Received January 26, 2019, accepted February 8, 2019, date of publication February 27, 2019, date of current version March 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2899099

LDPart: Effective Location-Record Data Publication via Local Differential Privacy

XIANGGUO ZHAO[1], YANHUI LI[1], YE YUAN[1], XIN BI[2], and GUOREN WANG[3]

[1]College of Computer Science and Engineering, Northeastern University, Shenyang, China
[2]Sino-Dutch Biomedical and Information Engineering, Northeastern University, Shenyang, China
[3]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China

Corresponding author: Yanhui Li (lyhneu506822328@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672145, Grant 61572121, Grant 61702086, Grant 61622202, and Grant 61572119.

ABSTRACT Driven by the advance of positioning technology and the tremendous popularity of location-based services, location-record data have become unprecedentedly available. Publishing such data is of vital importance to the advancement of a wide spectrum of applications, such as marketing analysis, targeted advertising, and urban planning. However, the data collection may pose considerable threats to the individuals privacy. Local differential privacy (LDP) has recently emerged as a strong privacy standard for collecting sensitive information from users. Due to the inherent high dimensionality, it is particularly challenging to publish the location-record data under LDP. In this paper, we propose *LDPart*, a probabilistic top-down partitioning algorithm to effectively generate a sanitized location-record data. Our approach employs a carefully designed *partition tree* model to extract essential information in terms of location records. Furthermore, it also makes use of a novel adaptive user allocation scheme and a series of optimization techniques to improve the accuracy of the released data. The extensive experiments conducted on real-world datasets demonstrate that the proposed approach maintains high utility while providing privacy guarantees.

INDEX TERMS Big data privacy, local differential privacy, location-record publication.

I. INTRODUCTION

The location-record data refers to the data in which each record owner is associated with a set of locations drawn from a universe of locations. With the advancement of positioning technology and the popularity of location-based services [1]–[3], this type of data have become unprecedentedly available. The information hidden in such data, when properly defined and extracted, can greatly benefit tasks like marketing analysis, targeted advertising and trend monitoring. Despite the usefulness of location-record information, the massive data collection could incur significant privacy risks to both user contributors and the data collector, as demonstrated in several notable incidences (e.g., AOL,[1] and Netflix[2]), where unexpected leakage of sensitive data results in public outrage, reputation damage, and legal actions against the data collector (i.e., aggregator).

A promising methodology for addressing the privacy concerns in data collection is local differential privacy (LDP), which has been deployed in popular software systems such as Google Chrome browser [4], [5], Apple iOS [6], and Microsoft [7]. Its main idea is to guarantee that the aggregator (i) never gains access to the precise personal information of any individual, and yet (ii) would still be able to derive aggregates about the users. In particular, in LDP, each user locally perturbs his own data, and only the perturbed version is transmitted to the aggregator. As such, LDP protects not only the privacy of data contributors but also the collection itself against the risk of potential data leakage.

In the past few years, LDP has also drawn considerable interest from the data management community. However, as we review later in Section VI, existing solutions are either limited to the case where each user possesses a tuple of numeric or categorical value [4]–[6], [8], or focus mainly on basic statistics such as counts and mean. As a result, they are inadequate for more complex types of data mining tasks over high-dimensional location-record data.

The associate editor coordinating the review of this manuscript and approving it for publication was Xiangliang Zhang.

[1]https://en.wikipedia.org/wiki/AOL_search_data_leak
[2]https://www.wired.com/2009/12/netflix-privacy-lawsuit/

To our knowledge, only two studies [9], [10] support analysis on such data, but with severely restricted functionality (i.e., frequency estimation for single items [9], [10] and set cardinality estimation [10]). It is therefore urgent to develop a LDP-enabled data publication mechanism to meet various requirements of privacy-preserving location-record data analysis.

Due to the inherent high-dimensionality of location-record data, it is particularly challenging to apply LDP to location-record data. In particular, naively injecting noise to the count of each possible combination of locations (i.e., *location record*) negatively impacts both privacy and utility. First, unless we always enumerate all possible combinations (whether or not in the data), which is computationally infeasible even for moderately size universe. Second, the privacy budget must be split among deriving these counts, leading a tiny portion of the budget share assigned on each. As a consequence, the introduced heavy perturbation reduces released data utility considerably. In this paper, our proposed *LDPart* tackles this challenge through a well-established *partition tree* model to effectively generate a sanitized location-record data. With this model, it helps us to significantly reduce the space of location records. In fact, rather than exploring the entire universe of all possible distinct location records, we focus only on those potentially "non-zero". More precisely, LDPart estimates the count of *each generalized location record* (corresponding to an internal node) roughly, such as $\{R_2, R_3\}$ in Figure 2. This estimated count is accurate enough to determine whether or not to split the node. In this way, LDPart retains most of the users on estimating counts of actual common location records, i.e., leaf nodes in the partition tree. Finally, with these common location records, LDPart enables to accurately "reconstructing" the original data.

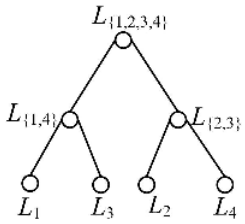
In summary, we make the following contributions:

- This is the first study that proposes an efficient sanitized approach scalable to high-dimensional location-record data under local differential privacy.
- We develop a series of techniques to guarantee good utility under the partition tree model, including an adaptive user allocation scheme, a formal choice of the count threshold and several optimized strategies.
- Through formal privacy analysis, we show that our proposed algorithms satisfy ϵ-LDP.
- We implement and evaluate LDPart on real-world datasets. Experimental results demonstrate the efficiency and effectiveness of our proposed data synthesis mechanisms.

The remainder of this paper is organized as follows. Section II presents some necessary preliminaries. Section III introduces the proposed privacy framework, whereas Section IV details the proposed solutions. The experimental evaluation is reported in Section V, followed by a survey of related work in Section VI. Finally, Section VII concludes this paper.

UID	Record
u_1	$\{L_1, L_4\}$
u_2	$\{L_4\}$
u_3	$\{L_3\}$
u_4	$\{L_1, L_2, L_4\}$
u_5	$\{L_1\}$
u_6	$\{L_4\}$
u_7	$\{L_1, L_4\}$
u_8	$\{L_1, L_4\}$

(a)



(b)

FIGURE 1. A *RT-tree* for the sampled location-record dataset.

