2021 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-2398-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDM51629.2021.00014

Differentially Private String Sanitization for Frequency-Based Mining Tasks

Huiping Chen[1], Changyu Dong[2], Liyue Fan[3], Grigorios Loukides[1], Solon P. Pissis[4,5,6], and Leen Stougie[4,5,6]
[1]King's College London, United Kingdom
[2]Newcastle University, United Kingdom
[3]University of North Carolina at Charlotte, USA
[4]ERABLE Team, France
[5]CWI, The Netherlands
[6]Vrije Universiteit, The Netherlands
[1]{huiping.chen, grigorios.loukides}@kcl.ac.uk, [2]changyu.dong@newcastle.ac.uk, [3]liyue.fan@uncc.edu, and
[5]{solon.pissis,leen.stougie}@cwi.nl

***Abstract*—Strings are used to model genomic, natural language, and web activity data, and are thus often shared broadly. However, string data sharing has raised privacy concerns stemming from the fact that knowledge of length-k substrings of a string and their frequencies (multiplicities) may be sufficient to uniquely reconstruct the string; and from that the inference of such substrings may leak confidential information. We thus introduce the problem of protecting length-k substrings of *a single string* S by applying Differential Privacy (DP) while maximizing data utility for frequency-based mining tasks. Our theoretical and empirical evidence suggests that classic DP mechanisms are not suitable to address the problem. In response, we employ the order-k de Bruijn graph G of S and propose a sampling-based mechanism for enforcing DP on G. We consider the task of enforcing DP on G using our mechanism while preserving the normalized edge multiplicities in G. We define an optimization problem on integer edge weights that is central to this task and develop an algorithm based on dynamic programming to solve it exactly. We also consider two variants of this problem with real edge weights. By relaxing the constraint of integer edge weights, we are able to develop linear-time exact algorithms for these variants, which we use as stepping stones towards effective heuristics. An extensive experimental evaluation using real-world large-scale strings (in the order of billions of letters) shows that our heuristics are efficient and produce near-optimal solutions which preserve data utility for frequency-based mining tasks.**

***Index Terms*—differential privacy, data sanitization, string algorithms, frequent pattern mining**

I. Introduction

Strings (i.e., sequences of letters over some alphabet) are typically used to model genetic material of organisms, natural language, as well as web activity or movement of individuals. Consequently, strings are featured in a gamut of applications. Examples of such applications are genomic pattern discovery [27], text summarization [19], as well as news recommendation [20] or route planning [11]. To fuel these applications, strings collected from individuals are often shared broadly. For instance, individuals' genomic [14], web [22], or movement [13] data are often disseminated in the context of outsourcing mining tasks. However, disseminating string data may lead to privacy concerns [12], [9], [37], [38], [8].

Our Privacy and Utility Setting. This work formalizes the problem of protecting length-k substrings of *a single string* S over an alphabet Σ, by applying Differential Privacy (DP) [15], while maximizing data utility for frequency-based mining tasks on S. We assume that the *set* of length-$(k-1)$ substrings of S is public and aim to protect the presence or the absence of length-k substrings in S. That is, we want to prevent the inference of the exact frequency (multiplicity) of any length-k substring in S. This prevents the unique reconstruction of S. In particular, knowledge of the set of length-k substrings and their multiplicities in S for a sufficiently large k is a sufficient condition for *uniquely* reconstructing S [18]. Intuitively, this is because there exists a value of k for which there is a single way to combine these length-k substrings to reconstruct S. In fact, this value of k is expected to be small: it is in $O(\log_{|\Sigma|}|S|)$, where $|S|$ is the length of S and $|\Sigma|$ is the alphabet size. Furthermore, this unique reconstruction of S can be performed in linear time in $|S|$ [18]. We also draw our motivation from application domains, in which length-k substrings represent confidential information. In natural language datasets, the dictionary of all 1-grams (words) in the language is often known and, hence, words that may exist in a private document may also be known. However, it is more difficult to know which 2-grams (2-word phrases) occur in a private document, and inferring such phrases may leak confidential information: e.g., short phrases may leak a patient's sensitive information in a medical text [30]. Similarly, in genome datasets, it is known that the DNA sequence of every *Homo sapiens* individual contains every possible length-10 string over the DNA alphabet as a substring but not every possible length-11 string [7], while specific length-k substrings are markers for diseases [35]. In addition to enforcing DP on the multiset of length-k substrings of S, we seek to preserve the normalized multiplicities of length-k substrings (i.e., the relative frequencies of these substrings in S). This is to preserve data utility for frequency-based mining tasks. It should be clear that preserving the relative frequencies helps the accuracy of frequent pattern mining [27], since it avoids changes to the set of frequent length-k substrings (i.e., the set

978-1-6654-2398-4/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDM51629.2021.00014

of length-k substrings with relative frequency at least equal to a given threshold). Preserving the relative frequencies also helps the accuracy of frequency-based clustering [25], since the distances between strings in the collection of strings to be clustered are based on the relative frequency of length-k substrings of the strings [25].

We remark that our setting is fundamentally different from settings seeking to protect the presence of an individual string in *a string collection* by enforcing DP (e.g., [12], [9]); or to prevent the mining of confidential patterns from a string by enforcing privacy principles other than DP (e.g., [16], [17]).

Representing a Multiset of Length-k Substrings. An order-k de Bruijn graph (dBG) over string S, denoted by $G = (V, E)$, has a node $v \in V$ for every distinct length-$(k-1)$ substring of S and an edge $(u, v) \in E$ connecting nodes u and v if their corresponding substrings start at successive positions in S. An example of the order-3 dBG over $S = \texttt{abaabbabba}$ is in Fig. 1. The set of length-2 substrings (nodes) is $V = \{\texttt{aa}, \texttt{ab}, \texttt{ba}, \texttt{bb}\}$. The multiset of length-3 substrings (edges) is $E = \{\texttt{aab}, \texttt{aba}, \texttt{abb}, \texttt{abb}, \texttt{baa}, \texttt{bab}, \texttt{bba}, \texttt{bba}\}$. Substrings ab and bb occur successively in S, so there is an edge e_1 that spells abb. Edge multiplicities are in red; e.g., abb (e_1) occurs twice in S. It should be clear that the size of G is $|E| = |S|-k+1$, where $|E|$ is counted with multiplicities and $|S|$ is the length of S. A dBG thus represents a multiset of length-k strings specified by edge multiplicities.

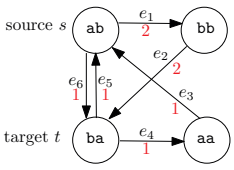


Fig. 1: Order-3 de Bruijn graph $G(V, E)$ for $S = \texttt{abaabbabba}$.

The seminal work of Pevzner et al. [32] proposed to use the dBG over a collection of short DNA sequences, read by sequencing technologies, to assemble these sequences into a string S corresponding to the entire genome. Namely, S corresponds to an Eulerian path in this dBG (i.e., a trail that visits every edge once). In Fig. 1, the Eulerian path $e_6e_5e_1e_2e_4e_3e_1e_2$ from node s to node t is the trail spelling S. Here we borrow the idea of the dBG representation *to perform the reverse*: we disassemble S into its length-k substrings, use an order-k dBG $G(V, E)$ to represent the residual multiset E, and enforce DP on E.

We release an (ϵ, δ)-differentially private [15] version $\widehat{E}$ of E. Since the set of length-$(k-1)$ substrings of S is public (i.e., set V of G is public), we can alternatively release $\widehat{G}(V, \widehat{E})$. We assume Σ is an ordered set and simply represent E as a sequence $M = (m_1, \ldots, m_n)$ of n multiplicities, where $n \leq |E| = |S| - k + 1$ is the size of the set of length-k substrings of S and m_i is the frequency of the element in E whose *lexicographic rank* in E is i. In Fig. 1, we have $M = (1, 1, 2, 1, 1, 2)$: $m_1 = 1$ tells us that aab, whose lexicographic rank in S is 1, occurs 1 time in S; $m_3 = 2$ tells us that abb, whose lexicographic rank in S is 3, occurs 2 times in S.

Our Contributions.

1. We investigate the suitability of applying classic DP mechanisms in our setting. In particular, we provide theoretical and empirical evidence demonstrating that the Laplace [15], Gaussian [15], and randomized response [26] mechanisms are not suitable for our setting. This is because these mechanisms must consider all edges that do or could exist in G (referred to as *feasible edges*), which is often prohibitively expensive, and because they must add noise to a very large number of edges that could but do not exist in G (referred to as *fake feasible edges*), which incurs excessive utility loss. See Section IV.

