(1/\delta)})^2$ yields $\rho + 2 \cdot \sqrt{\rho \log(1/\delta)} = \varepsilon$.

Utility Guarantee. The utility analysis is more involved, so we defer it to the next section, where we introduce a simplified implementation of Algorithm 2 and establish its equivalence to Algorithm 1. This allows the straightforward utility analysis of Algorithm 1 to transfer directly to Algorithm 2, demonstrating the benefit of proving the algorithms' equivalence.

It is worth noting that the utility guarantee of Algorithm 2 can be proven directly, although it is much more complicated and requires a much deeper insight into Kruskal's algorithm. A proof is provided in Appendix D for reference.

□

3.3 One-Pass Private Kruskal

Finally, we introduce the last private MST algorithm. It achieves the same running time as Algorithm 2, but features a significantly simplified structure, allowing it to be implemented using existing libraries for non-private Kruskal's algorithm. More importantly, it serves as a bridge to establish the equivalence between Algorithm 1 and Algorithm 2.

The algorithm is outlined in Algorithm 3. It begins by generating noisy edge weights using $\tilde{w}_e \doteq \text{Exp}(1) / e^{-(\varepsilon'/2) w_e}$ and then applies the non-private Kruskal's algorithm to the noisy weights.

THEOREM 3.3. *Algorithm 3 shares the properties of Algorithm 2 as stated in Theorem 3.2.*

Proof of Theorem 3.3. We analyze the runtime, utility, and privacy guarantees of Algorithm 3. In the process, we establish the equivalence among Algorithm 1, Algorithm 2, and Algorithm 3, i.e., they have the same output distributions.

Algorithm 3 $\mathcal{A}_{\text{one-pass-private-Kruskal}}$ **Input:** Graph $G = (V, E, \mathbf{W})$, privacy parameter (ϵ, δ)

- 1: $\rho \leftarrow (\sqrt{\epsilon + \log(1/\delta)} - \sqrt{\log(1/\delta)})^2$; $\epsilon' \leftarrow \sqrt{2 \cdot \rho / (n-1)}$;
- 2: Initialize $T \leftarrow \emptyset$, $F \leftarrow E$
- 3: $\tilde{w}_e \leftarrow \text{Exp}(1) / e^{-\frac{\epsilon'}{2} \cdot w_e}$ $\triangleright \text{Exp}(1)$ represents exponential noise
- 4: **for** $i \leftarrow 1$ to $|V| - 1$ **do**
- 5: $e \leftarrow \arg \min_{e \in F} \tilde{w}_e$
- 6: $T \leftarrow T \cup \{e\}$
- 7: $F \leftarrow F \setminus \{e' \in F : \{e'\} \cup T \text{ contains a cycle}\}$
- 8: **end for**
- 9: **return** T

Running Time. The running time is dominated by Kruskal's algorithm, which is $O(n + m \log n)$.

Utility Guarantee. The utility guarantee is established through the equivalence between Algorithm 1 and Algorithm 3, allowing the utility guarantee of the former to transfer to the latter.

First, observe that in line 5 of Algorithm 3, the returned edge satisfies

$$\arg \min_{e \in F} \tilde{w}_e = \arg \min_{e \in F} (w_e + \frac{2}{\epsilon'} \cdot \ln \text{Exp}(1)). \quad (12)$$

Hence, we can replace the noisy weight $\tilde{w}_e \doteq \text{Exp}(1) / e^{-\frac{\epsilon'}{2} \cdot w_e}$ in Algorithm 3 by $\tilde{w}_e \doteq w_e + \frac{2}{\epsilon'} \cdot \ln \text{Exp}(1)$. Second, since $\frac{2}{\epsilon'} \cdot \ln \text{Exp}(1)$ is a continuous random variable, the MST on the noisy graph is unique with probability 1. Consequently, *any* MST algorithm computes the same solution, thereby establishing the equivalence between Algorithm 1 and Algorithm 3.

Privacy Guarantee. The privacy guarantee is established through the equivalence between Algorithms 2 and 3, allowing the privacy guarantee of the former to transfer to the latter.

To establish the equivalence, we introduce a generalized model, *Probability Proportional to Sizes with Adaptive Candidate Removal (PPSACR)*, for top- k sampling, along with a new sampling technique for this model. As will be shown, Algorithm 2 can be viewed as a special case of PPSACR, and Algorithm 3 directly implements the new sampling technique, thereby demonstrating its equivalence to Algorithm 2. It is worth noting that the equivalence between Algorithms 2 and 3 can also be established directly, without relying on PPSACR. However, we choose to present PPSACR here, as it is of independent interest and may have broader applications to other problems.

PPSACR Model: The model is described in Algorithm 4. Let U be a finite set of items, indexed from 1 to $|U|$. Each item $j \in U$ is associated with a weight $s(j) \in \mathbb{R}^+$. PPSACR iteratively samples k items. It maintains a candidate set C , initialized as U , and a selected item list $\mathcal{I}$, which is initially empty. At each step, an item $j \in C$ is sampled with probability proportional to $s(j)$ and appended to $\mathcal{I}$. Afterward, the sampled item j and an additional subset of items, denoted by $f(\mathcal{I})$, are removed from C . Here, $f : (\bigcup_{i \in [k]} U^i) \rightarrow 2^U$ is a function that determines the additional items to be removed based on the sequence of selected items so far. Since the set $U \setminus C$ at each step is fully determined by $\mathcal{I}$ and f , the function f , given $\mathcal{I}$ as input, effectively encodes the information about $U \setminus C$.

EXAMPLE 3.4. Algorithm 2 can be interpreted as a special case of this sampling model, where U represents the set of edges E , $s(e)$ is the exponential mechanism sampling weight $e^{-\lambda \cdot w_e^*}$ for each $e \in E$, k is $n - 1$ (the number of edges in a spanning tree), and f is the function that removes edges forming cycles based on the previously selected edges.

Algorithm 4 PPSACR

```

1:  $C \leftarrow U$ ,  $\mathcal{I} \leftarrow$  An empty list  $\triangleright$  possible candidates and sampled items so far
2: while  $|\mathcal{I}| \leq k$  and  $C \neq \emptyset$  do
3:   Sample  $j \in C$  with probability proportional to  $s(j)$ 
4:   Add  $j$  to the back of  $\mathcal{I}$ 
5:    $C \leftarrow C \setminus (\{j\} \cup f(\mathcal{I}))$ 
6: end while
7: return  $\mathcal{I}$ 

```

THEOREM 3.5 (*One-Shot-PPSACR*, ALGORITHM 6). *Algorithm 4 can be equivalently implemented with the following modifications:*

- (1) Add a noisy weights generating step before line 1: $\tilde{s}(j) \leftarrow \text{Exp}(1) / s(j), \forall j \in U$.
- (2) Replace line 3 with $j \leftarrow \arg \min_{j' \in C} \tilde{s}(j')$.

The modified algorithm is referred to as *One-Shot-PPSACR*.

Due to a cleaner exposition, the pseudo codes of the modified algorithm (Algorithm 6) is given in Appendix B. Note that if Algorithm 2 is viewed as a special case of PPSACR, as described in Example 3.4, then Algorithm 3 directly implements the modifications proposed in Theorem 3.5. Therefore, Theorem 3.5 immediately establishes the equivalence between Algorithms 2 and 3.

To proceed, we require the following properties of the exponential distribution, the proofs of which are included in Appendix C.

FACT 3.6 (SCALING). *If $X \sim \text{Exp}(1)$, then for all $\lambda > 0$, then $Y \doteq X/\lambda$ has distribution $\text{Exp}(\lambda)$.*

FACT 3.7 (MINIMUM). *If $X_1 \sim \text{Exp}(\lambda_1), \dots, X_d \sim \text{Exp}(\lambda_d)$, then $\Pr[X_i = \min_{j \in [d]} X_j] = \frac{\lambda_i}{\sum_{j \in [d]} \lambda_j}$.*

FACT 3.8 (MEMORYLESS). *If $X \sim \text{Exp}(\lambda)$, then $\Pr[X \geq x + y \mid X \geq x] = \Pr[X \geq y], \forall x, y \geq 0$.*

PROOF OF THEOREM 3.5. To simplify the discussion, we assume that k sampled items are returned by PPSACR. The proof for the case when less than k items are returned are similar. Let $\mathcal{J} \doteq (j_1, j_2, \dots, j_k) \in U^k$ be a feasible output sequence of PPSACR. We will prove that *One-Shot-PPSACR* outputs $\mathcal{J}$ with the same probability as PPSACR.