TABLE 1. Summary of notations.

Symbol	Meaning
H/h	Random taxonomy tree and its height
$T/(l_{max}+1)$	Partition tree and its height
f	Fanout of H
$[N]$	User population
$v.cut$	Associated cut of node v
$c(v)/c'(v)$	True count of node v and its noisy count
$D/\widetilde{D}$	Original dataset, released dataset
X_i/X_i'	True value of i-th user u_i and his perturbed value
θ	Count threshold

II. MODEL AND PRELIMINARIES

A. PROBLEM STATEMENT

Let $L = \{L_1, L_2, \ldots, L_{|I|}\}$ be the universe of locations, where $|I|$ is the size of the universe. In our setting, each user u_i possesses a location record t_i which consists of a set of locations. Figure 1(a) presents a sample location-record dataset with $L = \{L_1, L_2, L_3, L_4\}$. To protect privacy, each user u_i perturbs his record t_i using a randomized mechanism. An untrusted aggregator (e.g., Google) is interested in gathering information on these perturbed data records from the population of users. In this paper, we aim to develop a solution for publishing a synthetic location-record data under LDP that has *similar property* with real data. Table 1 summarizes frequent notations used throughout the paper.

Problem Definition: Formally, let N be the total number of users and sufficiently large. Let $D = \{t_1, t_2, \ldots, t_N\}$ be the location-record dataset where each data record is owned by a user individually. Our goal is to release an approximate dataset $\widetilde{D}$ such that $S_t(D) \approx S_t(D')$, $\forall_{I' \subseteq L} Q_D(I') \approx Q_{D'}(I')$, where $S_t(D)$ ($S_t(D')$) be the set of distinct records in D (D'), and $Q_D(I')$ ($Q_{D'}(I')$) be the count of the location set I' in D (D'). Here, the reason for choosing counting property is that they are crucial to several data mining tasks, such as mining frequent patterns and association rules. In addition, the definition about Q is shown as follows.

Definition 1 (Counting Query): Given a set of location $I' \subseteq L$, a counting query Q over a user population $[N]$ is defined to be $Q(I') = |\{I' \subseteq t_i | u_i \in [N]\}|$.

In the classic privacy mechanisms [11], [12], the universe of an attribute is also accompanied by a *taxonomy tree*,

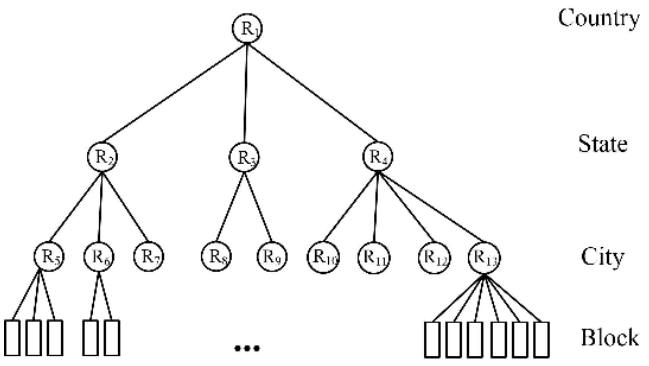


FIGURE 2. The administrative division of a country.

which indicates the publicly-known hierarchy among the possible values. To illustrate the concept, consider Figure 2, which demonstrates a simple taxonomy tree on attribute *Region* [12]. For such a tree, a lower level node provides more details than a higher level one, and an intermediate node summarizes its subtree. Broadly speaking, such internal node is often semantically consistent with the original items under its subtree. In our approach, these original items are independent of each other, and thus the taxonomy tree could be random.

Definition 2 (Random Taxonomy Tree, RT-Tree*):* Given the universe of locations $L = \{L_1, L_2, \ldots, L_{|I|}\}$ and a fan-out f, a random taxonomy tree H is built by hierarchically merged randomized location space into h levels, where $h = \lceil \log_f |I| \rceil + 1$ and its internal nodes are a set of their leaves.

Figure 1(b) illustrates an example *RT-tree* for the sampled dataset in Figure 1(a). As observed, there are 3 internal nodes $L_{\{1,4\}} = \{L_1, L_4\}$, $L_{\{2,3\}} = \{L_2, L_3\}$, and $L_{\{1,2,3,4\}} = \{L_1, L_2, L_3, L_4\}$. For a *RT-tree* H, the total number of internal nodes of its subtree rooted at r_j is donated as $ItNode(r_j)$. It is quite obvious that this quantity indicates the number of split operation needed to completely reach leaf nodes. Especially, we refer the number of internal nodes at level i for this subtree as $ItNode_i(r_j)$. In addition, the total number of internal nodes (or internal nodes at level i) of H are represented as $ItNode(H)$ and $Level_i(H)$, respectively. For a node r_j, the height $h(r_j)$ is defined as its level in H. Note that the leaves have level 0 and the root has level h. Further, if a location is in the set of a RT-tree node, then we say that this location can be *generalized* to the node. For instance, since $L_4 \in \{L_1, L_4\}$, and then L_4 can be generalized to $L_{\{1,4\}}$.

B. LOCAL DIFFERENTIAL PRIVACY

Initial work on differential privacy (DP) mainly focuses on central differential privacy [13]–[15]. In the centralized model, it assumes the participation of a trusted aggregator who has access to the private data of individuals and releases information through a DP algorithm. In practice, individuals may be reluctant to share private information with the central data curator. Instead, local differential privacy (LDP) captures the case where the data contributors trust no one else but themselves.

In the local setting, the private data of each user is locally randomized before being sent from the device, so the aggregator never collects or possesses the exact raw data. Intuitively, LDP requires that no matter which data record a user has, the aggregator should learn almost the same information. In other words, by seeing the perturbed information, an adversary with arbitrary background knowledge cannot distinguish the user's original data record with another one with high confidence. In this sense, LDP provides plausible deniability to the user. Formally, LDP is defined as follow. Here the parameter, ϵ, specifies the degree of privacy offered. A smaller ϵ leads to stronger privacy protection, and vice versa.