2. In response, we propose a mechanism, called the Multi-Edge Sampling (MES) mechanism, which enforces (ϵ, δ)-DP and is suitable for our setting. MES operates directly on edges that exist in G. These edges are often significantly fewer than all feasible edges. Thus, MES avoids the limitations of the classic DP mechanisms. We formalize our main computational task as the following optimization problem, which is referred to as the *Private and Similar Multiplicities* (PSM) problem:

Given a sequence M of n integers and parameters ϵ and δ, construct a sequence X of n integers such that the normalized multiplicities in X are *as similar as possible* to those of M, with respect to the L_1 distance $\sum_{i=1}^{n} |m_i / \sum_{j=1}^{n} m_j - x_i / \sum_{j=1}^{n} x_j|$, subject to the *set of constraints* specified by the MES mechanism. When the L_1 distance is minimized, the normalized multiplicities in M, and hence the relative frequencies of length-k substrings in S, are preserved. When the constraints are satisfied, X is used to obtain an (ϵ, δ)-DP $\widehat{E}$. The indistinguishability is guaranteed for any two sequences M and M' such that $|M - M'| = 1$. See Section V.

3. We develop EXACT-DP, an exact algorithm for PSM. The algorithm invokes iteratively a dynamic programming procedure that solves PSM for a fixed output graph size and returns the best solution found over all iterations. EXACT-DP solves PSM in $O(|S|^4)$ time using $O(|S|^2)$ space (see Section VI). Thus, it is impractical for long strings. In response, we consider two relaxed variants of PSM, both for graphs with *real* edge weights: a variant with a fixed output graph size; and another one with arbitrary output graph size. By relaxing the constraint of *integer* edge weights in PSM, we develop efficient, exact algorithms for these variants, which are also used as stepping stones towards effective heuristics. Notably, we show that both algorithms for these PSM variants take $O(|S|)$ time using $O(|S|)$ space. See Section VII.

4. We develop three $O(|S|)$-time heuristics for PSM based on theoretical insight: (I) The MSH heuristic constructs a feasible solution to PSM to obtain a graph of *maximum size*. It works remarkably well for input graphs with large multiplicities and is the most efficient. (II) The FSH heuristic solves the PSM variant with real edge weights, obtaining a graph of *fixed size*, and then performs a linear-time rounding to construct a feasible integer solution to PSM. FSH is slightly less efficient in practice but much better in terms of utility. (III) The ASH heuristic solves the PSM variant with real edge weights to obtain a graph of *arbitrary size* based on a linear programming (LP) formulation, and then performs a linear-time rounding to construct a feasible integer solution to PSM. ASH is slightly

worse than FSH in terms of utility, but it is more efficient and it does not require the size of the graph obtained by the PSM variant as an input parameter. See Section VIII.

5. We conducted experiments with publicly available, real-world, large-scale datasets showing that our heuristics: (I) produced near-optimal solutions; (II) outperformed the classic mechanisms in terms of utility (e.g., they incurred up to 9.9 times lower JS divergence [28]); (III) were substantially more efficient than these mechanisms (e.g., they required a few seconds to protect billion-letter strings, when existing mechanisms did not terminate within 24 hours); and (IV) incurred insignificant or no utility loss in frequency-based mining tasks, unlike these mechanisms. See Section IX.

II. Teaser: Frequency-Based Mining Tasks

We show the applicability of our methods in frequent pattern mining and in frequency-based clustering. Our methods offer rigorous privacy, near-optimal utility and are highly scalable.

Example 1. We constructed the order-3 dBG of the full human genome, whose length is about 3 billion letters [1]. We applied FSH on the sequence M that represents the dBG, with $\epsilon = \delta = 0.0001$, and then produced an (ϵ, δ)-DP multiset $\widehat{E}$. The set of frequent length-k substrings mined from E and $\widehat{E}$ with the smallest possible relative frequency threshold $\tau = 0.0022$ is *identical*, implying no utility loss for frequent pattern mining. Furthermore, the normalized multiplicities of edges in E and $\widehat{E}$ are *almost identical* (see Fig. 2), implying insignificant utility loss for other frequency-based mining tasks. Since the nodes of G are known, we can also publish an (ϵ, δ)-differentially private graph $\widehat{G}(V, \widehat{E})$ whose normalized edge multiplicities are almost identical to those of $G(V, E)$.

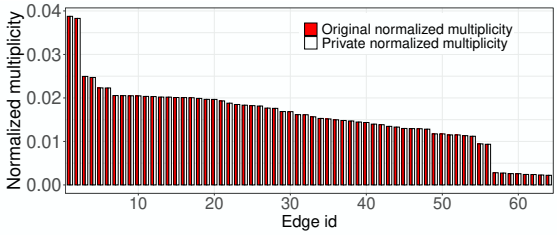


Fig. 2: Normalized multiplicities (in descending order) in an order-3 dBG of the full human genome [1] and in a $(0.0001, 0.0001)$-differentially private order-3 dBG, constructed using FSH and MES.

Example 2. We constructed the order-k dBG for every sequence S_i in a benchmark collection of DNA sequences. We solved PSM on every sequence M_i of multiplicities representing the dBG of sequence S_i, and produced an (ϵ, δ)-differentially private $\widehat{E}_i$. We then constructed two phylogenetic trees: one based on the original multiplicities; and another one based on the differentially private ones. The trees represent clusterings of the collection and were constructed using the methodology in [25]. We then measured how similar the trees are with the well-established normalized Robinson-Foulds (nRF) distance. As can be seen in Fig. 3, for varying $\epsilon = \delta$, our methods resulted in *very similar* trees to those constructed over

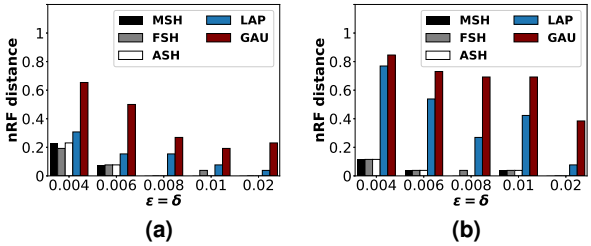


Fig. 3: nRF distance vs. $\epsilon = \delta$ (a) $k = 3$ and (b) $k = 4$.

the original sequences (nRF was very low), unlike the Laplace (LAP) and Gaussian (GAU) mechanisms. The mechanism based on randomized response performed worse than GAU and thus was omitted. We include more details in [2].

III. Background and Related Work

Strings. An *alphabet* Σ is a finite non-empty set whose elements are called *letters*. A *string* is a sequence of letters from Σ. We fix a string $S = S[0]S[1]\cdots S[|S|-1]$ of length $|S|$ over an ordered alphabet Σ. By $S[i\,..\,j] = S[i]\cdots S[j]$, we denote the *substring* of S starting at position i and ending at position j of S. A substring $S[i\,..\,j]$ is called a *prefix* if $i = 0$; it is called a *suffix* if $j = |S| - 1$.

de Bruijn Graphs. We fix an integer $k > 1$. Given S and k, the *de Bruijn graph* (dBG) of order k, over a given string S of length $|S|$, is a directed multigraph $G = (V, E)$, where V is a set of nodes and E is a multiset of edges (i.e., of ordered pairs of nodes). V is the set of length-$(k-1)$ substrings of S. The *multiplicity* of an edge $(u, v) \in E$ is the number of occurrences of the pair (u, v) in the multiset E. The cardinality $|E|$ of E (i.e., the sum of edge multiplicities) is referred to as the *size* of G. G contains an edge (u, v) with multiplicity $m_{u,v}$ if and only if the string $u[0] \cdot v$ is equal to the string $u \cdot v[k-2]$ and these strings occur exactly $m_{u,v}$ times in S. Thus, $|E| = |S| - k + 1$. Let s and t be the nodes of G corresponding, respectively, to the prefix and to the suffix of length $k-1$ of S. The string S corresponds to an Eulerian path in G that starts at s and ends at $t \neq s$ (if $s = t$, it corresponds to an Eulerian cycle starting from s); see Fig. 1. The graph G may contain other Eulerian paths (resp., cycles).

Differential Privacy (DP). DP [15] is a rigorous, well-established privacy principle which ensures that the output of a mechanism (algorithm) is not strongly dependent on any element of an input dataset. A relaxation of DP often leading to enhanced utility is (ϵ, δ)-DP [15], defined as follows. A randomized mechanism $f : \mathcal{D} \to \mathcal{R}$ gives (ϵ, δ)-DP, if for all neighboring datasets D and D' in domain $\mathcal{D}$ and all $R \subseteq \mathcal{R}$, $\Pr[f(D) \in R] \leq \mathrm{e}^{\epsilon} \cdot \Pr[f(D') \in R] + \delta$, where $\epsilon \in (0, \infty)$ and $\delta \in [0, 1]$ are real numbers and e is the base of the natural logarithm. (ϵ, δ)-DP is equivalent to pure ϵ-DP when $\delta = 0$. With $\delta > 0$, there is a small probability that ϵ-DP fails.

We consider two dBGs $G = (V, E)$ and $G' = (V', E')$ to be *neighboring* if $V = V'$ and there exist two nodes $u^*, v^* \in V$ such that $E = E' \bigcup \{(u^*, v^*)\}$ or $E' = E \bigcup \{(u^*, v^*)\}$. The

multiplicities of those neighboring graphs thus follow: $m_{u,v} = m'_{u,v}$ for $(u,v) \neq (u^*, v^*)$ and $|m_{u^*,v^*} - m'_{u^*,v^*}| = 1$.