PPSACR: For each $i \in [k]$, denote C_i the set of candidates to be sampled in Algorithm 4, Line 3 during the i -th iteration. The probability of selecting j_i is $s(j_i) / (\sum_{j \in C_i} s(j))$. Thus, the probability that PPSACR outputs $\mathcal{J}$ is

$$\frac{s(j_1)}{\sum_{j \in C_1} s(j)} \cdot \frac{s(j_2)}{\sum_{j \in C_2} s(j)} \cdots \frac{s(j_k)}{\sum_{j \in C_k} s(j)}. \quad (13)$$

One-Shot-PPSACR: For each $i \in [k]$, let $\mathcal{E}_i$ be the event that $\tilde{s}(j_i) = \min_{j \in C_i} \tilde{s}(j)$, and define $\mathcal{E}_{1:i} \doteq \mathcal{E}_1 \wedge \dots \wedge \mathcal{E}_i$. First, by Fact 3.6, for each $j \in U$, $\tilde{s}(j) \doteq \text{Exp}(1) / s(j)$ follows distribution $\text{Exp}(s(j))$. Based on Fact 3.7, the minimum property of a collection of exponential random variables, it holds that

$$\Pr[\mathcal{E}_1] = \frac{s(j_1)}{\sum_{j \in C_1} s(j)}. \quad (14)$$

Second, let $z_1 \leq z_2 \leq \dots \leq z_k$ be a feasible realization of $\tilde{s}(j_1), \dots, \tilde{s}(j_k)$. For each $i \in [2 \dots k]$, define $\tilde{s}(j_{1:i-1}) \doteq (\tilde{s}(j_1), \dots, \tilde{s}(j_{i-1}))$ and $z_{1:i-1} = (z_1, \dots, z_{i-1})$. Conditioned on the events $\mathcal{E}_{1:i-1}$ and $\tilde{s}(j_{1:i-1}) = z_{1:i-1}$, we know that for each $j \in C_i$, $\tilde{s}(j) \geq (\max z_{1:i-1}) = z_{i-1}$. Since the $\tilde{s}(j)$ are independent, by the memoryless property of the exponential distribution (Fact 3.8), $\tilde{s}(j) - z_{i-1}$ still follows the distribution $\text{Exp}(s(j))$. Together with the fact that the minimum is invariant under

a constant shift, the event $\mathcal{E}_i$ is now equivalent to $\tilde{s}(j_i) - z_{i-1} = \min_{j \in C_i} (\tilde{s}(j) - z_{i-1})$, applying Fact 3.7 again yields:

$$\begin{aligned} \Pr [\mathcal{E}_i \mid \mathcal{E}_{1:i-1}, \tilde{s}(j_{1:i-1}) = z_{1:i-1}] &= \Pr \left[\tilde{s}(j_i) - z_{i-1} = \min_{j \in C_i} (\tilde{s}(j) - z_{i-1}) \mid \mathcal{E}_{1:i-1}, \tilde{s}(j_{1:i-1}) = z_{1:i-1} \right] \\ &= \frac{s(j_i)}{\sum_{j \in C_i} s(j)}. \end{aligned}$$

Taking expectation over $\tilde{s}(j_{1:i-1})$ gives

$$\mathbb{E}_{\tilde{s}(j_{1:i-1})} [\Pr [\mathcal{E}_i \mid \mathcal{E}_{1:i-1}, \tilde{s}(j_{1:i-1})]] = \Pr [\mathcal{E}_i \mid \mathcal{E}_{1:i-1}] = \frac{s(j_i)}{\sum_{j \in C_i} s(j)}.$$

Finally, based on chain rule of probability, $\mathcal{E}_{1:k}$ has exactly the same probability as Equation (13):

$$\Pr [\mathcal{E}_{1:k}] = \Pr [\mathcal{E}_1] \cdot \prod_{i=2}^k \Pr [\mathcal{E}_i \mid \mathcal{E}_{1:i-1}] = \frac{s(j_1)}{\sum_{j \in C_1} s(j)} \cdot \frac{s(j_2)}{\sum_{j \in C_2} s(j)} \cdots \frac{s(j_k)}{\sum_{j \in C_k} s(j)}. \quad (15)$$

□

4 Lower Bound for Approximate DP with ℓ_∞ Neighboring Relationship

In this section, we prove the lower bound stated in Theorem 1.2. Our proof consists of two steps: in Section 4.1, we present a simple reduction from (ε, δ) -DP MST algorithms to $(1, \delta)$ -DP MST algorithms to facilitate the derivation of lower bounds; in Section 4.2, we present a lower bound for all $(1, \delta)$ -DP MST algorithms, which can be extended to (ε, δ) -DP lower bound using the previous reduction.

4.1 Reduction from (ε, δ) -DP to $(1, \delta)$ -DP

LEMMA 4.1. Assume that $\varepsilon < 1$. Suppose that there is an $(\varepsilon, \frac{e^\varepsilon - 1}{e - 1} \cdot \delta)$ -DP MST algorithm $\mathcal{M}$ under the ℓ_∞ neighboring relationship, such that for every input graph $G = (V, E, \mathbf{W})$ (with the MST denoted by T^*), it holds that

$$\mathbb{E}_{\mathcal{M}} \left[\sum_{e \in \mathcal{M}(G)} w_e \right] - \sum_{e \in T^*} w_e \in o\left((1/\varepsilon) \cdot n^{3/2} \cdot \ln n\right). \quad (16)$$

Then there exists a $(1, \delta)$ -DP MST algorithm $\mathcal{M}'$, such that for every input graph $G' = (V', E', W')$ (with the MST denoted by T'), it holds that

$$\mathbb{E}_{\mathcal{M}'} \left[\sum_{e \in \mathcal{M}'(G')} w'_e \right] - \sum_{e \in T'} w'_e \in o(n^{3/2} \cdot \ln n), \quad (17)$$

PROOF. We demonstrate how to construct $\mathcal{M}'$ based on $\mathcal{M}$. Given a graph $G' = (V', E', W')$, we create a new graph $G = (V, E, W)$ as follows: $V = V'$, $E = E'$, and $W = (1/\varepsilon) \cdot W'$. To simplify the discussion, we assume that $1/\varepsilon \in \mathbb{N}^+$; otherwise, we replace $1/\varepsilon$ with $\lceil 1/\varepsilon \rceil$, which affects the privacy and utility guarantees only by a constant factor, as $\varepsilon \leq 1$. Based on this construction, we have $T' = T^*$. Finally, we define $\mathcal{M}'$ as: $\mathcal{M}'(G') \doteq \mathcal{M}(G)$. Then based on group privacy property (Fact 2.7), $\mathcal{M}'$ is $\left(\varepsilon \cdot \frac{1}{\varepsilon}, \frac{e^{\frac{1}{\varepsilon} \cdot \varepsilon} - 1}{e^{\varepsilon} - 1} \cdot \frac{e^\varepsilon - 1}{e - 1} \cdot \delta \right) = (1, \delta)$ -DP. We conclude the proof by observing that

$$\begin{aligned} \mathbb{E}_{\mathcal{M}'} \left[\sum_{e \in \mathcal{M}'(G')} w'_e \right] &= \mathbb{E}_{\mathcal{M}} \left[\sum_{e \in \mathcal{M}(G)} w'_e \right] = \mathbb{E}_{\mathcal{M}} \left[\sum_{e \in \mathcal{M}(G)} \varepsilon \cdot w_e \right] = \varepsilon \cdot \mathbb{E}_{\mathcal{M}} \left[\sum_{e \in \mathcal{M}(G)} w_e \right] \\ &\in \varepsilon \cdot \left(\sum_{e \in T^*} w_e + o\left((1/\varepsilon) \cdot n^{3/2} \cdot \ln n\right) \right) = \sum_{e \in T'} w'_e + o(n^{3/2} \cdot \ln n). \end{aligned} \quad (18)$$

□

4.2 $(1, \delta)$ -DP Lower Bound

In this subsection, we show that no $(1, \delta)$ -DP algorithm $\mathcal{M}$ satisfies Equation (17) for $\delta = O(1/\sqrt{n})$, thereby proving Theorem 1.2. Our proof heavily relies on a technique originally developed by Steinke and Ullman [38] for establishing lower bounds for the private maximum top- k selection problem. To apply this technique, we negate the weights of the input graph. Instead of proving a lower bound for the minimum spanning tree problem, we demonstrate that any $(1, \delta)$ -DP maximum spanning tree algorithm $\mathcal{M}$ incurs an error of $\Omega(n^{3/2} \log n)$ for $\delta = O(1/\sqrt{n})$.

THEOREM 4.2 (THEOREM 3 IN [38]). *Let $\beta, \gamma, \Delta, k > 0$ and $s, d \in \mathbb{N}^+$ be a fixed set of parameters. Let $P = (P_1, \dots, P_d)$ be independent samples from the beta distribution $\text{Beta}(\beta, \beta)$, and let $X \in \{0, 1\}^{s \times d}$ be a random dataset such that every $X_{i,j}$ is an independent sample from Bernoulli distribution with mean P_j for every $i \in [s]$ and $j \in [d]$.*

Let $\mathcal{A} : \{0, 1\}^{s \times d} \rightarrow \mathbb{R}^d$ be a $(1, \beta\gamma k/(s\Delta))$ -differentially private algorithm (where two datasets X and X' are neighboring if and only if they differ by at most one row). Assume $\mathbb{E}_{P,X,\mathcal{A}} [\|\mathcal{A}(X)\|_2^2] = k$ and $\Pr [\|\mathcal{A}(X)\|_1 \leq \Delta] = 1$ and

$$\mathbb{E}_{P,X,\mathcal{A}} \left[\sum_{j \in [d]} \mathcal{A}(X)_j \cdot \left(P_j - \frac{1}{2}\right) \right] \geq \gamma k, \quad (19)$$

then $s \geq \beta\gamma\sqrt{k}$.

Informally speaking, from Theorem 4.2, any DP algorithm with a good utility requires a lot of samples. To translate the lower bound on samples to a lower bound on the utility for MST, our goal is to construct a hard distribution of graphs, where the weight vector simulates the random dataset X and the algorithm's utility ties to the same quantity in the theorem. The proof is by contradiction. When the weight vector, or equivalently, the number of samples, is fixed, Theorem 4.2 shows that any DP algorithm with a "good" utility requires more samples than this fixed number, leading to a contradiction. We will present the proof formally in the rest of this section.