Definition 3 (Local Differential Privacy): A randomized algorithm $\mathcal{A}$ is ϵ-LDP if for any two location records t and t', and for any possible output $O \subseteq Range(\mathcal{A})$,

$$Pr(\mathcal{A}(t) \in O) \leq e^{\epsilon} Pr(\mathcal{A}(t') \in O) \tag{1}$$

Theorem 1 (Sequential Composition [16]): Let $\mathcal{A}_i, \ldots, \mathcal{A}_m$ be m algorithms, each provides ϵ_i-LDP. A sequential of algorithms $\mathcal{A}_i(t)$ provides $(\sum_i \epsilon_i)$-LDP.

1) FUNDAMENTAL PRIVATE MECHANISMS

We describe primitives for LDP on simple inputs, which serve as the basic building blocks for our protocols.

1 bit Randomized Response: A fundamental mechanism for achieving ϵ-LDP is Randomized Response (RR) [17], a widely used surveying technique for collecting statistics on sensitive topics where survey respondents wish to retain confidentiality. More specifically, in its simplest form, one reports the true value X_i (i.e., either 0 or 1) with probability p and the flip $(1 - X_i)$ of the true value with probability 1-p. This satisfies $(ln\frac{p}{1-p})$-LDP.

Optimized Parallel Randomized Response: In general, we often encounter cases where each user holds a sparse vector of length m with 1 at exactly one position and 0's at remaining places. In these cases, accuracy could be improved by an alternative approach which applies m instances of 1 bit RR, each with parameter $\epsilon/2$. We called this approach as Parallel Randomized Response (also known as Unary Encoding [18] and Basic RAPPOR [4]). Recently, Wang *et al.* [18] slightly improve this mechanism by setting the probability of retaining the sole 1 to be $p = \frac{1}{2}$, and the probability of flipping each 0 to be $q = (1 + e^{\epsilon})^{-1}$. Then, the aggregator computes $\frac{\sum_{i=1}^{n} X_i' - nq}{p-q}$, as an estimate for the target value of each position, i.e., $\sum_{i=1}^{n} X_i$. We refer to the improved mechanism as Optimized PRR. Note that the output of optimized PRR at each position is binary. Thus, it is sufficient for each user to transmit only 1 bit to the aggregator, which minimizes communication overhead.

III. OVERVIEW

Intuitively, a locally differentially private publication of a location-record dataset could be generated by adding noise

to the counts of a set of all possible location records. However, this would be infeasible since it is computationally prohibitive to enumerate all possible location records in any non-trivial sized universe. In addition, the privacy budget must be split among deriving these counts, leading a small portion of the budget assigned on each. As a consequence, it leads to inaccurate estimate of count values, and thus the utility of the released data is drastically reduced.

We first provide an overview of the proposed approaches. Given the analysis above, the key challenge of publishing location-record dataset is the unknown space of location records. We do not have any prior knowledge about the location records that actually appear in D. To identify the set of such high-quality location records, we propose a well-designed tree structure, called *partition tree* (*PTree*, for short). In particular, it starts by creating a single partition with a unified representation. This unified representation, also named as *hierarchy cut*, is composed of a set of *RT-tree* nodes. Subsequently, it iteratively splits the partition into disjoint sub-partitions with more detailed representations in a top-down manner based on the *RT-tree*. During this process, for each sub-partition, it requires to determine whether it is ''empty'' in a noisy way and further split the ''non-empty'' sub-partitions. The entire partition process terminates when no sub-partition can be further expanded. Finally, we make use of the noisy count of *each leaf cut* to construct the release. A leaf cut is a partition that each node in its hierarchy cut is a leaf node of the corresponding *RT-tree*, for example, the associated cuts of the nodes v_4, v_5 and v_6 in Figure 3 are leaf cuts.

IV. SANITIZATION ALGORITHMS

A. BASIC METHOD

Before presenting our main approaches UniPart and LDPart for publishing location-record data under LDP, we first describe a simple solution called UniBSL based on an LDP-compliant PTree. On the whole, these approaches follow the unified framework as stated in previous section, but differ in counting strategies at the aggregator. Given the hierarchy cut of a node v, its count $c(v)$ is defined as the number of location records that *match* it. Concretely, a user's location record is said to match a cut iff (i) each location in his record could be generalized to a node in the cut; and (ii) every node in the cut generalizes some locations in the record. For instance, the record $\{L_1, L_4\}$ of u_1 matches to the hierarchy cuts $\{L_1, L_4\}$, $\{L_{\{1,4\}}\}$ and $\{L_{\{1,2,3,4\}}\}$, but not $\{L_{\{1,4\}}, \{L_{\{2,3\}}\}\}$.

In UniBSL, the aggregator builds a PTree T under ϵ_1-LDP, and adds each leaf cut terms in T to the sanitized dataset $\widetilde{D}$. Algorithm 1 presents our UniBSL approach in more detail. Specifically, it starts by performing a series of initialization operations (Lines 1-4), including creating a root node, setting the noisy count, splitting root's cut, and adding new generating nodes into queue. Then, in Lines 5-12, the algorithm iteratively constructs each level of T. Given a node v, the aggregator collects data under LDP to compute an estimate $c'(v)$ of $c(v)$, the true count for

Algorithm 1 UniBSL_Build_PTree(ϵ_1, l_{max}, H)

```
1  Create a PTree T with a root node v_r;
2  Initialize v_r.cut ← the root of H, c'(v_r) = N;
3  Sub-partitions V_p ← SubPart_Gen(v_r.cut, H);
4  Add V_p to an initially empty queue Q;
5  while Q ≠ φ do
6      Dequeue v from Q;
7      Apply optimized PRR to collect information from
       all users to estimate c'(v) of v.cut using privacy
       budget ε1 / l_max;
8      if v.cut is a leaf cut then
9          Add c'(v) copies of v.cut to D̃;
10     else if c'(v) ≥ θ then
11         Sub-partitions V'_p ← SubPart_Gen(v.cut, H);
12         Add V'_p to Q;
13 Return D̃;
```

v's corresponding cut. After that, it checks if v's corresponding cut is a leaf cut. If so, $c'(v)$ copies of v's cut are added to $\widetilde{D}$; otherwise, we decide whether to split the node v by comparing its estimated count $c'(v)$ with a count threshold θ. If $c'(v) \geq \theta$, v is expanded by explicitly considering every possible sub-partitions as a child of v. For a non-leaf cut whose $c'(v) < \theta$, we treat it as ''empty'' and do not consider its child nodes further. The count threshold θ is a pre-defined threshold that will be discussed later.