DP in String Data. There are several works for enforcing DP on a collection of strings [13], [12], [9], [37], [38]. These works output a synthetic collection of strings [13], [12], or analysis results obtained from the input collection [9], [37], [38]. Since these works adopt DP to protect the presence of any individual string in the input collection of strings, they are inapplicable to our setting which calls for protecting length-k substrings of a single string.

DP in Graph Data. A number of studies apply DP to protect edges (i.e., nodes are considered public), while releasing aggregate graph statistics [31], [21] or generating synthetic graphs [33], [24], [34]. We aim at publishing a dBG, so we describe the latter in detail. [33] is set in a decentralized environment where each data owner only sees part of the graph. It is thus inapplicable to our problem setting where the data owner has access to the entire graph. [24] addresses privacy in releasing attributed social graphs where attributes such as age and gender are attached to nodes. [34] proposed to first analyze degree correlation statistics, i.e., dK-graphs, with DP, and then generate synthetic graphs according to the statistics. While [34] may preserve the structural information in the input graph, it is likely to introduce fake edges. Also, unlike most existing studies that work with undirected social network graphs, we release a truthful, directed graph.

IV. Classic DP Mechanisms & Their Limitations

A Classic DP Problem. A pair of nodes u and v in V of G is called *feasible* if $u[0] \cdot v = u \cdot v[k-2]$. For instance, in Fig. 1, edges (ab, ba) and (aa, ab) are feasible. A feasible edge (u,v) is called *true* if its multiplicity $m_{u,v}$ in G is larger than zero and *fake* otherwise. Let $F = (f_1, \ldots, f_{|F|})$ be the sequence in which f_i is the multiplicity of the feasible edge i in G and i represents the lexicographic rank in the set of feasible edges. It can be seen that F represents a classic *histogram* query [15], where each f_i counts the number of occurrences for the i-th feasible edge. Therefore, for neighboring graphs G and G', $|F - F'| = 1$ and $||F - F'||_2 = 1$ as F' and F differ only in one element by one. Thus, we can enforce DP on F by *classic mechanisms* for publishing histograms, such as Laplace or Gaussian [15]. In [2], we discuss these mechanisms in detail and also discuss our MERR mechanism, which is based on randomized response [26].

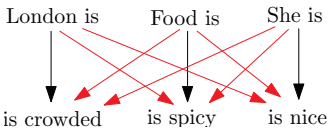
Limitations. The aforementioned classic mechanisms and the advanced approaches for histogram publication that are built on top of those mechanisms (e.g., [21], [36]) have a common significant drawback in our context: they need to consider and perturb *all feasible edges* (i.e., both true and fake feasible edges) in G. The drawback stems from the fact that the number of true feasible edges is often much smaller compared to that of feasible edges, i.e., $|M| \ll |F|$, where M is the sequence of edge multiplicities. To highlight this drawback in our context, we draw a novel connection between fake feasible edges in dBGs and minimal forbidden words [6].

A *minimal forbidden word* of S is a string w that: (I) does not occur in S; and (II) all its proper substrings occur in S. For instance, string $w =$ abc is a minimal forbidden word of $S =$ cbccbcab over $\Sigma = \{$a, b, c$\}$ because: (I) abc does not occur in S; and (II) a, ab, b, bc, c do occur in S. For a fixed k, a string w with $|w| = k$ is a minimal forbidden word of S if and only if $u = w[0\,..\,k-2]$ is a node of G, $v = w[1\,..\,k-1]$ is a node of G, but (u,v) is not in E. In our instance, ab and bc are in V but the edge (ab, bc) is not in E, thus it is a fake feasible edge.

The consideration of feasible edges imposes a *computational efficiency challenge*. The number of *true* feasible edges is upper bounded by $|S| - k + 1 = O(|S|)$. However, the number of minimal forbidden words of length k of S, and thus the number of *fake* feasible edges, is in $O(|\Sigma||S|)$, and this bound is *tight* for sufficiently large $|\Sigma|$ [6]. When $|\Sigma| = \Theta(|S|)$ the bound becomes quadratic, and thus it takes quadratic time to enumerate feasible edges. Thus, the time complexity of Laplace/Gaussian becomes quadratic in $|S|$ and that of MERR even larger by a multiplicative factor of $|S|-k+1$, as $|S|-k+1$ randomized trials are performed for each feasible edge [2].

A large number of fake feasible edges (i.e., many elements with multiplicity zero in F) imposes also a *utility challenge*: Laplace, Gaussian, and MERR mechanisms report positive multiplicities for fake feasible edges, and when the number of fake feasible edges is large the utility loss of these mechanisms is excessive; inspect Example 3 in this regard.

Example 3. Let S be a natural language text. Let $k = 3$ and the following three 3-grams occurring in S: London is crowded, Food is spicy, and She is nice. Then, we have 2-gram nodes: London is, is crowded, Food is, is spicy, She is, and is nice. We thus have the following feasible edges in the graph below (fake feasible edges are in red). Clearly, we observe a larger number of fake feasible than true feasible edges. While applying domain knowledge in post-processing could eliminate some fake feasible edges, e.g., Food is crowded, it may not eliminate all of them, e.g., London is nice.



In real natural language datasets, the large number of distinct words implies a huge number of fake feasible edges, which are not only prohibitively expensive to enumerate but would also yield significant noise in the outcome of the aforementioned mechanisms. For instance, both the number $|\Sigma|$ of distinct 1-grams and the length $|S|$ for the natural language datasets from Wikipedia [3] and Yahoo [4] we used in our experiments are in the order of millions, implying that the number of fake feasible edges may be up to billions, which greatly surpasses the number of true feasible edges.

Even in datasets over a small alphabet, the problems resulting from fake feasible edges remain. In Table I, we present the number of edges that are true feasible, fake feasible, and feasible in the order-k dBG of the full human genome [1] that

k	# true feasible edges	# fake feasible edges	# feasible edges	# fake feasible / # feasible edges (%)
10	1,048,576	0	1,048,576	0
11	4,193,336	968	4,194,304	0.0231
12	16,611,188	158,569	16,769,757	0.946
13	62,327,333	3,619,419	65,946,752	5.488
14	202,881,636	36,453,528	239,335,164	15.231
15	547,221,045	167,419,132	714,640,177	23.427

TABLE I: Number of true feasible, fake feasible, and feasible edges in the order-k dBG of the DNA dataset [1], for $k \in [10, 15]$, as well as the percentage of fake feasible to feasible edges in the same dBG.

has alphabet size 4, for all $k \in [10, 15]$. We used the optimal $O(|\Sigma||S|)$-time algorithm of [7] to make this computation. The percentage of fake feasible edges when k increases is more than 23% of the total feasible edges, even though $|\Sigma| = 4$.

V. MES Mechanism and PSM Problem

MES Mechanism. MES overcomes the limitations of the classic DP mechanisms, as it considers only true feasible edges. Thus, it is significantly faster and improves utility. MES is inspired by the DP mechanism of [22], which cannot be readily applied in our setting, as it protects a user's record (set of query-url-count tuples) in a collection of records.

Given the input graph $G = (V, E)$, MES operates on the sequence of edge multiplicities $M = (m_1, \ldots, m_n)$ extracted from E, which is often significantly shorter than the sequence of feasible edge multiplicities F. MES requires the sequence of edge multiplicities $X = (x_1, \ldots, x_n)$ in the output graph $\widehat{G}(V, \widehat{E})$ to satisfy the following constraints:

$$\begin{cases} \left(\frac{m_i}{m_i - 1}\right)^{x_i} \leq \mathrm{e}^{\epsilon}, & \text{for } i \in [1, n] \quad (1) \\ 1 - \left(\frac{m_i - 1}{m_i}\right)^{x_i} \leq \delta, & \text{for } i \in [1, n] \quad (2) \\ x_i \in \mathbb{Z}, & \text{for } i \in [1, n] \quad (3) \end{cases}$$

where e is the base of the natural logarithm.

Given X that satisfies the above constraints, MES uniformly at random samples edges from the input graph, i.e., from E, to produce an (ϵ, δ)-DP $\widehat{E}$ (see proof of privacy below). Specifically, let (u, v) be an edge with multiplicity m_i. MES views (u, v) as a set of m_i edges each having multiplicity 1 and samples x_i of these edges uniformly at random. As a simple example, consider edge (u, v) with multiplicity $m = 3$, which accounts for three different occurrences of ab in input string $S = \underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}}\mathtt{a}\underline{\mathtt{ab}}$. Suppose $x = 1$ satisfies the constraints above, for some ϵ and δ values. The output $\widehat{E}$, or the graph $\widehat{G}(V, \widehat{E})$, would contain one randomly sampled occurrence of ab, i.e., the multiplicity of (u, v) in $\widehat{E}$ is 1. Note that we only consider edges in the input graph with multiplicity $m_i \geq 2$. For edges with $m_i < 2$, the solution of x_i is trivial, i.e., only $x_i = 0$ would satisfy the constraints.