DEFINITION 4.3 (HARD DISTRIBUTION). *Consider a complete graph G with n -vertices and let $m = n(n-1)/2$ be the number of edges. Let $\beta, \gamma > 0$ and $s \in \mathbb{N}^+$ be parameters to be determined later. For each edge e , we first sample P_e from $\text{Beta}(\beta, \beta)$ independently. Then let $X \in \{0, 1\}^{s \times m}$ be a random dataset such that $X_{i,e}$ is an independent sample from Bernoulli distribution with mean P_e , $\forall i \in [s], e \in [m]$. Finally, let $w_e \doteq \sum_{i \in [s]} X_{i,e}$ for each $e \in [m]$. Hence, w_e follows binomial distribution $\mathcal{B}(s, P_e)$.*

Since a spanning tree algorithm $\mathcal{M}$ has a different input and output format from the algorithm in Theorem 4.2, a conversion is required.

DEFINITION 4.4 (CONVERSION). *Let G be the random graph and X the random dataset generated as described in Definition 4.3. Given an algorithm $\mathcal{M}$ that takes G as input and outputs a set of spanning tree edges, define $\mathcal{A}_{\mathcal{M}}$ as the algorithm that, given input X , outputs an indicator vector $\mathcal{A}_{\mathcal{M}}(X) \in \{0, 1\}^{|E|}$, where $\mathcal{A}_{\mathcal{M}}(X)_e = 1$ if and only if $e \in \mathcal{M}(G)$.*

It is straightforward to see that if $\mathcal{M}$ is $(1, \delta)$ -DP (where two input graphs are considered neighboring if the weights of each edge differ by at most 1), then $\mathcal{A}_{\mathcal{M}}$ is also $(1, \delta)$ -DP (where two input datasets are neighboring if they differ in at most one row). Therefore, under the hard distribution, the sample complexity lower bound in Theorem 4.2 for the algorithm $\mathcal{A}_{\mathcal{M}}$ transfer to $\mathcal{M}$.

Before presenting our main lower bound, we state a lemma that is critical to our proof.

LEMMA 4.5. *For $s \geq 10$, $\beta = \frac{1}{2} \ln n$ and $n \geq 2 \times 10^7$, let G be a random graph sampled from the hard distribution in Definition 4.3. With probability at least 0.99, there exists a spanning tree of G where each edge has weight at least $3/4 \cdot s$.*

The complete proof of Lemma 4.5 is included in Appendix F. Intuitively, the result follows from the properties of the beta distribution and the Erdős–Rényi random graph model [12]. Let H be the subgraph of G that includes all vertices and all edges e such that $w_e \geq \frac{3}{4} \cdot s$. Since the edge weights are independent and due to our choice of β , the graph H is an instance of the Erdős–Rényi model with edge sampling probability $p = \Pr [w_e \geq s \cdot 3/4] \in \Omega(\ln(n)/n)$. Finally, G has a spanning tree where all edges e satisfy $w_e \geq s \cdot 3/4$ if and only if H is connected, which occurs almost surely when $p \in \Omega(\ln(n)/n)$ [12].

Assume Lemma 4.5 holds, we directly have

$$\mathbb{E}_G [\sum_{e \in T^*} w_e] \geq 0.99 \cdot \frac{3}{4} \cdot s \cdot (n-1). \quad (20)$$

where T^* be the maximum spanning tree in G . Next, we will present the main theorem in this section that captures the utility lower bound.

THEOREM 4.6 (MST LOWER-BOUND). *Let $n \geq 9 \times 10^8$, $\beta = \frac{1}{2} \ln n$ and $s = \frac{1}{100} \sqrt{n} \ln n$, G be a random graph sampled from the hard distribution in Definition 4.3. Let $\gamma = 0.04$ and $\delta = \beta\gamma/s = 2/\sqrt{n}$. For any $(1, \delta)$ -differentially private maximum spanning tree algorithm $\mathcal{M}$, it holds that*

$$\mathbb{E}_{G, \mathcal{M}} [\sum_{e \in M(G)} w_e] \leq \mathbb{E}_G [\sum_{e \in T_G^*} w_e] - \frac{n^{3/2} \ln n}{1000}, \quad (21)$$

where T_G^* be the maximum spanning tree given G .

PROOF. The proof proceeds by contradiction. Assume there exists a $(1, \delta)$ -DP maximum spanning tree algorithm $\mathcal{M}$, s.t.

$$\mathbb{E}_{G, \mathcal{M}} [\sum_{e \in M(G)} w_e] > \mathbb{E}_G [\sum_{e \in T_G^*} w_e] - \frac{n^{3/2} \ln n}{1000}, \quad (22)$$

Let X be the random dataset generated when G is constructed, as described in Definition 4.3, and let $\mathcal{A}_M$ denote the algorithm converted from $\mathcal{M}$, as described in Definition 4.4. We will prove that

$$\mathbb{E}_{P, X, \mathcal{A}_M} [\sum_{e \in [m]} (\mathcal{A}_M(X))_e (P_e - \frac{1}{2})] > \gamma(n-1), \quad (23)$$

Since $\mathcal{M}$ outputs $(n-1)$ edges, we always have $\|\mathcal{A}_M(X)\|_2^2 = \|\mathcal{A}_M(X)\|_1 = n-1$. Hence, $\mathcal{A}_M$ satisfies all conditions in Theorem 4.2, where we set $k = \Delta = (n-1)$ and $\delta = \beta\gamma k/(s\Delta) = \beta\gamma/s$. Therefore, Theorem 4.2 implies that $s \geq \beta\gamma\sqrt{n-1}$, leading to a contradiction since the assumption in Theorem 4.6 implies $s = 0.01\sqrt{n} \ln n < \beta\gamma\sqrt{n-1}$.

Proving Equation (23): Since $w_e \sim \mathcal{B}(s, P_e)$, we have $\mathbb{E}[w_e] = s \cdot P_e$. Further, the following lemma holds, whose proof employs a standard technique using Hoeffding's inequality and is deferred to Appendix G.

$$\text{LEMMA 4.7. } \mathbb{E} [\max_{e \in E} |w_e - s \cdot P_e|] \leq \sqrt{3s \ln n}.$$

Combing Lemma 4.7 and that $\|\mathcal{A}_M(X)\|_1 \equiv n-1$, we see

$$\mathbb{E}_{P, X, \mathcal{A}_M} [\sum_{e \in [m]} (\mathcal{A}_M(X))_e (s \cdot P_e - \frac{s}{2})] \geq \mathbb{E}_{P, X, \mathcal{A}_M} [\sum_{e \in [m]} (\mathcal{A}_M(X))_e w_e] - (n-1)(\sqrt{3s \ln n} + \frac{s}{2}) \quad (24)$$

By the construction of $\mathcal{A}_M$ (Definition 4.4), along with the assumption that $\mathcal{M}$ violates Equation (21),

$$\mathbb{E}_{P, X, \mathcal{A}_M} [\sum_{e \in [m]} (\mathcal{A}_M(X))_e w_e] = \mathbb{E}_{G, \mathcal{M}} [\sum_{e \in M(G)} w_e] > \mathbb{E}_G [\sum_{e \in T_G^*} w_e] - \frac{n^{3/2} \ln n}{1000}. \quad (25)$$

Hence

$$\begin{aligned}
\mathbb{E}_{P, X, \mathcal{A}_M} \left[\sum_{e \in [m]} (\mathcal{A}_M(X))_e (s \cdot P_e - \frac{s}{2}) \right] &> \mathbb{E}_G \left[\sum_{e \in T_G^*} w_e \right] - \frac{n^{3/2} \ln n}{1000} - (n-1)(\sqrt{3s \ln n} + \frac{s}{2}) \\
&> 0.99 \cdot \frac{3}{4} s(n-1) - \frac{n^{3/2} \ln n}{1000} - (n-1)(\sqrt{3s \ln n} + \frac{s}{2}) \\
&= 0.2425s(n-1) - \frac{n^{3/2} \ln n}{1000} - (n-1)\sqrt{3s \ln n}.
\end{aligned}$$

Since $n \geq 9 \times 10^8$, $s = \frac{1}{100} \sqrt{n \ln n}$ and $\gamma = 0.04$, it follows that

$$\mathbb{E}_{P, X, \mathcal{A}_M} \left[\sum_{e \in [m]} (\mathcal{A}_M(X))_e (P_e - \frac{1}{2}) \right] > 0.2424(n-1) - \frac{1}{s} \left((n-1)\sqrt{3s \ln n} + \frac{n^{3/2} \ln n}{1000} \right) > \gamma(n-1).$$

□

5 Related Work

Releasing various graph statistics under differential privacy constraints is a fundamental and well-studied task. For more background, see [23, 27].

5.1 Top- k Selection.

The private MST problem is closely related to the private top- k selection problem. Formulated in the framework of this paper, the problem involves m items with weights $w_1, \dots, w_m$, and the goal of private top- k selection is to privately sample k items with approximately maximum or minimum weights. Notably, maximization and minimization are equivalent by flipping the sign of all weights. The private MST problem can be seen as a variant of top- $(n-1)$ selection with the additional topological constraint that the selected items must form a spanning tree.

Durfee and Rogers [8] introduced the first linear-time algorithm for private top- k selection with asymptotically optimal privacy-utility trade-off under approximate DP. They showed that iteratively applying the exponential mechanism [26] to select k items with an approximately maximum weight, is equivalent to adding random noise following the Gumbel distribution (see Definition C.1) to each item weight and then returning the k items with maximum noisy weights. This technique has a rich research history in the context of the non-private top- k problem [7, 28, 36, 42], appearing under various names. For the top-1 maximum selection problem, Yellott [42] showed very early that adding Gumbel noise to the $(\ln w_i)$'s and selecting the maximum allows sampling an item with probability proportional to their weights. Rosén [36] studied the top- k maximum selection problem under the name of "order sampling" and proposed a method to sample k items with probability proportional to their weights, by generating noisy scores $\text{Exp}(w_i) = \text{Exp}(1) / w_i$ for $i \in [m]$ and selecting k items with the minimum scores. This is equivalent to finding the k items maximizing $-\ln \text{Exp}(1) + \ln w_i$, where $-\ln \text{Exp}(1)$ follows exactly a Gumbel distribution (see Fact C.2).