For a ''non-empty'' non-leaf partition v, the function SubPart_Gen($v.cut, H$) is used to generate a candidate set of sub-partitions. In particular, it randomly selects a node r from $v.cut$ to expand, and then exhaustively generates the sub-partitions by replacing r by the combinations of its children. If the number of r's children in H is n_l ($n_l \leq f$), then the total number of generated sub-partitions is $2^{n_l} - 1$. Take the node v_3 in Figure 3 as an example. Through expanding $\{L_{\{2,3\}}\}$, it generates three sub-partitions, i.e., v_7, v_8, and v_9. Besides, to derive the count of each cut privately in UniBSL, the privacy budget uniformly assigned on each level is $\frac{\epsilon_1}{l_{max}}$, which largely relies on l_{max}. Below, we provide insights for the aggregator to select a proper l_{max} value in practice.

One possible setting for l_{max} is $l_1 = ItNode(H)$. The rationale is that the maximum number of partition operations needed to reach leaf cut for T is $ItNode(H)$. Unfortunately, it tends to be rather large especially when the size of L is large or f is small. Nevertheless, in reality, the majority of the records in the data are short and only few of them are very long [19], [20]. In this scenario, many (or even all) root-to-leaf paths have a length much shorter than l_1. Hence, setting l_{max} solely based on l_1 is clearly not optimal. According to the construction of H, most of internal nodes in H are at level 1. If we could reduce the number of involved such nodes, the number of partition operations would be significantly reduced. Inspired by this fact, an alternative choice is to make

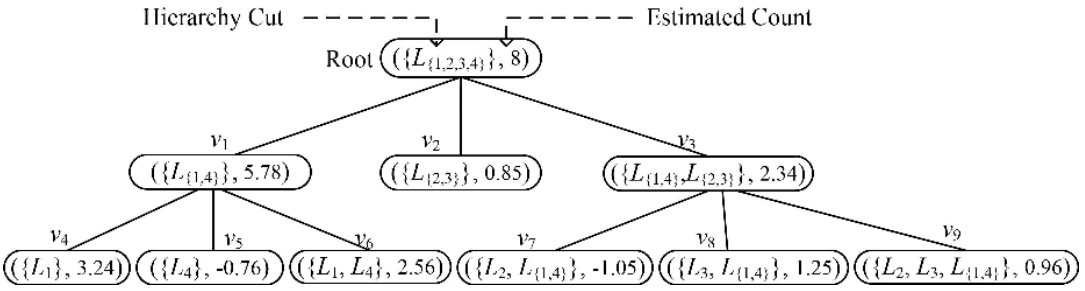


FIGURE 3. The partition tree of the sample data.

better use of statistic information subtracted from the dataset. For a record t_i of length l, the amount of nodes that locate in at level 1 of H in its matched cut is at most l. In other words, the nodes in its matched cut at level 1 could be bound by $min\{Level_1(H), l\}$. For example, the nodes in the matched cut at level 1 are no more than one for u_2's record in Figure 1. Similarly, the nodes in its matched cut at any level i also be bound by $min\{Level_i(H), l\}$. From the analysis above, we can conclude that the upper bound of internal nodes in a cut (i.e., l_{max}) is $\sum_i^h min\{Level_i(H), l_m\}$, where l_m is the maximum length of user records.

Theorem 2: Our UniBSL algorithm satisfies ϵ-LDP.

Proof (Sketch): Let $Level_k(T)$ $(0 \leq k \leq l_{max})$ be the set of nodes on the k-th level in T. Apparently, the associated cuts of these nodes are different. For a user u_i, his record can only match one of them. It means that the true vector of u_i is a sparse vector: exactly one entry is 1, and the rest are 0. Based on optimized PRR [18], the process of collecting such perturbed binary vector is $\frac{\epsilon_1}{l_{max}}$-LDP if the $\frac{\epsilon}{l_{max}}$ privacy budget is assigned. The splitting process of T invokes PRR l_{max} times, one for each level of T, and thus it satisfies $\frac{\epsilon_1}{l_{max}} \times l_{max} = \epsilon_1$-LDP (Theorem 1). In addition, the budget ϵ_2 is assigned to determine l_m, which described in subsection IV-C. As evident, this procedure satisfies ϵ_2-LDP. The sequential composition property guarantees that the whole process satisfies $\epsilon_1 + \epsilon_2 = \epsilon$-LDP.

Algorithm 2 UniPart(ϵ, H)

```
Same as Lines 1-4 in Algorithm 1;
Set U_{v_r} = {u_1, u_2, ..., u_N};
while Q ≠ φ do
    Dequeue v from Q;
    Calculate available user set U_v;
    if v.cut is a leaf cut then
        Apply optimized PRR to collect information
        from U_v to estimate c'(v) using ε;
        Add c'(v) copies of v.cut to D̃;
    else
        Estimate h_v = Σ_{r_j ∈ v.cut} |ItNode(r_j)| + 1, and
        l_v = |U_v| / h_v;
        while U_v ≠ φ do
            Randomly sample a set of users U'_v of
            length l_v;
            Apply optimized PRR to collect information
            from U'_v using ε and incrementally refine
            the estimated count c'(v);
            Update θ, and Remove all users in U'_v from
            U_v;
            if c'(v) ≥ θ and U_v ≠ φ then
                Repeat Lines 11-12 in Algorithm 1;
                Break;
Return D̃;
```

B. LDPART

In the design of LDP protocols, one important principle is "one should divide the user population, instead of dividing the privacy budget" [21]. Inspired by this, a straightforward approach is to randomly divide the user population into l_{max} equal-sized groups, one for each level of the PTree except the root node. However, this naive extension suffers from two serious drawbacks as UniBSL. First, it requires to impose a limit l_{max} on the recursion depth in the splitting of T. Second and more importantly, each run of optimized PRR is allocated a rather small portion of the user population, which dominates overall result inaccuracy.