Theorem 1. *By imposing the constraints in Eqs. 1 to 3,* MES *satisfies* (ϵ, δ)*-DP.*

Proof. Assume two neighboring graphs $G(V, E)$, $G'(V', E')$, with $V = V'$ and $E = E' \bigcup \{e_i^*\}$, and that e_i^* is a single edge with multiplicity 1. Let X be a feasible sequence of output multiplicities for E. We partition the space of all output graphs $\mathcal{R}$ into $\mathcal{R}_1$ and $\mathcal{R}_2$, where $\mathcal{R}_1$ represents all outputs containing the edge e_i^* and $\mathcal{R}_2$ represents those which do not. Since E' does not contain e_i^*, $\Pr[\text{MES}(E') \in \mathcal{R}_1] = 0$. On the other hand, outputting e_i^* for E is equivalent to sampling e_i^* at least once among x_i independent trials, and it occurs with probability $\Pr[\text{MES}(E) \in \mathcal{R}_1] = 1 - \Pr[\mathcal{E}]$, where $\mathcal{E}$ is the event "e_i^* is not sampled in any trial". In the following, we compute $\Pr[\mathcal{E}]$. The probability of the event "e_i^* is sampled in a trial" is $\frac{1}{m_i}$, since we sample uniformly at random. Thus, the probability of the event "e_i^* is not sampled in a trial" is $1 - \frac{1}{m_i} = \frac{m_i - 1}{m_i}$. Since the x_i trials are independent, $\Pr[\mathcal{E}] = (\frac{m_i - 1}{m_i})^{x_i}$. Thus,

$$\Pr[\text{MES}(E) \in \mathcal{R}_1] = 1 - (\frac{m_i - 1}{m_i})^{x_i} \leq \delta. \quad (4)$$

The inequality holds due to Eq. 2. Let us now consider any output $r \in \mathcal{R}_2$. We have that:

$$\begin{aligned} \frac{\Pr[\text{MES}(E') = r]}{\Pr[\text{MES}(E) = r]} &= \frac{\Pr[\text{MES}(E') \in \mathcal{R}_2]}{\Pr[\mathcal{E}]} \\ &= \frac{1 - \Pr[\text{MES}(E') \in \mathcal{R}_1]}{\Pr[\mathcal{E}]} = \\ &= \frac{1 - 0}{(\frac{m_i - 1}{m_i})^{x_i}} = (\frac{m_i}{m_i - 1})^{x_i} \leq e^{\epsilon}. \quad (5) \end{aligned}$$

The last inequality holds due to Eq. 1. Similarly, the same bound can be proved for $E' = E \bigcup \{e_i^*\}$.

Last, we show that for E and E' and any subset S of $\mathcal{R}$, (ϵ, δ)-DP holds. Let $S = S_1 \cup S_2$, where $S_1 = S \cap \mathcal{R}_1$, and $S_2 = S \cap \mathcal{R}_2$. We have that

$$\begin{aligned} \Pr[\text{MES}(E') \in S] &= \Pr[\text{MES}(E') \in S_1] + \Pr[\text{MES}(E') \in S_2] \\ &\leq 0 + e^{\epsilon} \Pr[\text{MES}(E) \in S_2] \\ &\leq e^{\epsilon} \Pr[\text{MES}(E) \in S] + \delta, \end{aligned}$$

so (ϵ, δ)-DP holds. The first inequality holds from $\Pr[\text{MES}(E') \in \mathcal{R}_1] = 0$ and Eq. 5. The second inequality holds from $\Pr[\text{MES}(E') \in S_2] \leq \Pr[\text{MES}(E') \in S]$, since $S_2 \subseteq S$, and from $\delta \geq 0$. □

For simplicity, we introduce $\gamma = \min(\epsilon, \ln(\frac{1}{1-\delta}))$. The sequence X of MES consists of integers x_i satisfying Eqs. 1 and 2: $x_i \in [0, \lfloor \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \rfloor]$. This is because solving Eqs. 1 and 2 for x_i, gives $x_i \leq \epsilon \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1}$ and $x_i \leq \ln(\frac{1}{1-\delta}) \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1}$, respectively, and combining them gives $x_i \leq \min(\epsilon, \ln(\frac{1}{1-\delta})) \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} = \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1}$. Due to Eq. 3, the following also holds:

$$x_i \leq \lfloor \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \rfloor. \quad (6)$$

Note that each single edge of G corresponds to an occurrence of a length-k substring of S. Thus, it is natural to consider $x_i \leq m_i$, for all i, i.e., to sample a subset of occurrences. Furthermore, a small x_i would likely satisfy Eq. 1, as $(\frac{m_i}{m_i - 1}) > 1$. To ensure that $x_i \leq m_i$, for all i, we require $\gamma \leq 1$.

Lemma 1. *When* $\gamma \leq 1$*,* $x_i \leq m_i$ *holds for each* x_i*,* $i \in [1, n]$*.*

We include all omitted proofs in [2].

PSM Problem. PSM aims to find a useful sequence X for frequency-based mining tasks that can be used by MES.

Problem 1 (PSM)**.** Given a sequence $M = (m_1, \ldots, m_n)$ of n integers, such that each $m_i \geq 2$, $i \in [1, n]$, along with a real number $\gamma \leq 1$, find a sequence $X = (x_1, \ldots, x_n)$ so as to minimize

$$L_1(M, X) = \sum_{i \in [1,n]} |\frac{m_i}{\sum_{j \in [1,n]} m_j} - \frac{x_i}{\sum_{j \in [1,n]} x_j}|$$

subject to:

(I) $x_i \leq \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1}$, for all $i \in [1, n]$,
(II) $x_i \in \mathbb{Z}_{\geq 0}$, for all $i \in [1, n]$ (Integer edge multiplicities), and
(III) $\sum_{i \in [1,n]} x_i \geq 1$ (Nonempty graph $\widehat{G}$).

We require the sequence X to have a minimal normalized L_1 distance from the sequence $M = (m_1, \ldots, m_n)$ of multiplicities that represents the edge multiset E of the input graph G. We also require X to be the non-zero sequence and thus $\widehat{G}$ to be nonempty (Constraint III), so that the objective function $L_1(M, X)$ is defined. Note that requiring $\gamma \leq 1$ is equivalent to requiring $\epsilon \leq 1$ or $\delta \leq 1 - \frac{1}{\mathrm{e}}$ (this follows from the definition of γ), which is aligned with our goal of providing strong (ϵ, δ)-DP. Last, note that we only need to solve PSM for edges with multiplicity $m_i \geq 2$. As discussed before, MES sets $x_i = 0$ for every other edge (if any) and then produces an (ϵ, δ)-DP $\widehat{E}$, or, equivalently, an (ϵ, δ)-DP graph $\widehat{G}(V, \widehat{E})$.

VI. Exact Algorithm for PSM

Our EXACT-DP algorithm is founded upon two crucial observations: (I) PSM for a graph $\widehat{G}$ of fixed integer size $\zeta = \sum_{j \in [1,n]} x_j$ and $x_j \in \mathbb{Z}_{\geq 0}$, for each $j \in [1, n]$, can be solved exactly using dynamic programming. (II) The size of $\widehat{G}$ is no larger than that of G (due to Lemma 1), which is no larger than the length of string S. Thus, PSM can be solved optimally in polynomial time in $|S|$, by applying dynamic programming for all possible integer ζ's and selecting the solution X with the minimum normalized L_1 distance to M.

We first outline the dynamic programming procedure for fixed ζ and then our EXACT-DP algorithm.

Algorithm for Fixed ζ. The main idea is to fill up a $(\zeta+1) \times n$ dynamic programming matrix $\mathcal{A}$ in which cell $\mathcal{A}[i][j]$ stores the optimal cost of a partial solution $(x_1, \ldots, x_j)$ of size $x_1 + \ldots + x_j = i$. This cost is computed by considering all ways of constructing the partial solution from a partial solution $(x_1, \ldots, x_{j-1})$ and size $i - r$ and an element x_j (partial solution of length 1) of size r, for each $r \in [0, i]$. Specifically, we implement this in the following steps:

1. Let $f_i = \frac{m_i}{\sum_{l=1}^{n} m_l}$. Fill a $(\zeta + 1) \times n$ matrix $\mathcal{A}$ as follows:

- 0-th row initialization: For $j \in [0, n-1]$, entry $\mathcal{A}[0][j]$ contains the cost of assigning $(x_1, \ldots, x_j)$ (i.e., the first j elements of X) to zero, which is computed by:

$$\mathcal{A}[0][j] = \begin{cases} \sum_{r \in [0,j]} f_{r+1}, & j \in [0, n-2] \\ \infty, & \text{otherwise} \end{cases}.$$

  The cost is ∞ for assigning all n elements of X to zero, since this violates Constraint III of Problem PSM.