Our sampling model, PPSACR, and the corresponding technique in Algorithm 4, extend this line of research [7, 36, 42] by enabling the adaptive removal of candidate items after each sampling step, based on previously selected items.

Qiao et al. [34] proposed a linear-time private top- k selection algorithm that achieves an asymptotically optimal privacy-utility trade-off (up to a logarithmic factor) under approximate differential privacy. Their approach involves adding independent Laplace noise to each item weight and returning the set of top- k items with the smallest noisy weights. However, their current privacy-utility trade-off analysis heavily depends on the returned items corresponding to the true k smallest noisy weights. Extending their approach and analysis to the private MST problem remains open, as the selected items or edges may not necessarily form a spanning tree.

Reference	PN	NH	Error	Time	Technique
[37]	ε -DP	ℓ_1	$O((1/\varepsilon) \cdot n \log n)$	$MST + O(m)$	Input privatization
[31]	ε -DP	ℓ_1	$O((1/\varepsilon) \cdot n \log n)$	$O(nm)$	In-place noise
[16]	ε -DP	ℓ_1	$\Omega((1/\varepsilon) \cdot n \log n)$	–	Lower bound
[37]	(ε, δ) -DP	ℓ_1	$O\left((1/\varepsilon) \cdot n \sqrt{(\log n) \cdot \log(1/\delta)}\right)$	$MST + O(m)$	Input privatization
[31]	(ε, δ) -DP	ℓ_1	$O\left((1/\varepsilon) \cdot n \sqrt{(\log n) \cdot \log(1/\delta)}\right)$	$O(nm)$	In-place noise
[37]	(ε, δ) -DP	ℓ_1	$\Omega(n)$	–	Lower bound
[37]	ε -DP	ℓ_∞	$O((1/\varepsilon) \cdot nm \log n)$	$MST + O(m)$	Input privatization
[31]	ε -DP	ℓ_∞	$O((1/\varepsilon) \cdot n^2 \log n)$	$O(nm)$	In-place noise
[16]	ε -DP	ℓ_∞	$\Omega((1/\varepsilon) \cdot n^2 \log n)$	–	Lower bound

Table 2. Extended landscape of results and complementing Table 1 with further known results for a graph with n vertices and m edges. **PN** and **NH** denote the Privacy Notation and the Neighboring Relationship respectively. The table compares the previous works [16, 31, 37] for both the ℓ_1 and ℓ_∞ neighborhood relation. “MST” in the **Time** column is the running time of any non-private MST algorithm.

5.2 Releasing an MST under DP

We supplement results for *approximate DP* (Table 1) with results for *pure DP* in Table 2. Before our work there were two approaches to private MST: *input privatization* and *in-place* algorithms [32] that we review next.

In-place. Both Prim-Jarník’s and Kruskal’s algorithm [17, 21, 33] start with an empty set of edges and then iteratively grow it while guaranteeing that it is still a subset of an MST. In each step, they greedily select the lightest new edge between cuts that respect the edges already chosen. One can privatize them by injecting noise whenever a weight is accessed during the computation of an MST. We can replace the selection step using any differentially private selection mechanism, for instance, *Report-Noisy-Max* [10], *Permute-and-Flip* [25], or the *Exponential Mechanism* [26] and get overall privacy by composition. PAMST [31] is based on the Prim-Jarník algorithm and gives the same asymptotic utility as our approach. Furthermore, a private version of Kruskal’s algorithm has been suggested by McKenna et al. [24] as a subroutine in privately generating synthetic data using a probabilistic graphical model, but its utility has never been discussed. We provide such an analysis in Section 3.2. Compared to the *input privatization* in the next paragraph [37], this approach gives strictly better utility under the ℓ_∞ neighboring relationship, assuming the graph is not too sparse. Unfortunately, the caveat is the running time because of the cost of the noisy selection. In an unpublished manuscript, Pagh and Retschmeier [2024] brought down the running time to $O(m + n^{3/2} \log(n) / \sqrt{\rho})$ by designing a special priority queue for Prim’s algorithm, which simulates *Report-Noisy-Max* in sublinear time $O(\sqrt{n/\rho} \log n)$, where ρ can be chosen as in Algorithm 1 to ensure (ε, δ) -DP.

Input privatization. One simple idea is to release a private synthetic graph by adding noise to *all* the edge weights and obtain privacy by post-processing. Sealfon [37] was the first to analyze *input perturbation* using Laplace noise to achieve ε -DP under the ℓ_1 neighborhood and gave error of $O(n \log n)$ which is known to be asymptotically optimal [16]. However, under ε -DP with ℓ_∞ neighboring relationship, this technique gives an additive error of $O(nm \log n)$. For dense graphs, this can leave a gap of up to a factor of $O(n)$ to the known lower bound of $\Omega(n^2 \log n)$ under ε -DP. The advantage is that it allows flexibility in choosing any (non-private) MST algorithm e.g. the expected linear time algorithm by Karger et al. [19], the deterministic near-linear time algorithm by Chazelle [5], or the deterministic linear time for dense graphs by Fredman and Tarjan [13]. The overall running time is compounded by the $O(m)$ time it takes to add fresh noise to each edge and the running time of the chosen MST algorithm. Another advantage is that releasing a single private

synthetic graph simultaneously allows the computation of other graph statistics, such as finding *shortest paths* or *minimum-weight (perfect) matchings* [37].

Lower Bounds. Recent work by Hladík and Tětek [16] showed tight asymptotical worst-case bounds for ϵ -DP using a packing argument: $\Omega(n \log n / \epsilon)$ for the ℓ_1 neighborhood and $\Omega(n^2 \log n / \epsilon)$ for ℓ_∞ . They improved Sealfon's ℓ_1 bound of $\Omega(n)$. Under the ℓ_∞ neighborhood, we show in Theorem 1.1 together with Theorem 1.2, that an expected error of $\tilde{\Theta}(n^{3/2})$ is asymptotically tight.

6 Empirical Evaluation

To confirm our theoretical claims, we implemented PAMST [31], Sealfon's input privatization [37], and our Algorithm 1 (instantiated with Prim-Jarník) in *Python 3.9*. The implementation relies on version 3.2.1 of the *NetworkX* library [14]. All experiments ran locally on a MacBook Pro with an Apple M2 Pro processor (10 Cores, up to 3.7GHz) and 16GB of RAM. We present two experiments that support our theoretical claims. The first experiment resembles a natural setting in the context of synthetic data generation, and the second explores the influence of the graph's density. The experiments indicate that the output distribution of our algorithm indeed matches PAMST and, hence, outperforms the *input privatization* approach if the graph is not too sparse. The results are shown in Figure 2, and we refer for more details to Appendix H.

Mutual Information. One natural setting is finding the Chow-Liu tree [6], which is the MST on the graph encoding the negated mutual information matrix on all pairwise attributes.¹ Let $X_1, \dots, X_n \in \{0, 1\}$ be random bits. We draw $X_1 \sim \text{Ber}(1/2)$, and then recursively X_i by flipping the bit X_{i-1} with probability $0 < p < 1/2$. This simulates a natural scenario where there is mutual information between X_i and X_{i-1} controlled by p . The mst is formed by the edges on the path $P = (X_1, \dots, X_n)$ (visualized in Figure 3). A potential underlying dataset of size d has the sensitivity of mutual information, which is $\Delta_{mi} = \log_2(d)/d$. As later shown in Appendix H, for this process the pairwise mutual information between all X_i and X_j define a complete graph $G = (V, E)$ with weights

$$\forall i, j \in [n], i \neq j : w_{ij} = \frac{1}{2} (p_1 \log_2(p_1) + p_2 \log_2(p_2))$$

where $p_1 = 1 + (1 - 2p)^k$ and $p_2 = 1 - (1 - 2p)^k$. In the experiment, we used $n = 1000$ with flip probability of $p = 0.05$. We set the underlying dataset size to $d = 10^5$, thus $\Delta_\infty = \Delta_{mi} \approx 0.00133$. The results are shown in Figure 2-a). We can see that the error of our approach closely resembles PAMST, outperforming the input privatization approach.

The Effect of the density. The second experiment follows the setup proposed in [31]. We sample an Erdős-Rényi random graph $\mathcal{G}(n, p)$ where we draw the weights $w_e \sim U(0, 100)$ uniformly. The experiments shown in Figure 2-b) were run on graphs of size $n = 1000$ using $\Delta_\infty = 0.1$. As shown, *input privatization* does significantly worse as the graph's density increases, whereas our approach resembles the output distribution of PAMST.

7 Conclusion and Open Problems

Our work shows that a simple input perturbation yields a privacy guarantee for the output of any MST algorithm far better than what is implied by the post-processing property of DP. It is natural to wonder if something similar holds for other problems where the best existing in-place private algorithms add noise during the computation.

¹Note that in the Chow-Liu setting, we want to find the maximum spanning tree, which is the same as the mst on the negated weights.