To address these drawbacks, we propose the UniPart algorithm that utilizes an *adaptive user allocation scheme* but completely eliminates the dependency on a pre-defined l_{max}. The first idea behind it is that *it is not necessary to allocate each user to a single level of the PTree.* Actually, if there are no node is an ancestor of another in a set of nodes V_i, then a user u_i could participate in the count estimation process for all nodes in V_i, each with the entire privacy budget ϵ. The correctness of this conclusion directly follows the property of optimized PRR, described in Section II-B. Therefore, the available user set U_v for a node v in the PTree is the set of users who have not participated in the count estimation for any of v's ancestor nodes. For instance, in Figure 3, U_{v_7} is comprised of users who have not contributed to data collection related to v_3. From this perspective, *the user sets participated in count estimation for nodes in a path should be disjoint, and different paths could share the whole user population.*

Based on above analysis, the second idea is that *a desirable user allocation scheme should take the length of a root-to-leaf path into account*. Generally, each node in a shorter path should receive more users; In contrast, each node in a longer path should use less users. Hence, we independently estimate the maximum length of each path and then distribute the remaining users as per the estimated length. To be specific, given a node v of a path and an associated *RT-tree* H, the maximum length of the path rooted at v is estimated as $(\sum_{r_j \in v.cut} ItNode(r_j) + 1)$.

Obviously, the node at a higher level in a PTree T has a larger real count than its descendant nodes. Once such a node is eventually pruned due to inadequate count, its descendants can be omitted from further consideration, and thus we could exploit all its available users to estimate the count. On the other hand, more users can contribute to more accurate estimation [21]. These facts motivate the third idea of our adaptive user allocation scheme: *incrementally check whether the estimated count of a node pass θ, rather than only one time test*. Towards this end, the detailed procedure of our proposed UniPart is shown in Algorithm 2. Furthermore, the following theorem establishes the theoretical guarantee of Algorithm 2.

Theorem 3: UniPart satisfies ϵ-LDP.

Proof (Sketch): For the user u_i, there are no node is an ancestor of another in the set of its participated nodes V_i. In other words, these nodes are *disjoint*. It can be easily verified that, the data collection for the entire V_i by applying optimized PRR with ϵ budget guarantees ϵ-LDP. In the same manner, the data collection for other users also ensure ϵ-LDP. In summary, the whole process satisfies ϵ-LDP.

1) IMPROVED APPROACH: LDPart

Compared to UniBSL, UniPart achieves adaptive estimation of node counts, and eliminates the dependence on predefined l_{max}. Overall, the accuracy of released location-record dataset is increased, as validated in Section V. Recall that the final leaf cuts are used to construct the released data in our framework. For this reason, if the counts of these leaf cuts are more accurate, then the utility of released result would be significantly improved. Hence, we propose a more sophisticated adaptive user allocation scheme. In particular, we reserve half of users to obtain the noisy counts of leaf cuts, which are used to construct the release, and use the remaining users to guide the splitting process. This improved approach is named as *LDPart*. In a nutshell, LDPart is similar to UniPart in that it also (i) generates T by iteratively splitting a root node v_r whose cut covers the whole universe of locations, and (ii) incrementally decides whether a node v should be split based on its estimated count. However, the method for obtaining noisy counts of leaf cuts marks the crucial difference between the two approaches.

2) OPTIMIZATIONS

To further improve the performance of LDPart, in the following, we present two-folded optimization strategies.

Non-Negative Counts (NNT): Once the PTree T is obtained, all leaf cuts are used to reconstruct the sanitized set-valued dataset $\widetilde{D}$. In order to guarantee user privacy, the uncertainty is injected to the true counts of these cuts. In a non-perturbed PTree T, the counts of all nodes must be non-negative. However, this constraint may be violated in the noisy tree created by LDPart due to the noisy injected to each node. Therefore, this reconstruction process can further be improved by publishing $c'(v)$ copies of $v.cut$ only if the estimated count of a leaf cut does not result in negative value.

Smart Estimation (SE): In both UniPart and LDPart, our proposed user allocation scheme depends on the estimated length of each path, which heavily affected by the structure of *RT-tree* H. Similar to UniBSL, the estimated value may tend to be large. Consequently, less users are assigned to each node, and multiple rounds of check could be required, reducing the efficiency of the algorithms. To tackle this issue, we present a more smarter path length estimation strategy. Intuitively, for the records less than l_m in length, the number of nodes at level i of H in its matched cut is at most l_m. More precisely, if some nodes of the matched cut already are single items, the number of nodes at level i could further bound by $(l_m - n_s)$, where n_s is the number of single items. Thus, given a node v and its cut $v.cut$, the maximum length of this path could be calculated as $h_v = \sum_{i=1}^{h_p} \min\{\sum_{r_j \in v.cut} ItNode_i(r_j), l_m - n_s\} + 1$, where $h_p = \max_{r_j \in v.cut} h(r_j)$. With the help of smart estimation, LDPart would estimate each path more accurately.

Theorem 4: LDPart satisfies ϵ-LDP.

Proof (Sketch): In LDPart, half of users involve in building PTree, and the other half of users are used to estimate the counts of leaf nodes V_l. The former satisfies ϵ-LDP, as shown in theorem 3. Since the nodes in V_l are disjoint, the latter could use optimized PRR to participate the count estimation of all nodes in V_l. Therefore, for the latter, it also satisfies ϵ-LDP. The optimization strategy NNT is conducted on noisy data, rather than the original data, and thus it does not violate the privacy. Analogously, the SE strategy does not access the original data as well, and does not have the effect on the privacy of LDPart. Therefore, our LDPart as a whole gives ϵ-LDP.

C. PARAMETER SETTING

1) COMPUTING THRESHOLD θ

In the construction of T, a node (partition) is not further split if its estimated count is lower than the count threshold θ. The main source of error in T comes from the nodes that are of a true count of zero (i.e., empty node) but of an estimated count greater than θ. We refer such nodes as *false nodes*. In order to reduce the magnitude of noise in T, we propose a count threshold to limit the total number of false nodes.

Basically, a false node v will generate, on average, $m_f P_\theta$ false children nodes in each expansion, where $m_f = 2^f - 1$, and the parameter P_θ donates the probability of perturbation noise passing θ. This is because any descendant of an empty

TABLE 2. Real dataset parameters.

Name of Dataset	Records	Items	Max Length	Average Length	Estimated l_m
MSNBC	989818	17	17	1.72	3
T10I4D100K	100000	870	29	10.1	15

node must have an actual count of zero. For the expansion of T, the total number of false nodes accumulates exponentially with the factor of $m_f P_\theta$, leading to excessive noise. To limit the explosive growth of false nodes in this process, we set $m_f P_\theta \leq 1$, i.e., $P_\theta \leq \frac{1}{m_f}$.