- 0-th column initialization: For $i \in [1, \zeta]$, entry $\mathcal{A}[i][0]$ contains the cost of assigning $x_1 = i$, which is computed by:

$$\mathcal{A}[i][0] = \begin{cases} |f_1 - \frac{i}{\zeta}|, & i \in [1, \zeta] : i \leq \gamma \cdot (\ln(\frac{m_1}{m_1 - 1}))^{-1} \\ \infty, & i \in [1, \zeta] : i > \gamma \cdot (\ln(\frac{m_1}{m_1 - 1}))^{-1} \end{cases}.$$

  The cost is ∞ when $x_1 = i$ violates Constraint I of Problem PSM.

- Filling up all other entries: For $i \in [1, \zeta]$, $j \in [1, n-1]$, entry $\mathcal{A}[i][j]$, contains the optimal cost of assigning $(x_1, \ldots, x_j)$, so that $\sum_{l=1}^{j} x_l = i$, which is computed by:

$$\mathcal{A}[i][j] = \min_{r \in [0,i]} \{\mathcal{A}[i - r][j - 1] + |f_{j+1} - \frac{r}{\zeta}|\},$$

  if there is $r \in [0, i]$ such that $r \leq \gamma \cdot (\ln(\frac{m_{j+1}}{m_{j+1} - 1}))^{-1}$ and by $\mathcal{A}[i][j] = \infty$, otherwise. The cost is computed as the minimum of all possible costs for assigning $(x_1, \ldots, x_{j-1})$ and x_j so that $\sum_{l=1}^{j} x_l = i$, when Constraint I of PSM is satisfied. When this constraint is violated the cost is ∞.

2. Fill an auxiliary $(\zeta + 1) \times n$ matrix $\mathcal{B}$ as follows:

- $\mathcal{B}[0][j] = \begin{cases} \langle 0, j-1 \rangle, & j \in [0, n-2] \\ \infty, & \text{otherwise} \end{cases}$, for $j \in [1, n]$.
- $\mathcal{B}[i][0] = \begin{cases} \langle i, -1 \rangle, & i \in [1, \zeta] : i \leq \gamma \cdot (\ln(\frac{m_1}{m_1 - 1}))^{-1} \\ \infty, & i \in [1, \zeta] : i > \gamma \cdot (\ln(\frac{m_1}{m_1 - 1}))^{-1} \end{cases}$, for $i \in [1, \zeta]$.
- $\mathcal{B}[i][j] = \begin{cases} \langle r^*, i - r^* \rangle, & r^* \in \arg\min_{r \in [0,i]} \{\mathcal{A}[i - r][j - 1] \\ & \quad + |f_{j+1} - \frac{r}{\zeta}|\}, \text{if there is } r \in [0, 1] \\ & \quad \text{such that } r \leq \gamma \cdot (\ln(\frac{m_{j+1}}{m_{j+1} - 1}))^{-1} \\ \infty, & \text{otherwise} \end{cases}$

  for $i \in [1, \zeta]$, $j \in [1, n-1]$.

Each entry $\mathcal{B}[i][j]$ corresponds to $\mathcal{A}[i][j]$ and contains the value of the j-th element of X that has optimal cost, and a pointer to the next entry of $\mathcal{B}$ that will be followed during backtracking in Step 3 below.

3. Construct X by following a standard backtracking procedure starting from $\mathcal{B}[\zeta][n-1]$: If $\mathcal{B}[\zeta][n-1] = \infty$, report FAIL, since PSM has no solution for the given ζ. Otherwise, set $x_n = \mathcal{B}[\zeta][n-1].1$ and backtrack to $\mathcal{B}[\mathcal{B}[\zeta][n-1].2][n-2]$, where .1 and .2 denote the first and second element of the tuple in $\mathcal{B}[\zeta][n-1]$, respectively. The process continues similarly, until reaching an entry $\mathcal{B}[i][0]$, $i \in [0, \zeta]$. At this point, we set $x_1 = \mathcal{B}[i][0].1$ and return X.

Lemma 2. *The dynamic programming procedure for fixed ζ solves exactly* PSM *with fixed ζ in $O(\zeta^2 n)$ time using $O(\zeta n)$ space.*

EXACT-DP Algorithm. The minimal value of $\sum_{i \in [1,n]} x_i$ is 1 (due to Constraint III of PSM) and its maximal value is $\sum_{i \in [1,n]} \lfloor \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \rfloor$ (this follows from Constraint I of PSM). We thus solve PSM by executing the dynamic programming procedure for every integer $\zeta \in [1, \sum_{i \in [1,n]} \lfloor \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \rfloor]$ and returning the solution with the minimum

cost. This is precisely EXACT-DP. Note that, using the maximal ζ value does not necessarily minimize $L_1(M, X)$, as this function is not monotonic with respect to ζ [1].

Time and space complexity of EXACT-DP. While in general $\sum_{i\in[1,n]} \lfloor \gamma \cdot (\ln(\frac{m_i}{m_i-1}))^{-1} \rfloor$ is unbounded, in PSM it is polynomial in $|E| = \sum_{i\in[1,n]} m_i$, the size of G, as shown below. Specifically, Lemma 1 leads to the following corollary.

Corollary 2. *For* $\gamma \leq 1$, $\sum_{i=1}^{n} \lfloor \gamma \cdot (\ln(\frac{m_i}{m_i-1}))^{-1} \rfloor \leq |E|$.

Corollary 2, in turn, leads to the following result.

Theorem 3. PSM *can be solved in* $O(|E|^4)$ *time using* $O(|E|^2)$ *space.*

Since for any string S we have $n \leq |E| = |S| - k + 1 = O(|S|)$, EXACT-DP takes $O(|S|^4)$ time and $O(|S|^2)$ space.

VII. EXACT ALGORITHMS FOR PSM WITH REAL EDGE WEIGHTS

As EXACT-DP is impractical for large-scale input datasets, we relax the requirement that solution X is comprised of only integer values. We develop exact algorithms for when X may be comprised of real values. Our results here are of independent interest: one could consider a graph G consisting of a set of edges with real weights instead of a multigraph.

Algorithm for Fixed ζ. Consider a PSM variant with a fixed output graph size $\zeta = \sum_{i=1}^{n} x_i$ and $x_i \in \mathbb{R}_{\geq 0}$, for each $i \in [1, n]$. We refer to this variant as PSM$_\zeta$. Clearly, the objective function of PSM$_\zeta$ is $\sum_{i=1}^{n} |\frac{m_i}{\sum_{j=1}^{n} m_j} - \frac{x_i}{\zeta}|$, or $\sum_{i=1}^{n} |f_i - \frac{x_i}{\zeta}|$, if we define $f_i = \frac{m_i}{\sum_{j=1}^{n} m_j}$.

An exact, linear-time algorithm for the PSM$_\zeta$ variant is provided in Algorithm 1. The intuition is as follows. Let $a_i = \gamma \cdot (\ln(\frac{m_i}{m_i-1}))^{-1}$. We classify the n items into two types, depending on the minimum between α_i and $\zeta \cdot f_i$. For all items of the first type, we set $x_i = \alpha_i$. For all other items, we set $x_i = \zeta \cdot f_i$, as this leads to $|f_i - \frac{x_i}{\zeta}| = 0$. Notice that this solution is infeasible, since $\sum_{i=1}^{n} x_i < \zeta$. Thus, we distribute any remaining budget $b = \zeta - \sum_{i=1}^{n} x_i$ over items of the second type, i.e., until ζ is reached, but not exceeded, and without violating the constraints $x_i \leq \alpha_i$.

Theorem 4. FIXED-SIZE-LINEAR *solves* PSM$_\zeta$ *in* $O(n)$ *time using* $O(n)$ *space.*

EXACT-LP Algorithm. Specifying ζ is generally useful (e.g., to control the size of graph $\widehat{G}$). Yet, it is not necessary in applications, such as in frequent pattern mining or frequency-based clustering, where high accuracy is achieved by simply preserving the normalized multiplicities of edges. We thus propose an algorithm, referred to as PSM$_{\text{real}}$, which solves the variant of the PSM problem with real edge weights.