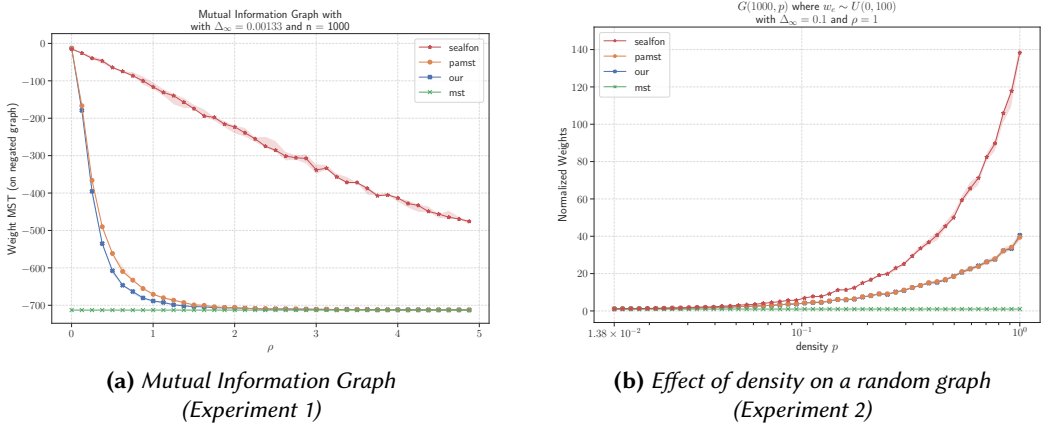


Fig. 2. The results of our experiment. **a)** Shows the instance described in Experiment 1. Note that we have to negate all weights to find the maximum spanning tree on the mutual information graph. **b)** Shows the impact of the graph's density on random graphs with $n = 1000$ vertices for a fixed privacy level $\rho = 1$: The figure shows the ratio between the real mst and the private one, where each edge weight is uniformly drawn from the interval $[0, 100]$. Because the noise scale of Sealfon's input perturbation scales with the number of edges in the graph, we see a larger gap for denser graphs. Each data point shows the median of ten runs.

There remains a small gap of $O(\sqrt{\log(1/\delta)})$ between known upper and lower utility bounds. As discussed in Section 5.1, extending the Laplace noise-based approach for top- k selection by Qiao et al. [34] to the private MST problem remains an open question.

Acknowledgments

Pagh, Wu, Zhang, and Retschmeier carried out this work at Basic Algorithms Research Copenhagen (BARC), which was supported by the VILLUM Foundation grant 54451. *Providentia*, a Data Science Distinguished Investigator grant from the Novo Nordisk Fonden, supported Pagh, Retschmeier, and Wu. Hanwen Zhang is also partially supported by Starting Grant 1054-00032B from the Independent Research Fund Denmark under the Sapere Aude research career programme. We thank Edith Cohen for providing us with historical context for PPSACR and anonymous reviewers of a previous version of this paper for their valuable feedback.

A Algorithm for Maximum Weight Independent Set in a Matroid

A matroid $(U, \mathcal{I})$ is defined by a finite ground set U and a family of *independent sets* $\mathcal{I} \subseteq 2^U$. The family $\mathcal{I}$ satisfies the following three properties:

- $\emptyset \in \mathcal{I}$;
- $\forall S \in \mathcal{I}$ and $T \subseteq S, T \in \mathcal{I}$;
- $\forall S, T \in \mathcal{I}$ and $|S| < |T|$, there exists $t \in T \setminus S$ such that $S \cup \{t\} \in \mathcal{I}$.

As an example, let U be the set of edges in a graph and $\mathcal{I}$ be the family of all subsets of edges that form a forest. It's easy to verify that $(U, \mathcal{I})$ is indeed a matroid, more concretely a *graphic matroid*. Given a weight function $w : U \rightarrow \mathbb{R}^+$, the weight of an independent set S , $w(S)$, is defined as the sum of the weight of its elements. Therefore, MST problem is a special case of finding a maximum weight independent set in this matroid. Kruskal's [21] algorithm was generalized to the problem of finding a maximum weight independent set in a matroid by Edmonds [11]. The algorithm first sorts all the elements in the matroid by weight in decreasing order and then tries to insert elements one by one into the independent set if possible. Since the analysis for Algorithm 1 is based on Kruskal's algorithm, it naturally generalizes to finding a maximum weight independent set in general matroids. For matroids with rank n and m elements and since there is no upper bound for m on n , the expected error we have will be $O(n^{3/2} \log m / \sqrt{\rho})$.

Algorithm 5 Private Maximum-Weight-Independent-Set-in-Matroid Framework

Input: A matroid $(U, \mathcal{I})$ and the weights $\mathbf{W}$, algorithm $\mathcal{A}_{\text{MWIS}(\text{matroid})}$, privacy parameters (ϵ, δ)

- 1: $\rho \leftarrow (\sqrt{\epsilon + \log(1/\delta)} - \sqrt{\log(1/\delta)})^2$; $\epsilon' \leftarrow \sqrt{2 \cdot \rho / (n-1)}$;
 - 2: $\tilde{w}_e \leftarrow w_e + (2/\epsilon') \cdot \ln(\text{Exp}(1))$ for all $e \in U$
 - 3: **return** $\mathcal{A}_{\text{MWIS}(\text{matroid})}(U, \mathcal{I}, \tilde{\mathbf{W}})$
-

B One-Shot-PPSACR

Algorithm 6 One-Shot-PPSACR

- 1: $\tilde{s}(j) \leftarrow \text{Exp}(1) / s(j), \forall j \in U$.
 - 2: $C \leftarrow U, \mathcal{I} \leftarrow$ An empty list ▷ possible candidates and sampled items so far
 - 3: **while** $|\mathcal{I}| \leq k$ and $C \neq \emptyset$ **do**
 - 4: $j \leftarrow \arg \min_{j' \in C} \tilde{s}(j')$.
 - 5: Add j to the back of $\mathcal{I}$
 - 6: $C \leftarrow C \setminus (\{j\} \cup f(\mathcal{I}))$
 - 7: **end while**
 - 8: **return** $\mathcal{I}$
-

C Probabilities

The distribution of the logarithm of an exponential random variable is closely connected to the *Gumbel distribution*.

DEFINITION C.1 (GUMBEL DISTRIBUTION). *Given parameter $b \in \mathbb{R}$, the Gumbel distribution, $\text{Gumbel}(b)$, has probability density function $p(z) = \frac{1}{b} \cdot \exp\left(-\left(\frac{z}{b} + \exp\left(-\frac{z}{b}\right)\right)\right)$, $\forall z \in \mathbb{R}$, and cumulative distribution function $F(z; b) = \exp\left(-e^{-z/b}\right)$, $\forall z \in \mathbb{R}$.*

FACT C.2. *If $Z \sim \text{Exp}(1)$, then $-\ln Z \sim \text{Gumbel}(1)$.*

PROOF. $\Pr[-\ln X \leq z] = \Pr[X \leq \exp(-z)] = \exp(-e^{-z})$ for all $z \in \mathbb{R}$. $\square$

Proof of Fact 3.1. For our proof, we require the following fact, which we will prove later.

FACT C.3. *If $Z_i \sim \ln \text{Exp}(1)$, then for each $\beta \in (0, 1)$,*

$$\Pr\left[Z_i > \ln \ln \frac{1}{\beta}\right] = \Pr\left[Z_i < \ln \ln \frac{1}{1-\beta}\right] = \beta.$$

Since $\ln(1+x) \leq x$, $\forall x > -1$, it holds that $\ln \ln \frac{1}{1-\beta} = \ln(-\ln(1-\beta)) \geq \ln \beta = -\ln \frac{1}{\beta}$. It follows that

$$\Pr\left[Z_i < -\ln \frac{1}{\beta}\right] \leq \Pr\left[Z_i > \ln \ln \frac{1}{\beta}\right] = \beta.$$

Define $X \doteq \max_{i \in [m]} |Z_i|$. By a union bound, we have

$$\Pr\left[|X| \geq \ln \frac{2 \cdot m}{\beta}\right] \leq \beta, \quad \forall \beta \in (0, 1).$$

Define $t \doteq \ln \frac{2 \cdot m}{\beta} \geq \ln(2 \cdot m)$. Then $\beta = (2 \cdot m) \cdot e^{-t}$. Therefore,

$$\begin{aligned} \mathbb{E}[|X|] &= \int_0^{\ln(2 \cdot m)} \Pr[|X| \geq t] dt + \int_{\ln(2 \cdot m)}^{\infty} \Pr[|X| \geq t] dt \\ &\leq \ln(2 \cdot m) + \int_{\ln(2 \cdot m)}^{\infty} (2 \cdot m) \cdot e^{-t} dt \leq \ln(2 \cdot m) + (2 \cdot m) \cdot e^{-\ln(2 \cdot m)} = \ln(2e \cdot m). \end{aligned}$$

PROOF OF FACT C.3. If $Y \sim \text{Exp}(1)$ where $\Pr[Y \leq y] = e^{-y}$ we have for each $\beta \in (0, 1)$,

$$\Pr\left[Y > \ln \frac{1}{\beta}\right] = \exp\left(-\ln \frac{1}{\beta}\right) = \beta, \quad \text{and} \quad \Pr\left[Y < \ln \frac{1}{1-\beta}\right] = 1 - \exp\left(-\ln \frac{1}{1-\beta}\right) = \beta.$$

It follows that

$$\Pr\left[\ln Y > \ln \ln \frac{1}{\beta}\right] = \Pr\left[\ln Y < \ln \ln \frac{1}{1-\beta}\right] = \beta.$$

$\square$

Proof of Fact 3.6.

$$p_Y(y) = p_X(\lambda \cdot y) \cdot \frac{dX}{dY} = \lambda \cdot e^{-\lambda \cdot y}, \quad \forall y \geq 0.$$

$\square$

FACT C.4 (TWO VARIABLES). *If $X \sim \text{Exp}(\lambda_1)$, and $Y \sim \text{Exp}(\lambda_2)$, then*

$$\Pr[X \leq Y] = \int_0^{\infty} \lambda_1 e^{-\lambda_1 x} \cdot \Pr[Y \geq x] dx = \int_0^{\infty} \lambda_1 e^{-\lambda_1 x} e^{-\lambda_2 x} dx = \frac{\lambda_1}{\lambda_1 + \lambda_2}.$$