Suppose that there are n_v users participating in the estimation of a node v's count. Since v is an empty node, i.e., $X_i = 0 (i \in [1, n_v])$. It means that the record of any user cannot match to $v.cut$. Under LDP setting, we obtain $I_v = \sum_{i=1}^{n_v} X_i'$, the number of $X_i' = 1$ is reported in noisy values, and then compute $E_v = \frac{I_v - n_v q}{p-q}$. On the other hand, I_v can be seen as the summation of n_v binomial variables $(X_i' - X_i)$, whose probability of being 1 is q. Central limit theorem shows that, in most cases (as long as n_v is large), I_v can be approximated by the normal distribution with mean $\mu = n_v q$ and variance $\sigma^2 = n_v q(1-q)$. Since both $n_v q$ and $p - q$ are constant, the variable E_v also is a normal random variable but with different $\mu = 0$ and $\sigma^2 = \frac{n_v q(1-q)}{(p-q)^2}$.

For any false node v, the probability of the estimated total count above θ is $Pr(c'(v) = \frac{N}{n_v} E_v \geq \theta)$. Given that $Pr(c'(v) \geq \theta) \leq \frac{1}{m_f}$, we have

$$
\begin{aligned}
Pr(\frac{N}{n_v} E_v \geq \theta) \leq \frac{1}{m_f} &\Rightarrow Pr(E_v \geq \frac{n_v}{N}\theta) \leq \frac{1}{m_f} \\
&\Rightarrow Pr(E_v \leq \frac{n_v}{N}\theta) \geq 1 - \frac{1}{m_f} \\
&\Rightarrow \Phi(\frac{\frac{n_v}{N}\theta(p-q)}{\sqrt{n_v q(1-q)}}) \geq 1 - \frac{1}{m_f} \\
&\Rightarrow \Phi(-\frac{\frac{n_v}{N}\theta(p-q)}{\sqrt{n_v q(1-q)}}) \leq \frac{1}{m_f}.
\end{aligned}
$$

Solving, we derive that $\theta = \frac{N\Phi^{-1}(\frac{1}{m_f})\sqrt{n_v q(1-q)}}{n_v(q-p)}$, where Φ^{-1} is the inverse of cumulative density function. Since the parameters q and n_v are distinguished in the proposed approaches, the specific setting of θ is formulated in (2).

$$
\theta = \begin{cases} \dfrac{\Phi^{-1}(\frac{1}{m_f})\sqrt{Ne^{\frac{\epsilon}{l_{max}}}}}{1 - p(e^{\frac{\epsilon}{l_{max}}} + 1)} & \text{UniBSL} \\ \dfrac{N\Phi^{-1}(\frac{1}{m_f})\sqrt{e^{\epsilon}}}{\sqrt{n_v}(1 - p(e^{\epsilon} + 1))} & \text{UniPart and LDPart} \end{cases} \tag{2}
$$

2) CHOOSING l_m

In our proposed approaches, the parameter l_m plays an important role in the final utility. As described above, it is used to estimate the maximum length of each path in T. Clearly, the choice of l_m affects the privacy budget (or the user) assigned to each level of T, and therefore, is also related to the magnitude of noise. Unfortunately, it is sensitive and there is no systematic way to set this parameter. Here, we provide a heuristic method to determine a suitable value for l_m. More specifically, let l_τ is the upper bound of the length of the user's record, which is a public-known value. For each user who participated in the estimation of l_m, he perturbs his sparse vector of length l_τ by applying optimized PRR and sends the perturbed version to the aggregator. After that, the aggregator computes the estimated value of α_i, the number of records in length i. Starting from $l_m = 1$, we incrementally compute the percentage $p = (\sum_{i=1}^{l_m} \alpha_i)/n$ until p is at least η.

V. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the performance of our proposed algorithms. All programs are implemented on a machine with Inter(R) Core(TM)i7-2600 CPU 3.40GHz, using C++.

A. EXPERIMENTAL SETUP

1) REAL DATASETS

Our proposed approaches are not restricted to location-record data, but could applied to more general set-valued data [22], where each record owner is associated with a set of items drawn from a universe of items. In the experiments, we use two publicly available real datasets. The first is MSNBC,[3] which describes sequences of web pages browsed by users in time order. By ignoring the sequentiality and removing duplication, this dataset is converted into location-record data. The second dataset we used is T10I4D100K,[4] which was generated using the generator from the IBM Almaden Quest research group. Choosing these two datasets, we intend to evaluate the performance of publishing approaches for set-valued data at varying scales. A summary of these two datasets is reported in Table 2.

2) COMPETITORS

In the experiments, we implement three algorithms: UniBSL, UniPart and LDPart for publishing set-valued data. Other competitors are either on low-dimension data [4], [8], [23] or for severely restricted functionality [9], [10] and therefore not comparable.

3) PARAMETER CONFIGURATIONS

Regarding privacy parameters, in UniBSL, we allocate $\epsilon_1 = 0.9\epsilon$ for constructing the PTree, and remaining $\epsilon_2 = 0.1\epsilon$ for determining the maximum record length l_m.

[3] http://archive.ics.uci.edu/ml/datasets
[4] http://fimi.ua.ac.be/data

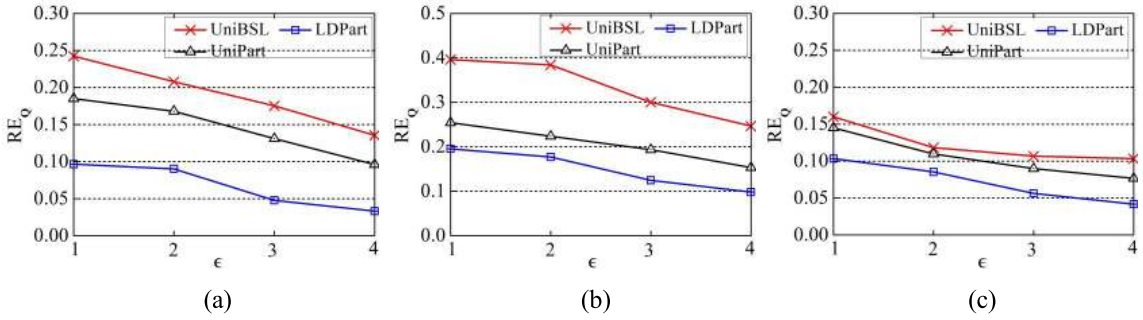


FIGURE 4. Accuracy of counting queries on MSNBC. (a) $k = 1$. (b) $k = 2$. (c) $k = 3$.