Our algorithm is based on a linear programming (LP) formulation of the problem. We obtain the LP in two steps. First, we use the Charnes-Cooper transformation [10] on the objective function $\sum_{i=1}^{n} |f_i - \frac{x_i}{\sum_{j=1}^{n} x_j}|$ of PSM, where $f_i = \frac{m_i}{\sum_{j=1}^{n} m_j}$. Specifically, we introduce a new variable $y_0 = \frac{1}{\sum_{j=1}^{n} x_j}$. Due to Constraint III in PSM, y_0 is always defined. This leads to the following formulation:

$$\min \quad \sum_{i=1}^{n} |f_i - x_i \cdot y_0| \tag{7a}$$

$$\text{s.t.} \quad y_0 \cdot \sum_{j=1}^{n} x_j = 1 \tag{7b}$$

$$x_i \cdot y_0 \leq \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \cdot y_0, \; i \in [1, n] \tag{7c}$$

$$y_0 \in \mathbb{R}_{\geq 0} \tag{7d}$$

$$x_i \in \mathbb{R}_{\geq 0}, \qquad i \in [1, n]. \tag{7e}$$

Eq. 7b is due to the introduction of y_0 and Eq. 7c is by MES. Next, we prevent the quadratic term $x_i \cdot y_0$ in Eq. 7a, by substituting $x_i \cdot y_0$ with y_i. Since $y_0 > 0$, we obtain:

$$\min \quad \sum_{i=1}^{n} |f_i - y_i| \tag{8a}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} y_j = 1 \tag{8b}$$

$$y_i \leq \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \cdot y_0, \quad i \in [1, n] \tag{8c}$$

$$y_0 \in \mathbb{R}_{\geq 0} \tag{8d}$$

$$y_i \in \mathbb{R}_{\geq 0}, \qquad i \in [1, n]. \tag{8e}$$

At this point, we observe that the resulting LP has a trivial solution. Indeed, we set $y_i = f_i$, for each $i \in [1, n]$, and choose an arbitrarily large $y_0 \geq \max_{i\in[1,n]}\{y_i \cdot \ln(\frac{m_i}{m_i-1}) \cdot \gamma^{-1}\}$ so that Eq. 8c holds. The solution of this LP is comprised of $y_0, \ldots, y_n$. We obtain $X = (x_1, \ldots, x_n)$ by setting $x_i = y_i/y_0$. For example, using $y_0 = \max_{i\in[1,n]}\{y_i \cdot \ln(\frac{m_i}{m_i-1}) \cdot \gamma^{-1}\}$ leads to the largest output graph with the optimal L_1 distance of zero. The resulting algorithm is referred to as EXACT-LP.

Theorem 5. EXACT-LP *solves* PSM$_{real}$ *in* $O(n)$ *time using* $O(n)$ *space.*

Algorithm 1 FIXED-SIZE-LINEAR

```
for each i ∈ [1, n] do
    α_i ← γ · (ln(m_i/(m_i − 1)))^{−1}
    x_i ← min{ζ · f_i, α_i}
A ← {i | α_i < ζ · f_i}                                   ▷ Items of first type
b ← ζ − (Σ_{i∈A} α_i + ζ · Σ_{i∈{1,...,n}\A} f_i)          ▷ Budget b
for each i ∈ {1, ..., n} \ A do                           ▷ Items of second type
    if α_i − ζ · f_i ≤ b then
        x_i ← α_i
        b ← b − (α_i − ζ · f_i)
    else
        x_i ← x_i + b
        break
return X ← (x_1, ..., x_n)
```

VIII. HEURISTICS FOR PSM

We present three heuristics for PSM, which are motivated by the inability of EXACT-DP to scale to large datasets.

Maximum Output Graph Size Heuristic (MSH). MSH returns $X = (\lfloor \gamma \cdot (\ln(\frac{m_1}{m_1-1}))^{-1} \rfloor, \ldots, \lfloor \gamma \cdot (\ln(\frac{m_n}{m_n-1}))^{-1} \rfloor)$, a

[1] To see this, note that for $M = (2, 2)$ and $X_1 = (1, 0)$, $X_2 = (1, 1)$, $X_3 = (1, 2)$ with sizes $|X_1| < |X_2| < |X_3|$, $L_1(M, X_1) = 1 > L_1(M, X_2) = 0 < L_1(M, X_3) = \frac{1}{3}$.

feasible solution to PSM where each x_i, $i \in [1, n]$, is as large as possible. Our heuristic takes $O(n)$ time: $O(n)$ time to read the input of PSM and $O(n)$ time to output X.

MSH works well for graphs with large multiplicities because, for two sufficiently large multiplicities, say m_1 and m_2 such that $m_1 \approx m_2$, we have $\ln(\frac{m_1}{m_1-1}) \approx \ln(\frac{m_2}{m_2-1})$. This is because the derivative $\frac{d}{dx}\ln(\frac{x}{x-1}) = -\frac{1}{x\cdot(x-1)}$ goes to 0 for sufficiently large x. Thus, $\gamma \cdot \ln(\frac{m_1}{m_1-1})^{-1} \approx \gamma \cdot \ln(\frac{m_2}{m_2-1})^{-1}$. For instance, let $\gamma = 0.5$, $m_1 = 999$, and $m_2 = 1001$. We have $\ln(\frac{m_1}{m_1-1}) = 0.0010015$ and $\ln(\frac{m_2}{m_2-1}) = 0.000999$, and thus $x_1 = 499$ and $x_2 = 500$. Since $x_1 \approx x_2$, their corresponding frequency in X will also be approximately the same: $x_1/\sum_{i=1}^{n}\lfloor\gamma \cdot (\ln(\frac{m_i}{m_i-1}))^{-1}\rfloor \approx x_2/\sum_{i=1}^{n}\lfloor\gamma \cdot (\ln(\frac{m_i}{m_i-1}))^{-1}\rfloor$, and $|\frac{m_1}{\sum_{j=1}^{n} m_j} - \frac{x_1}{\sum_{i=1}^{n}\lfloor\gamma\cdot(\ln(\frac{m_i}{m_i-1}))^{-1}\rfloor}| \approx |\frac{m_2}{\sum_{j=1}^{n} m_j} - \frac{x_2}{\sum_{i=1}^{n}\lfloor\gamma\cdot(\ln(\frac{m_i}{m_i-1}))^{-1}\rfloor}|$. When many m_i's are large and similar, the latter will hold for many x_i's, $L_1(M, X)$ will be small, and MSH will perform well. This is enhanced by the fact that $L_1(X, M)$ is mostly affected by large multiplicities.

Fixed Output Graph Size Heuristic (FSH). We apply the linear-time FIXED-SIZE-LINEAR algorithm and transform its solution X to a solution X' of PSM, in linear time, by applying the following rounding scheme to each x_i, $i \in [1, n]$:

$$x'_i = \lceil x_i \rceil, \text{ if } \lceil x_i \rceil - x_i \leq \frac{1}{2} \text{ and } \lceil x_i \rceil \leq \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \quad (9)$$

$$x'_i = \lfloor x_i \rfloor, \text{ otherwise.} \quad (10)$$

Note that each x'_i satisfies Constraint I in PSM, either because of Eq. 9 or because x_i already satisfies Constraint I and setting x'_i as in Eq. 10 implies $x'_i \leq x_i$. In our experiments, we show that ζ can be configured to produce a large graph with good L_1 distance and also investigate the impact of ζ on utility.

Arbitrary Output Graph Size Heuristic (ASH). In ASH, we apply the linear-time EXACT-LP algorithm by setting $y_i = f_i$ and then $y_0 = \max_{i\in[1,n]}\{y_i \cdot \ln(\frac{m_i}{m_i-1}) \cdot \gamma^{-1}\}$. Since the values of $x_i = y_i/y_0$ in the output of the algorithm are generally not integers, we employ the (linear-time) rounding of Eqs. 9 and 10 to obtain a feasible solution X' to PSM.

IX. EXPERIMENTAL EVALUATION

We evaluated the effectiveness and efficiency of our heuristics by comparing them to our exact algorithm (EDP) and to the classic mechanisms of Section IV.

Data. We used three large datasets: (I) The full human genome v. hg38 (HUM) [1]. (II) Wikipedia N-grams (WIK) [3], which was extracted from the corpus of the entire English Wikipedia. (III) Yahoo! N-grams, v. 2.0 (YAH) [4], which was extracted from a corpus of 14.6 million documents. Each element in WIK (respectively, YAH) is a k-gram (i.e., a true feasible edge) with $k \in [4, 7]$ (respectively, $k \in [3, 5]$). We also used a smaller dataset on which EDP could run in a reasonable amount of time: the complete genome of Coronavirus (COR) [5], extracted from a patient in Australia. Table IIa summarizes characteristics of the datasets we used and Table IIb shows parameter values we used. Note that strong DP would require $\delta \leq \frac{1}{|E|}$. However, we used larger δ values which are comparable to or stricter than those in other DP works (e.g., [22], [23], [29]) to *trade-off privacy for utility*.

Dataset	Alphabet size	dBG order k	Edge multiset size $\lvert E\rvert$
HUM	4	$[3, 15]$ (**11**)	2,937,639,113-k+1 (**2,937,639,103**)
WIK	4,908,001	$[4, 7]$ (**5**)	$[2.05 \cdot 10^8,\ 5.26 \cdot 10^8]$ (**$3.3 \cdot 10^8$**)
YAH	3,475,482	$[3, 5]$ (**5**)	$[2.9 \cdot 10^8,\ 1.74 \cdot 10^9]$ (**$2.9 \cdot 10^8$**)
COR	4	$[3, 5]$ (**4**)	$\{29890, 29891, 29892\}$ (**29891**)

(a)

Dataset	$\epsilon = \delta = \gamma = \min(\epsilon, \ln(\frac{1}{1-\delta}))$	Relative frequency threshold τ
HUM	$[10^{-6}, 0.3]$ (**0.01**)	$[5 \cdot 10^{-6}, 5 \cdot 10^{-4}]$ (**$5 \cdot 10^{-4}$**)
WIK	$[0.001, 0.5]$ (**0.01**)	$[5 \cdot 10^{-6}, 5 \cdot 10^{-6}]$ (**$5 \cdot 10^{-6}$**)
YAH	$[0.001, 0.5]$ (**0.01**)	$[10^{-6}, 10^{-3}]$ (**$6 \cdot 10^{-6}$**)
COR	$[0.01, 0.1]$ (**0.05**)	Not used

(b)

TABLE II: (a) Dataset characteristics. (b) Values of parameters for each dataset (default values are in bold).