Proof of Fact 3.7. Let $Y \doteq \min_{j \in [d] \setminus \{i\}} X_j$. Then

$$\Pr[Y \geq y] = \prod_{j \in [d] \setminus \{i\}} \Pr[X_j \geq y] = \exp\left(-y \sum_{j \in [d] \setminus \{i\}} \lambda_j\right). \quad (26)$$

Therefore, $Y \sim \text{Exp}(\sum_{j \in [d] \setminus \{i\}} \lambda_j)$. Applying Fact C.4 gives $\Pr[X_i = \min_{j \in [d]} X_j] = \frac{\lambda_i}{\sum_{j \in [d]} \lambda_j}$. $\square$

Proof of Fact 3.8. $\Pr[X \geq x + y \mid X \geq x] = \frac{\exp(x+y)}{\exp(x)} = \exp(y) = \Pr[X \geq y]$. $\square$

D Utility Guarantee of Private Kruskal

We prove the utility of private Kruskal's algorithm (Algorithm 2) directly. Suppose we use Kruskal's algorithm to find the MST on the noisy and original graphs in parallel. When we try to add the k -th edge in noisy graph, we are actually finding the noisy minimum-weighted edge among all the cycle-free candidates. Also, there must be a candidate from the first k added edges when we run Kruskal's algorithm in the original graph, as all forests form a matroid. Therefore, the real minimum weight is not greater than the payment for the k -th step on the original graph, so the extra cost induced in this step only comes from selecting the noisy-min. Accumulate the error over all steps, we show the desired utility guarantee of the algorithm.

PROOF. For each $i \in \{1, \dots, n-1\}$, let e_i be the edge added to T at the i^{th} iteration of Algorithm 2. Let $T_i \doteq \bigcup_{j \in [i]} \{e_j\}$ be the partial spanning forest after iteration i and for convenience, let $T_0 = \emptyset$. Further, initially set $F_0 = E$, and denote with F_i the set F immediately after the i^{th} iteration containing all edges that still could be added to T without creating a cycle. Assume that also run a non-private Kruskal's algorithm on the same graph with the true edge weights in parallel, to obtain the *real MST* $T_i^* = \bigcup_{j \in [i]} \{e_j^*\}$, where e_j^* is the edge added at iteration j .

To prove the utility guarantee, we want to show that for all steps, with probability $1 - \beta$ and for some universal constant $c \in \mathbb{R}^+$,

$$w_{e_i} \leq w_{e_i^*} + \frac{c}{\varepsilon'} \cdot \ln \frac{n}{\beta}. \quad (27)$$

Note that the utility guarantee of the exponential mechanism gives us with probability $1 - \beta$

$$w_{e_i} \leq \min_{e \in F_{i-1}} w_e + \frac{c}{\varepsilon'} \cdot \ln \frac{n}{\beta} \quad (28)$$

because e_i is exactly the noisy minimum of the set F_{i-1} chosen by the algorithm. Therefore, it suffices to show that

$$\min_{e \in F_{i-1}} w_e \leq w_{e_i^*}. \quad (29)$$

Note that $w_{e_{i-1}^*} \leq w_{e_i^*}$ by the way Kruskal adds new edges to T^* . Using this, we claim that F_{i-1} contains at least one edge which belongs to T_i^* and can be added to T_{i-1} without forming a cycle.

First, note that edges in T_{i-1} induce a forest. Let $\mathcal{T}$ be an arbitrary tree in this forest, and assume that it has t edges. Then, the number of edges $e = (u, v) \in T_i^*$, such that both $u, v \in \mathcal{T}$ is at most t without inducing a cycle.

There are $i-1$ edges in T_{i-1} , but i edges in T_i^* . Therefore, there must be *at least one* edge from T^* , which connects two trees in the forest induced by T_{i-1} . It follows that this edge belongs to F_{i-1} , proving Equation (29). Summing over all possible edges gives

$$\sum_{i \in [n-1]} w_{e_i} \leq \sum_{i \in [n-1]} w_{e_i^*} + \frac{c(n-1)}{\varepsilon'} \ln \frac{n}{\beta}.$$

By a standard integration technique, this also implies an expected error of $O(\frac{n}{\varepsilon'} \ln n)$. $\square$

E Efficiently Implementing $\mathcal{A}_{\text{priv-kruskal}}$

This section describes how $\mathcal{A}_{\text{priv-kruskal}}$ can efficiently be implemented. Although this result is, of course, overshadowed by our main result, we believe it is interesting enough to be stated here.

The algorithm can be implemented in $O(n + m \log n)$ time. In the initialization phase, we construct a complete binary tree where every level, except possibly the last, is fully filled, and all nodes in the last level are as far left as possible—with m leaf nodes. Each leaf node represents an edge and is assigned a weight $e^{-\varepsilon' w_e}$. Each internal node of the binary tree is assigned a weight equal to the sum of the weights of the leaf nodes in the subtree rooted at that node.

This binary tree is used to manage the nodes in F efficiently: it supports both the sampling operation (Algorithm 2, Line 4) and the update operation (Algorithm 2, Line 6) in $O(\log n)$ amortized time per edge.

Sampling. We apply a top-down approach starting from the tree's root to sample an edge (i.e., a leaf node). At each step, we move to the left child with probability equal to the ratio of the left child's weight to the current node's weight, and move to the right child otherwise. The sampling procedure terminates when the current node is a leaf. It is straightforward to verify via an induction on the number of levels that this method samples a leaf node with probability proportional to $e^{-\varepsilon' w_e}$ and that the sampling process completes in $O(\log n)$ time.

Update. We introduce two auxiliary data structures for this step. First, we maintain a union-find data structure [39], which allows us to determine the connected component (induced by the edges in T) to which a vertex belongs in $O(\alpha(n))$ amortized time, where $\alpha(n)$ is the inverse Ackermann function. Additionally, for each connected component, we maintain a linked list to track the vertices it contains. When a new edge $e = (u, v)$ is added to the set T , let C_u and C_v denote the connected components containing u and v , respectively. To merge C_u and C_v , we update both the union-find structure and the linked list, which can be performed in $O(\alpha(n) + 1)$ time. Next, we need to remove from F all edges connecting C_u and C_v . To do so efficiently, we choose the smaller of C_u and C_v (in term of the number of vertices)—assume, without loss of generality, it is C_u —and check all edges incident to it. For each edge, if its other endpoint belongs to C_v , we remove it from F .

We now analyze the number of times an edge is checked and the cost of its removal from F . For the former, define the rank of a vertex as the number of vertices in the connected component to which it belongs. Each time an edge is checked, the rank of one of its endpoints at least doubles. Since a vertex can have a rank of at most n , an edge can be checked at most $O(\log n)$ times.

For the latter, removing an edge from F is handled by removing the corresponding leaf node from the binary tree and then updating the weights of the nodes along the leaf-to-root path. This operation can be completed in $O(\log n)$ time.

F Proof of Lemma 4.5

Our proof relies on a standard result from the Erdős–Rényi model $\mathcal{G}(n, p)$, where $\mathcal{G}$ is a random graph with n vertices such that each edge is independently included with probability p . The following lemma is stated with explicit constants, as our subsequent proof relies on them. Despite extensive searching, we could not find a version of the lemma with explicit constants. For completeness, we provide a proof of this lemma with explicit constants in the complete version [30].

LEMMA F.1 ([12] CONNECTIVITY). *For any real $\varepsilon > 0$, if $p > \frac{(1+\varepsilon)\ln n}{n}$, then a random graph in the Erdős–Rényi model $\mathcal{G}(n, p)$ with n vertices is almost surely connected. In particular, if $p \geq \frac{2 \cdot \ln n}{n-1}$ and $n \geq 1700$, the probability that the random graph is not connected is at most $\frac{6}{n}$.*

Recall that the random graph G considered in Lemma 4.5 is generated as follows: G consists of n vertices and $m = \frac{n(n-1)}{2}$ edges. For each edge e , we first sample $P_e \sim \text{Beta}(\beta, \beta)$ and then sample $w_e \sim \mathcal{B}(s, P_e)$.

Let H be the subgraph that includes all vertices in G and all edges e such that $w_e \geq 3/4 \cdot s$. Since the edge weights are independent of each other, the graph H is an instance of Erdős–Rényi model with $p = \Pr[w_e \geq 3/4 \cdot s]$. Note that G has a spanning tree where all edges e have $w_e \geq 3/4 \cdot s$ if and only if H is connected. Given Lemma F.1, to prove Lemma 4.5, it just remain to show that for each edge e ,

$$\Pr\left[w_e \geq \frac{3}{4} \cdot s\right] > \frac{2 \ln n}{n-1}. \quad (30)$$

FACT F.2 (STEINKE AND ULLMAN [38]). *Let $\beta \geq 1$, and X be a random variable following beta distribution $\text{Beta}(\beta, \beta)$. Then for all $p \in [0, 1/2]$, it holds that*

$$\Pr[X \leq p] = \Pr[X \geq 1-p] \geq (4 \cdot p \cdot (1-p))^{\beta-1} \cdot \frac{p}{\beta} \quad (31)$$

Firstly, according to Fact F.2, we have $\Pr[P_e \geq 0.9] \geq \frac{0.36^{\beta-1}}{10\beta}$ and since $\mathbb{E}[w_e] = s \cdot P_e$, by Hoeffding's inequality,

$$\Pr[w_e > s \cdot P_e - 0.15s \mid P_e \geq 0.9] < \exp(-0.045s). \quad (32)$$

Therefore,

$$\Pr\left[w_e \geq \frac{3}{4}s\right] \geq \Pr[P_e \geq 0.9] \cdot \Pr[w_e \leq s \cdot P_e - 0.15s \mid P_e \geq 0.9] \geq \frac{0.36^{\beta-1}}{10\beta} \cdot (1 - \exp(-0.045s)).$$

For $s \geq 10$ and $\beta = \frac{1}{2} \ln n$ and $n \geq 2 \times 10^7$, we have

$$\Pr\left[w_e \geq \frac{3}{4}\right] \geq \frac{0.36^{\frac{1}{2} \ln n}}{1.8 \ln n} \cdot (1 - \exp(-0.45)) > \frac{0.36}{1.8n^{0.52} \ln n} > \frac{2 \ln n}{n-1}. \quad (33)$$

□

G Proof of Lemma 4.7

PROOF. Since w_e is a sum of s independent random variables between $[0, 1]$, and has expectation $s \cdot P_e$, by Hoeffding's inequality, it holds that $\Pr[|w_e - s \cdot P_e| \geq t] \leq 2 \cdot \exp\left(-\frac{2t^2}{s}\right)$ for all $t \geq 0$.