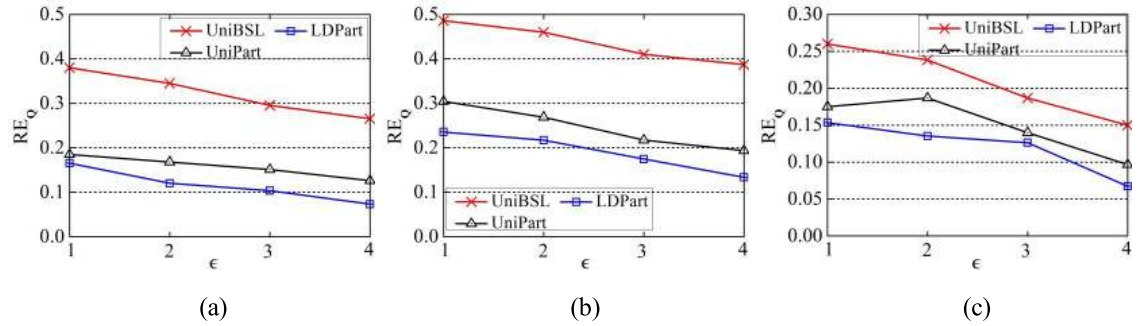


FIGURE 5. Accuracy of counting queries on T10I4D100K. (a) $k \in [1,5]$. (b) $k \in [6,10]$. (c) $k \in [11,15]$.

Similarly, in UniPart and LDPart, for the same purposes we use two sets of users with cardinality $n_1 = 0.9N$, and $n_2 = 0.1N$. The parameter η in our experiments is set to 0.9. As these algorithms involve randomization, we run each algorithm ten times and report its average results.

B. EXPERIMENTAL RESULTS

1) ACCURACY OF COUNTING QUERY

In line with previous works [24], [25], we measure the utility of the sanitized data in terms of counting query. In particular, *relative error* (*RE*) is used to measure the error with respect to the actual counts over the raw dataset D. It is defined as $\frac{|Q_D(I')-Q_{D'}(I')|}{\max\{Q_D(I'),0.5\%\cdot|D|\}}$, where $0.5\% \cdot |D|$ is a sanity bound that mitigates the influence of the queries with extremely small *selectivities* [24], [26]. A smaller RE indicates that the noisy counts returned by a technique are closer to the actual counts.

In our first set of experiments, we take a close examination of the impact of the privacy budget ϵ over the RE of counting queries. For both dataset, we randomly generate 2000 counting queries with varying length k ($k \in [1, l_m]$), and the items of each query are randomly drawn from the universe. Note that, the number of items in each query is referred as its length.

Figures 4 and 5 present the RE under varying privacy budget ϵ from 1 to 4 with fan-out $f = 4$ for each query set. From these figures, we can see that LDPart consistently outperforms the other two at the same level of privacy in terms of RE. The good accuracy of LDPart is mainly due to the effective adaptive user allocation scheme. Meanwhile, it is observed that both UniPart and LDPart perform better than UniBSL in most cases. This validates that allocating user is more effective to improve accuracy than splitting budget. Furthermore, in both datasets, increasing ϵ has beneficial impact on the RE. It conforms to the theoretical analysis that a larger ϵ makes the probability of retaining original "0" (i.e.,$1-q$) higher, resulting in less noise and therefore a more accurate result.

2) UTILITY OF RELEASED RECORDS

The second set of experiments are designed to evaluate the performance of the released records of various publishing models. In particular, we are interested in two critical measures: *F-score* and *RE*. F-score is a combination of *precision* and *recall* for the released records. In the same manner, RE is used to measure count errors of these records. Compared to the counting query, the true count of each real record is rather small, and the *selectivity* here is set to $0.05\% \cdot |D|$. In addition, to make a distinction between the RE of counting query and released record, we employ RE_Q and RE_R to donate them respectively.

The result is illustrated in Figure 6. As shown, LDPart achieves good utility in terms of released records. Moreover, we also investigate the precision of these records. As expected, LDPart achieves rather high precision, very close to 1. In contrast, it could not obtain comparable performance in terms of Fscore. This is reasonable. One main

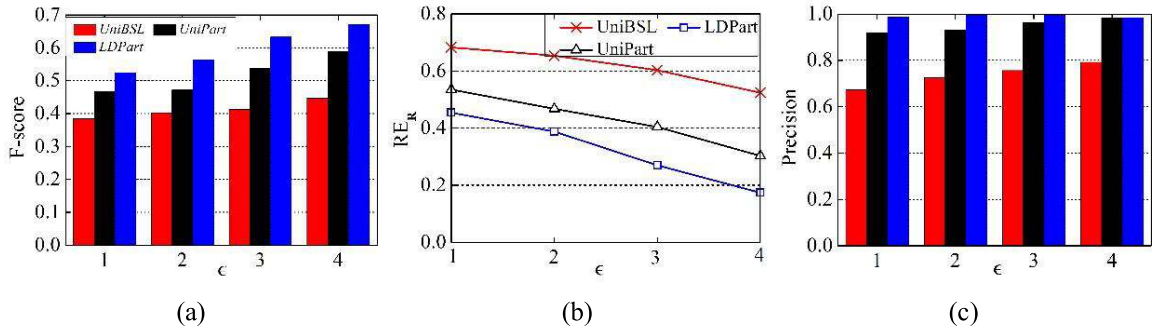


FIGURE 6. Utility of released records on MSNBC.