Experimental Setup. We evaluated data utility based on: (I) The similarity between the normalized multiplicities of edges in G and $\widehat{G}$, which is captured by L_1 distance, or by JS divergence (JSD) [28]. $L_1 = 0$ or JSD $= 0$ implies no accuracy loss in frequency-based mining tasks. (II) The accuracy of frequent length-k pattern mining captured by the $F1$ score (a.k.a F measure). Given a threshold $\tau \in [0, 1]$, let F_G (resp., $F_{\widehat{G}}$ and $F_{G,\widehat{G}}$) be the number of edges with normalized multiplicity at least τ in G (resp., in $\widehat{G}$ and in both G and $\widehat{G}$). $F1 = \frac{2\cdot\text{Prec}\cdot\text{Rec}}{\text{Prec}+\text{Rec}}$, where $\text{Prec} = \frac{F_{G,\widehat{G}}}{F_{\widehat{G}}}$ and $\text{Rec} = \frac{F_{G,\widehat{G}}}{F_G}$. $F1 = 1$ implies no accuracy loss. To stress test our methods, we used rather small values of τ (see Table IIb).

We compared our methods (all using MES) against the Laplace (LAP), Gaussian (GAU), or MERR mechanisms. As discussed in Section IV, these mechanisms have to operate on *all feasible edges*, which negatively impacts both their utility and efficiency. In our implementation, we rounded the output of LAP and GAU to the closest integer and replaced negative values with 0 and values larger than $|E|$ with $|E|$. We report the average result over 1,000 runs of each mechanism.

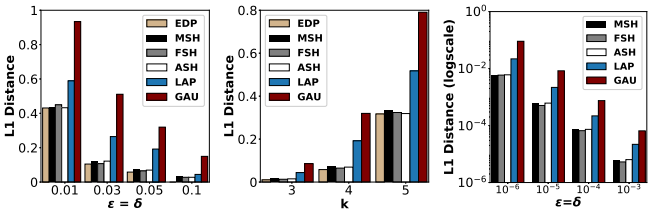
We do not report results for LAP and GAU on WIK and YAH because these mechanisms did not terminate within 24 hours. The reason is the enumeration of fake feasible edges that was too costly, due to the large alphabet of these datasets (see Section IV). Even for HUM the enumeration took one hour using [7], making LAP and GAU orders of magnitude less efficient than our heuristics. *We omit all results for MERR, as it was the slowest and worst in terms of utility mechanism.*

The default ζ in FSH was the one that led to the lowest L_1 distance after rounding, among all ζ's satisfying $\zeta = x \cdot \zeta_{\max}$, where x is a real number in $[0, 1]$ with two fractional digits and $\zeta_{\max} = \sum_{i\in[1,n]}\lfloor\gamma \cdot (\ln(\frac{m_i}{m_i-1}))^{-1}\rfloor$ is the maximum possible ζ. We investigate the impact of ζ on utility later.

All experiments ran on an Intel Xeon E5-2640@2.66GHz CPU with 160GB RAM running GNU/Linux. Our code is available at: https://bitbucket.org/string-dp/icdm-2021/.

Comparison with our Exact Algorithm. Our heuristics produced near-optimal solutions substantially outperforming the classic mechanisms (see Figs. 4a and 4b; results for JSD were analogous (omitted)). LAP outperformed GAU, but it was

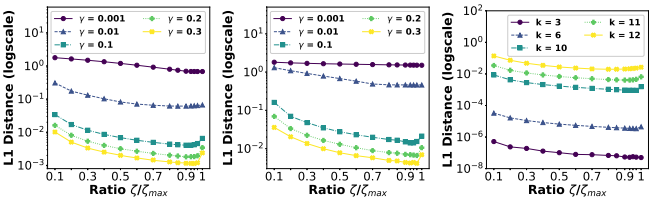
always much worse than our heuristics. In these experiments, there were no fake feasible edges; if there were, they would lower the utility of the classic mechanisms only. As expected, our heuristics were faster than EDP by orders of magnitude. We also show in Fig. 4c that, for the HUM dataset and $k = 3$, our heuristics substantially outperformed both LAP and GAU. In this experiment, EDP did not finish within 24 hours.



(a) COR **(b)** COR **(c)** HUM

Fig. 4: L_1 distance vs. (a) $\gamma = \epsilon = \delta$, (b) k, and (c) $\gamma = \epsilon = \delta$.

Impact of ζ on Utility. Fig. 5 shows how L_1 distance is affected by the parameter ζ used in FSH. We varied ζ in $[0.1 \cdot \zeta_{\max}, \zeta_{\max}]$, where the maximum possible ζ is $\zeta_{\max} = \sum_{i \in [1,n]} \lfloor \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \rfloor$. Increasing ζ improves L_1 distance but up to one point, since L_1 distance is not monotonic with respect to ζ (see Section VI). Interestingly, there are relatively large ζ's (in $[0.9, 1]$) for which FSH achieves the best result with these datasets. This implies that useful graphs of large size can be produced. The results with respect to other utility measures and datasets were analogous (omitted).

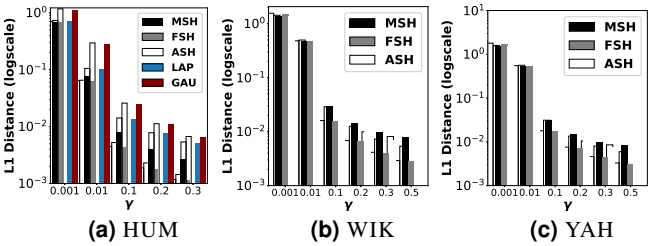


(a) HUM **(b)** WIK **(c)** HUM

Fig. 5: (a, b) L_1 distance vs. ratio $\zeta/\zeta_{\max}$ for varying $\gamma = \epsilon = \delta$. (c) L_1 distance vs. ratio $\zeta/\zeta_{\max}$ for varying k.

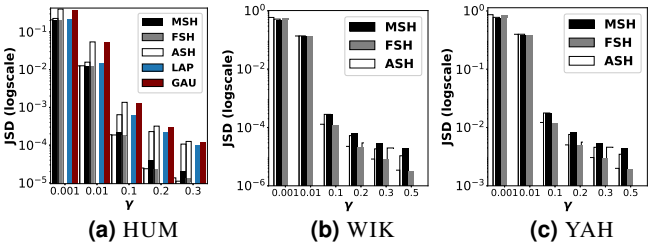
Impact of γ on Utility. Figs. 6 and 7 show the L_1 and JSD values as a function of γ (we set $\gamma = \epsilon = \delta$). We omit smaller γ values from these experiments, as our heuristics performed similarly for them, and show some large γ values solely to stress that our heuristics scale well with respect to utility.

Larger γ led to better utility, due to the privacy/utility trade-off. However, all our heuristics outperformed both LAP and GAU in all cases; the difference increases with γ, as larger γ provides more room for optimizing utility. Our heuristics performed similarly for $\gamma \leq 0.01$, while FSH and ASH outperformed MSH for larger γ. The reason is that, in the former case, FSH and ASH set x_i to $\gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1}$ for most i values, which is similar to what MSH does as well (recall that MSH sets each x_i to $\lfloor \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \rfloor$), whereas in the latter case there were smaller multiplicities that are specifically optimized by FSH and ASH (e.g., FSH sets $x_i = \zeta \cdot f_i$



(a) HUM **(b)** WIK **(c)** YAH

Fig. 6: (a, b, c) L_1 Distance vs. $\gamma = \epsilon = \delta$.

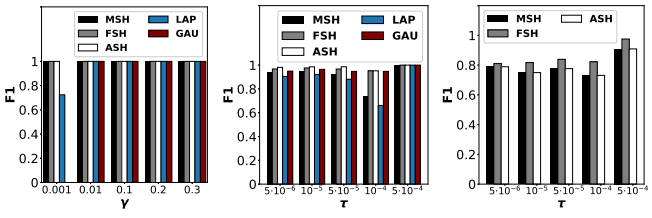


(a) HUM **(b)** WIK **(c)** YAH

Fig. 7: (a, b, c) JSD vs. $\gamma = \epsilon = \delta$.

for each such multiplicity m_i). As expected, FSH generally outperformed ASH, since it specifically optimized L_1 distance after rounding due to the selection of ζ, while ASH optimized L_1 distance for the problem with real weights (i.e., without considering the impact of rounding on L_1 distance). Also, our heuristics incurred no accuracy loss in frequent length-k pattern mining even for small γ, since the $F1$ score was always 1, unlike LAP and GAU (see Fig. 8a; the results for the other datasets were analogous and have been omitted).