By union bound, it holds that for all $t \geq 0$, $s \geq 1$ and $n \geq 100$:

$$\begin{aligned} \Pr\left[\max_{e \in E} |w_e - s \cdot P_e| \geq t + \sqrt{s \cdot \ln n}\right] &\leq 2 \cdot \frac{n(n-1)}{2} \cdot \exp\left(-2 \cdot \left(t + \sqrt{s \cdot \ln n}\right)^2 / s\right) \\ &\leq n^2 \cdot \exp\left(-2 \cdot t^2 / s\right) \cdot \exp\left(-2 \cdot s \cdot (\ln n) / s\right) = \exp\left(-2 \cdot t^2 / s\right). \end{aligned}$$

Therefore,

$$\begin{aligned} \mathbb{E}\left[\max_{e \in E} |w_e - s P_e|\right] &= \int_0^{\sqrt{s \ln n}} \Pr\left[\max_{e \in E} |w_e - s \cdot P_e| \geq t\right] dt + \int_{\sqrt{s \ln n}}^\infty \Pr\left[\max_{e \in E} |w_e - s \cdot P_e| \geq t\right] dt \\ &\leq \sqrt{s \cdot \ln n} + \int_0^\infty \Pr\left[\max_{e \in E} |w_e - s \cdot P_e| \geq t + \sqrt{s \cdot \ln n}\right] dt \\ &\leq \sqrt{s \cdot \ln n} + \int_0^\infty \exp\left(-2 \cdot t^2 / s\right) dt = \sqrt{s \cdot \ln n} + \frac{1}{2} \cdot \int_{-\infty}^\infty \exp\left(-\frac{t^2}{2 \cdot (s/4)}\right) dt \\ &= \sqrt{s \cdot \ln n} + \frac{\sqrt{2 \cdot \pi \cdot (s/4)}}{2} < \sqrt{3s \ln n}. \end{aligned}$$

□

H Proofs for the Experiments

Assume $0 < p < \frac{1}{2}$. The mutual information $I(X; Y)$ between two discrete random variables X and Y quantifies the amount of information that X contains about Y (or vice versa).

DEFINITION H.1 (MUTUAL INFORMATION [1]).

$$I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \Pr[X = x, Y = y] \log_2 \left(\frac{\Pr[X = x, Y = y]}{\Pr[X = x] \Pr[Y = y]} \right). \quad (34)$$

We now prove the validity of the weight function given in Section 6.

DEFINITION H.2. We define the random variables $X_1, \dots, X_n \in \{0, 1\}$ recursively by setting

$$X_1 \sim \text{Ber}\left(\frac{1}{2}\right) \text{ and } X_i = \begin{cases} 1 - X_{i-1} & \text{with probability } p \\ X & \text{else} \end{cases}$$

By definition, $\Pr[X_i = 0] = \Pr[X_i = 1] = \frac{1}{2}$. Note that mutual information is minimized for $p \approx \frac{1}{2}$ and note that for all X_i : $\Pr[X_i = 1] = \Pr[X_i = 0] = \frac{1}{2}$. We need another short lemma for the parity of a binomial random variable:

LEMMA H.3 (PARITY OF BINOMIALS.). Denote $Z \sim \mathcal{B}(n, p)$, then

$$\Pr[Z \text{ is even}] = 1 - \Pr[Z \text{ is odd}] = \frac{1}{2} + \frac{1}{2}(1 - 2p)^n \quad (35)$$

PROOF. By direct computation and application of the binomial theorem in the last step:

$$\begin{aligned} \Pr_{Z \sim \mathcal{B}(n, p)}[Z \text{ is even}] &= \sum_{k=0}^n \frac{(-1)^k + 1}{2} \Pr[Z = k] = \frac{1}{2} \left(\sum_{k=0}^n ((-1)^k \Pr[Z = k]) + \sum_{k=0}^n \Pr[Z = k] \right) \\ &= \frac{1}{2} \left(\sum_{k=0}^n ((-1)^k \Pr[Z = k]) + 1 \right) = \frac{1}{2} \left(\sum_{k=0}^n ((-1)^k \binom{n}{k} p^k (1-p)^{n-k}) + 1 \right) \\ &= \frac{1}{2} \left(\sum_{k=0}^n \left(\binom{n}{k} (-p)^k (1-p)^{n-k} \right) + 1 \right) = \frac{1}{2} (1 - 2p)^n + \frac{1}{2} \end{aligned}$$

□

CLAIM H.4. Assume $0 < p < \frac{1}{2}$, the mutual information score between X_i and X_j that are $k = |i - j|$ steps apart can directly be computed by

$$I(X_i; X_j) = \left(\frac{1}{2} + \frac{1}{2}(1 - 2p)^k \right) \log_2 \left(1 + (1 - 2p)^k \right) + \left(\frac{1}{2} - \frac{1}{2}(1 - 2p)^k \right) \log_2 \left(1 - (1 - 2p)^k \right) \quad (36)$$

PROOF. We shortly write $p_{00}(k) = \Pr[X_i = 0, X_j = 0]$ and $p_0 = \Pr[X_i = 0]$ (resp $p_{11}(k)$, $p_{10}(k)$ and $p_{01}(k)$, p_1). Furthermore, denote $p_{\text{even}}(k) = \Pr_{Z \sim \mathcal{B}(k, p)}[Z \text{ is even}]$ and $p_{\text{odd}}(k)$ as proven above.

Note that we can compute these probabilities:

$$\begin{aligned} p_{00}(k) &= \Pr[X_i = 0 | X_j = 0] \cdot \Pr[X_j = 0] = p_{\text{even}}(k) \cdot \frac{1}{2} = \frac{1}{4} + \frac{1}{4} \cdot (1 - 2p)^k \\ p_{10}(k) &= \Pr[X_i = 1 | X_j = 0] \cdot \Pr[X_j = 0] = p_{\text{odd}}(k) \cdot \frac{1}{2} = \frac{1}{4} - \frac{1}{4} (1 - 2p)^k \end{aligned}$$

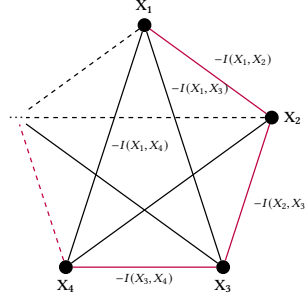


Fig. 3. An extract of the complete graph encoding the mutual information between the random variables $X_1, \dots, X_n$ described in Definition H.2 and used in Section 6. The weights encode the negated mutual information corresponding to the described process. The mst is formed by the vertices on the path $P(X_1, X_2, \dots)$. In our experiment with $n = 1000$ vertices and the flip probability $p = 0.05$, we have $-I(X_1, X_2) = I(X_2, X_3) = \dots \approx -0.7136$, $-I(X_1, X_3) = I(X_2, X_4) = \dots \approx -0.5471$ and $-I(X_1, X_4) \dots \approx -0.4277$.

We trivially get $p_{11}(k) = p_{00}(k)$ and $p_{01}(k) = p_{10}(k)$ by symmetry. Then, we can compute the mutual information directly.

$$\begin{aligned}
 I(X_i; X_j) &= p_{00}(k) \log_2 \frac{p_{00}(k)}{p_0 p_0} + p_{01}(k) \log_2 \frac{p_{01}(k)}{p_0 p_1} + p_{10}(k) \log_2 \frac{p_{10}(k)}{p_1 p_0} + p_{11}(k) \log_2 \frac{p_{11}(k)}{p_1 p_1} \\
 &= p_{00}(k) \log_2 (4p_{00}(k)) + p_{01}(k) \log_2 (4p_{01}(k)) + p_{10}(k) \log_2 (4p_{10}(k)) + p_{11}(k) \log_2 (4p_{11}(k)) \\
 &= 2p_{00}(k) \log_2 (4p_{00}(k)) + 2p_{01}(k) \log_2 (4p_{01}(k)) \\
 &= \left(\frac{1}{2} + \frac{1}{2} (1 - 2p)^k \right) \log_2 \left(1 + (1 - 2p)^k \right) + \left(\frac{1}{2} - \frac{1}{2} (1 - 2p)^k \right) \log_2 \left(1 - (1 - 2p)^k \right)
 \end{aligned}$$