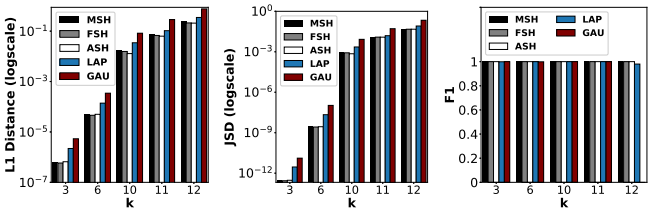
Impact of τ on Utility. Figs. 8b and 8c show the $F1$ scores for varying τ. The results suggest that our heuristics permit accurate length-k frequent pattern mining and that FSH generally outperformed ASH and MSH. Even MSH performed very well; its scores were on average 0.9 and 0.8 for HUM and WIK, respectively. We report analogous results for YAH in [2].



(a) HUM **(b)** HUM **(c)** WIK

Fig. 8: $F1$ vs. (a) $\gamma = \epsilon = \delta$ ($F1$ for GAU was undefined for $\gamma = 0.001$, as $F_{\widehat{G}} = 0$ (no edge was frequent in $\widehat{G}$)), and (b, c) τ.

Impact of k on Utility. Fig. 9 shows the impact of k on utility. A larger k leads to many edges with small multiplicities that incur utility loss with respect to L_1 and JSD, as these measures consider the multiplicities of all feasible edges. Our heuristics substantially outperformed LAP and GAU. $F1$ was mostly affected by edges with large multiplicities, so all methods

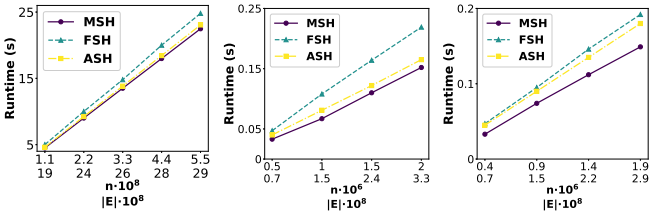


(a) HUM **(b)** HUM **(c)** HUM

Fig. 9: (a) L_1, (b) JSD, and (c) $F1$ vs. k for HUM ($F1$ for GAU was undefined for $k = 12$, as no edge was frequent in $\widehat{G}$).

performed similarly. Analogous results for WIK and YAH are shown in [2].

Efficiency. Fig. 10 shows the impact of the size n of sequence M on runtime. We used $k = 15$ (resp., $k = 4$) for HUM (resp., for WIK and YAH) and random subsets of the edge multiset E of each dataset (smaller subsets were contained in all larger ones). All our heuristics scale linearly with n, as expected by their $O(n)$-time complexity, and they are very efficient, requiring less than 25 seconds to protect HUM whose length $|S|$ is about 3 billion letters (see Fig. 10a). MSH was the fastest (since it fixes each x_i to $\lfloor \gamma \cdot (\ln(\frac{m_i}{m_i - 1}))^{-1} \rfloor$), ASH was slightly slower (due to rounding), and FSH was the slowest (due to budget redistribution and rounding).



(a) HUM **(b)** WIK **(c)** YAH

Fig. 10: Runtime vs. increasing n (and $|E|$).

ACKNOWLEDGMENTS

HC is supported by a CSC scholarship, LF by the National Science Foundation CNS-1951430, GL by the Leverhulme Trust RPG-2019-399, and LS by NWO through Gravitation-grant NETWORKS-024.002.003. This paper is part of the PANGAIA and ALPACA projects that have received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreements No 872539 and 956229, respectively.

REFERENCES

[1] https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.26/.
[2] https://bitbucket.org/string-dp/icdm-2021/src/master/sup.pdf.
[3] https://nlp.cs.nyu.edu/wikipedia-data/.
[4] https://webscope.sandbox.yahoo.com/catalog.php?datatype=l.
[5] https://www.ncbi.nlm.nih.gov/nuccore/MT007544.
[6] Y. Almirantis, P. Charalampopoulos, J. Gao, C. S. Iliopoulos, M. Mohamed, S. Pissis, and D. Polychronopoulos. On avoided words, absent words, and their application to biological sequence analysis. *Algorithms Mol. Biol.*, 12(1):5:1–5:12, 2017.
[7] C. Barton, A. Héliou, L. Mouchard, and S. Pissis. Linear-time computation of minimal absent words using suffix array. *BMC Bioinform.*, 15:388, 2014.
[8] G. Bernardini, H. Chen, G. Fici, G. Loukides, and S. Pissis. Reverse-safe data structures for text indexing. In *ALENEX*, pages 199–213, 2020.
[9] L. Bonomi and L. Xiong. A two-phase algorithm for mining sequential patterns with differential privacy. In *CIKM*, pages 269–278, 2013.
[10] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4):181–186, 1962.
[11] M. Chen, X. Yu, and Y. Liu. Mining moving patterns for predicting next location. *Inf. Syst.*, 54(C):156–168, December 2015.
[12] R. Chen, G. Acs, and C. Castelluccia. Differentially private sequential data publication via variable-length N-grams. In *CCS*, pages 638–649, 2012.
[13] R. Chen, B. CM Fung, B. C Desai, and N. M Sossou. Differentially private transit data publication: a case study on the montreal transportation system. In *KDD*, pages 213–221, 2012.
[14] R. Chen, Y. Peng, B. Choi, J. Xu, and H. Hu. A private DNA motif finding algorithm. *Journal of biomedical informatics*, 50:122–132, 2014.
[15] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, 2014.
[16] G. Bernardini et al. Hide and mine in strings: Hardness and algorithms. In *ICDM*, pages 924–929, 2020.
[17] G. Bernardini et al. Combinatorial algorithms for string sanitization. *ACM Trans. Knowl. Discov. Data*, 15(1):8:1–8:34, 2021.
[18] G. Fici, F. Mignosi, A. Restivo, and M. Sciortino. Word assembly through minimal forbidden words. *Theoretical Computer Science*, 359(1):214–230, 2006.
[19] K. Ganesan and and E. Viegas C. Zhai. Micropinion generation: An unsupervised approach to generating ultra-concise summaries of opinions. In *WWW*, pages 869–878, 2012.
[20] F. Garcin, C. Dimitrakakis, and B. Faltings. Personalized news recommendation with context trees. In *ACM RecSys*, pages 105–112, 2013.
[21] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178, 2009.
[22] Y. Hong, J. Vaidya, H. Lu, and M. Wu. Differentially private search log sanitization with optimal output utility. In *EDBT*, pages 50–61, 2012.
[23] A. S. Iyer, J. S. Nath, and S. Sarawagi. Privacy-preserving class ratio estimation. In *KDD*, pages 925–934, 2016.
[24] Z. Jorgensen, T. Yu, and G. Cormode. Publishing attributed social graphs with formal privacy guarantees. In *SIGMOD*, pages 107–122, 2016.
[25] S. Jun, G.E. Sims, G. A. Wu, and S. Kim. Whole-proteome phylogeny of prokaryotes by feature frequency profiles: An alignment-free method with optimal feature resolution. *PNAS*, 107(1):133–138, 2010.
[26] P. Kairouz, S. Oh, and P. Viswanath. Extremal mechanisms for local differential privacy. In *NIPS*, pages 2879–2887, 2014.
[27] U. Keich and P. A. Pevzner. Finding motifs in the twilight zone. *Bioinformatics*, 18(10):1374–1381, 2002.
[28] J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.
[29] F. Liu. Generalized Gaussian mechanism for differential privacy. *TKDE*, 31(4):747–756, 2019.
[30] M. Murugesan, L. Si, C. Clifton, and W. Jiang. t-plausibility: Semantic preserving text sanitization. In *ICCSE*, volume 2, pages 68–75, 2009.
[31] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.
[32] P. A. Pevzner, H. Tang, and M. S. Waterman. An Eulerian path approach to DNA fragment assembly. *PNAS*, 98(17):9748–9753, 2001.
[33] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren. Generating synthetic decentralized social graphs with local differential privacy. In *CCS*, pages 425–438, 2017.
[34] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y Zhao. Sharing graphs using differentially private graph models. In *SIGCOMM IMC*, pages 81–98, 2011.
[35] O. K. Steinlein. Genes and mutations in idiopathic epilepsy. *American Journal of Medical Genetics*, 106:139–145, 2001.
[36] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu. Differentially private histogram publication. In *ICDE*, pages 32–43, 2012.
[37] S. Xu, X. Cheng, S. Su, K. Xiao, and L. Xiong. Differentially private frequent sequence mining. *TKDE*, 28(11):2910–2926, 2016.
[38] J. Zhang, X. Xiao, and X. Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *SIGMOD*, pages 155–170, 2016.