□

I Table of Symbols

Symbol	Description
PPSACR	Probability Proportional to Sizes with Adaptive Candidate Removal
MST	Minimum Spanning Tree
$G = (V, E, \mathbf{W})$	Graph with weights $\mathbf{W} \in \mathbb{R}^{ E }$
n, m	Number of vertices and edges
$T \subseteq E$	A tree in G
$\mathcal{T}(G)$	Set of all spanning trees
$\Delta_1, \Delta_2, \Delta_\infty$	Sensitivity parameters
$W \sim W'$	Neighboring weights
$\rho, \varepsilon, \delta$	Privacy parameters
$I(X; Y)$	Mutual Information between X and Y
$\mathcal{B}(k, p)$	Binomial Distribution
$\text{Exp}(\lambda)$	Exponential Distribution
Gumbel(b)	Gumbel Distribution
$\mathcal{G}(n, p)$	Erdos-Renyi graphs

References

- [1] R.B. Ash. 1990. *Information Theory*. Dover Publications. <https://books.google.dk/books?id=nJ3UmGvdUCoC>
- [2] MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab S. Mirrokni. 2017. Affinity Clustering: Hierarchical Clustering at Scale. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6864–6874. <https://proceedings.neurips.cc/paper/2017/hash/2e1b24a664f5e9c18f407b2f9c73e821-Abstract.html>
- [3] Felipe T Brito, André LC Mendonça, and Javam C Machado. 2024. A Differentially Private Guide for Graph Analytics. In *EDBT*. 850–853.
- [4] Mark Bun and Thomas Steinke. 2016. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. In *Theory of Cryptography (Lecture Notes in Computer Science)*, Martin Hirt and Adam Smith (Eds.). Springer, Berlin, Heidelberg, 635–658. https://doi.org/10.1007/978-3-662-53641-4_24
- [5] Bernard Chazelle. 2000. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *J. ACM* 47, 6 (Nov. 2000), 1028–1047. <https://doi.org/10.1145/355541.355562>
- [6] C. K. Chow and Chao-Ming Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory* 14 (1968), 462–467. <https://api.semanticscholar.org/CorpusID:27127853>
- [7] Edith Cohen. 1997. Size-Estimation Framework with Applications to Transitive Closure and Reachability. *J. Comput. Syst. Sci.* 55, 3 (1997), 441–453. <https://doi.org/10.1006/JCSS.1997.1534>
- [8] David Durfee and Ryan M Rogers. 2019. Practical Differentially Private Top-k Selection with Pay-what-you-get Composition. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/b139e104214a08ae3f2ebcce149cdf6e-Paper.pdf
- [9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Third Conference on Theory of Cryptography* (New York, NY) (TCC'06). Springer-Verlag, Berlin, Heidelberg, 265–284. https://doi.org/10.1007/11681878_14
- [10] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407. <https://doi.org/10.1561/04000000042>
- [11] Jack Edmonds. 1971. Matroids and the greedy algorithm. *Mathematical programming* 1 (1971), 127–136.
- [12] P. Erdős and A. Rényi. 1959. On Random Graphs I. *Publicationes Mathematicae Debrecen* 6 (1959), 290.
- [13] Michael L. Fredman and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* 34, 3 (July 1987), 596–615. <https://doi.org/10.1145/28869.28874>
- [14] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.
- [15] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. 2009. Accurate Estimation of the Degree Distribution of Private Networks. In *2009 Ninth IEEE International Conference on Data Mining*. 169–178. <https://doi.org/10.1109/ICDM.2009.11>
- [16] Richard Hladík and Jakub Tětek. 2025. Near-Universally-Optimal Differentially Private Minimum Spanning Trees. *Symposium on the Foundations of Responsible Computing (FORC)* (2025). arXiv:2404.15035 [cs.CR]
- [17] Vojtěch Jarník. 1930. O jistém problému minimálním. (Z dopisu panu O. Borůvkovi) [On a certain problem of minimization]. *Práce moravské přírodovědecké společnosti* 6 (1930), 57–63. <https://dml.cz/handle/10338.dmlcz/500726?show=full>
- [18] Rajesh Jayaram, Vahab Mirrokni, Shyam Narayanan, and Peilin Zhong. 2024. Massively Parallel Algorithms for High-Dimensional Euclidean Minimum Spanning Tree. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, David P. Woodruff (Ed.). SIAM, 3960–3996. <https://doi.org/10.1137/1.9781611977912.139>
- [19] David R. Karger, Philip N. Klein, and Robert E. Tarjan. 1995. A randomized linear-time algorithm to find minimum spanning trees. *J. ACM* 42, 2 (March 1995), 321–328. <https://doi.org/10.1145/201019.201022>
- [20] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography* (Tokyo, Japan) (TCC'13). Springer-Verlag, Berlin, Heidelberg, 457–476. https://doi.org/10.1007/978-3-642-36594-2_26
- [21] Joseph B. Kruskal. 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proc. Amer. Math. Soc.* 7, 1 (1956), 48–50. <http://www.jstor.org/stable/2033241>
- [22] Chih Lai, Taras Rafa, and Dwight E. Nelson. 2009. Approximate minimum spanning tree clustering in high-dimensional space. *Intell. Data Anal.* 13, 4 (2009), 575–597. <https://doi.org/10.3233/IDA-2009-0382>
- [23] Yang Li, Michael Purcell, Thierry Rakotoarivelo, David Smith, Thilina Ranbaduge, and Kee Siong Ng. 2023. Private Graph Data Release: A Survey. *ACM Comput. Surv.* 55, 11, Article 226 (Feb. 2023), 39 pages. <https://doi.org/10.1145/3569085>

- [24] Ryan McKenna, Gerome Miklau, and Daniel Sheldon. 2021. Winning the NIST Contest: A scalable and general approach to differentially private synthetic data. *Journal of Privacy and Confidentiality* 11, 3 (Dec. 2021). <https://doi.org/10.29012/jpc.778>
- [25] Ryan McKenna and Daniel R Sheldon. 2020. Permute-and-Flip: A new mechanism for differentially private selection. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 193–203. https://proceedings.neurips.cc/paper_files/paper/2020/file/01e00f2f4bfcbb7505cb641066f2859b-Paper.pdf
- [26] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 94–103.
- [27] Tamara T. Mueller, Dmitrii Usynin, Johannes C. Paetzold, Daniel Rueckert, and Georgios Kaissis. 2022. SoK: Differential Privacy on Graph-Structured Data. arXiv:2203.09205 [cs.CR]
- [28] Esbjörn Ohlsson. 1990. *Sequential poisson sampling from a business register and its application to the Swedish consumer price index*. Statistiska centralbyrån.
- [29] Rasmus Pagh and Lukas Retschmeier. 2024. Faster Private Minimum Spanning Trees. arXiv:2408.06997 [cs.DS] <https://arxiv.org/abs/2408.06997>
- [30] Rasmus Pagh, Lukas Retschmeier, Hao Wu, and Hanwen Zhang. 2024. Optimal Bounds for Private Minimum Spanning Trees via Input Perturbation. CoRR abs/2412.10130 (2024). <https://doi.org/10.48550/ARXIV.2412.10130> arXiv:2412.10130
- [31] Rafael Pinot. 2018. Minimum spanning tree release under differential privacy constraints. *arXiv e-prints (Master Thesis)* (2018). arXiv:1801.06423 [cs.CR]
- [32] Rafael Pinot, Anne Morvan, Florian Yger, Cedric Gouy-Pailler, and Jamal Atif. 2018. Graph-based Clustering under Differential Privacy. In *Conference on Uncertainty in Artificial Intelligence (UAI 2018)*. Conference on Uncertainty in Artificial Intelligence (UAI 2018), Monterey, California, United States, 329–338. <https://hal.science/hal-02170699>
- [33] Robert Clay Prim. 1957. Shortest connection networks and some generalizations. *The Bell System Technical Journal* 36 (1957), 1389–1401. Issue 6. <https://doi.org/10.1002/j.1538-7305.1957.tb01515.x>
- [34] Gang Qiao, Weijie J. Su, and Li Zhang. 2021. Oneshot Differentially Private Top-k Selection. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8672–8681.
- [35] Sheldon Ross. 2018. *A First Course in Probability* (10 ed.). Pearson, Upper Saddle River, NJ.
- [36] Bengt Rosén. 1997. Asymptotic theory for order sampling. *Journal of Statistical Planning and Inference* 62, 2 (1997), 135–158. [https://doi.org/10.1016/S0378-3758\(96\)00185-1](https://doi.org/10.1016/S0378-3758(96)00185-1)
- [37] Adam Sealfon. 2016. Shortest Paths and Distances with Differential Privacy. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (, San Francisco, California, USA, (PODS '16)*. Association for Computing Machinery, New York, NY, USA, 29–41. <https://doi.org/10.1145/2902251.2902291>
- [38] Thomas Steinke and Jonathan R. Ullman. 2017. Tight Lower Bounds for Differentially Private Selection. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15–17, 2017*, Chris Umans (Ed.). IEEE Computer Society, Berkeley, USA, 552–563.
- [39] Robert Endre Tarjan. 1975. Efficiency of a Good But Not Linear Set Union Algorithm. *J. ACM* 22, 2 (April 1975), 215–225. <https://doi.org/10.1145/321879.321884>
- [40] Salil P. Vadhan. 2017. The Complexity of Differential Privacy. In *Tutorials on the Foundations of Cryptography*, Yehuda Lindell (Ed.). Springer International Publishing, 347–450. https://doi.org/10.1007/978-3-319-57048-8_7
- [41] Tim Van Erven and Peter Harremos. 2014. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory* 60, 7 (2014), 3797–3820.
- [42] John I. Yellott. 1977. The relationship between Luce’s Choice Axiom, Thurstone’s Theory of Comparative Judgment, and the double exponential distribution. *Journal of Mathematical Psychology* 15, 2 (1977), 109–144. [https://doi.org/10.1016/0022-2496\(77\)90026-8](https://doi.org/10.1016/0022-2496(77)90026-8)

Received December 2024; revised February 2025; accepted March 2025