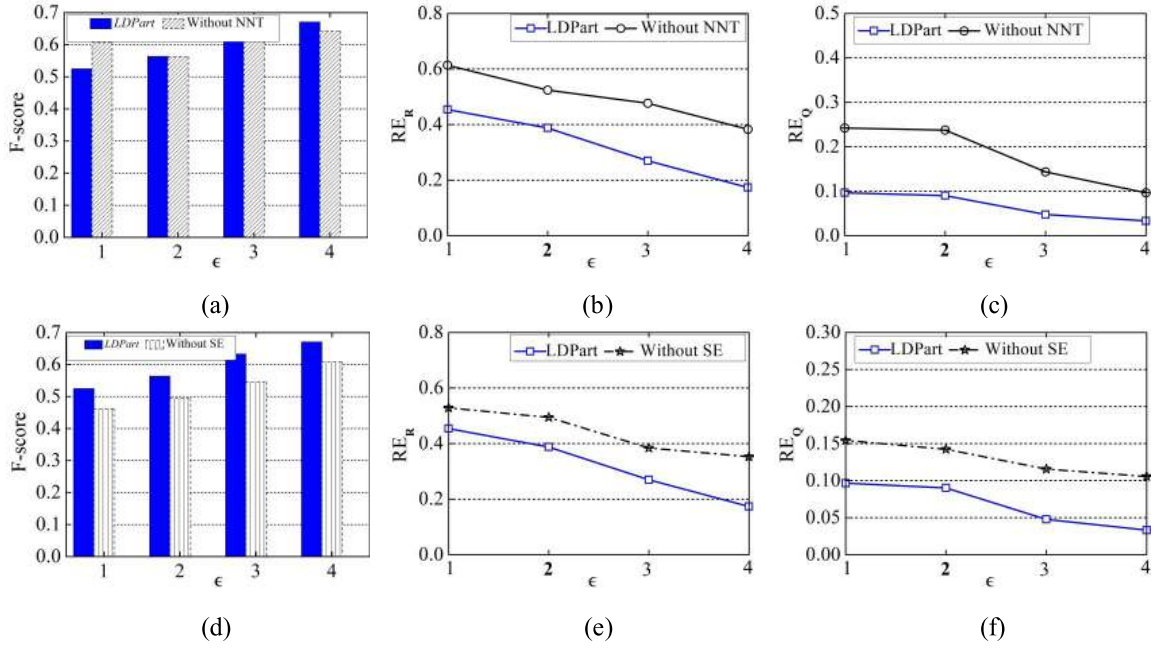


FIGURE 7. Effect of optimized strategies.

reason for this phenomenon is that the perturbed mechanisms of these models introduce noise, which is inevitable. The injected noise is more likely to dominate the noisy counts for the records with small real counts, leading to being pruned with high certainty. Consequently, the recall is lower and the performance is deteriorated in terms of Fscore. The experiment results and finding are similar on two real datasets, we omit the results of T10I4D100K.

3) EFFECT OF OPTIMIZED STRATEGIES

In the final set of experiments, we study the effect of optimized strategies on the performance of LDPart. As a reference point, we compare LDPart with its two variants: one is without non-negative counts strategy, the other is without smart estimation strategy. To measure RE of counting queries, we take $k = 1$ as an example. From Figures 7(d)-7(f), we can see that our proposed smart estimation substantially improves the performance of released data. With regard to non-negative counts shown in Figures 7(a)-7(c), it is interesting that the RE of both counting queries and released records are improved, but not Fscore. This is because, a noisier set of released records causes lower precision, but may result in higher recall. Therefore, the non-negative counts strategy has slightly impact on Fscore.

VI. RELATED WORK

Location-record data has been extensively studied in the past few decades [27], [28]. Essentially, it is a special case of set-valued data. Our work relates to two main streams of research, concerning publishing set-valued data and local differential privacy, respectively.

A. SET-VALUED DATA PUBLISHING

Initial efforts to ensure privacy of released set-valued data were based on partition-based privacy models [29] such as k-anonymity [11] (or its variant, k^m-anonymity [22])

and/or confidence bounding [30]. These techniques have been shown to be effective in several scenarios, however, due to the deterministic nature and the vulnerability to adversaries' background knowledge, they suffer from many types of privacy attacks [31], [32], leading to privacy compromise. Compared with the above models, centralized differential privacy [13], [14], [33] provides a formal and more stronger privacy guarantee, regardless of adversaries's background and ability. Under this model, only two techniques have been proposed to tackle the problem of privacy-preserving set-valued data publication. Chen *et al.* In [24] develop a probabilistic top-down partitioning algorithm to generate a differentially private release. Subsequently, Zhang *et al.* [34] propose an algorithm which leverages an item-free taxonomy tree and an update bounded mechanism to privately publish data on an incremental scenario.

B. DIFFERENTIAL PRIVACY IN LOCAL SETTING

The schemes mentioned above typically assumes that a data curator collects exact records from all users with the goal of releasing a modified version of the data that preserves privacy. Nevertheless, in reality, individuals may not trust any one but themselves. The recently proposed notion of LDP captures this case. Existing techniques for LDP have focused on tasks rather different from publishing set-valued data. In particular, one basic task in LDP is *frequency estimation* which estimates the frequency of each possible value. It has been well studied in [8] and [35]–[37]. When the domain of possible values is very large, it is infeasible to derive estimations for all values. A series of works [21], [23], [38] study the extension of frequency estimation, that is, the heavy hitter problem. Instead, this problem aims to identify *frequent* values. In addition, some efforts are also devoted to explore a variety of perturbation mechanisms for achieving theoretical optimal utility [8], [35], [39], [40].

The above research almost all assume that each user is only in possession of single data element. Due to the inherent high-dimensionality of set-valued data, these approaches are not applicable. To our knowledge, only two studies [9], [10] support the analysis on set-valued data. In particular, Qin *et al.* [9] propose a heavy hitter estimation over set-valued data based on two-round user-server interactions. Meanwhile, it is only limited to single items, and is not sufficient for more complex analysis tasks. Wang *et al.* [10] propose *PrivSet* for frequency estimation of single items and set cardinality estimation, and thus it suffers from the drawback as the work [9] as well.

VII. CONCLUSIONS

In this paper, we investigate the problem of publishing location-record data under local differential privacy. To address this issue, we present LDPart, a probabilistic top-down partitioning algorithm. It mainly through an adaptive user allocation scheme for building a partition tree under LDP, which clearly outperforms baseline solutions with simple partition tree construction approaches. In the future, we intend to explore frequent itemsets mining over set-valued data and further improve the accuracy of released data.

REFERENCES

[1] S. Shang, L. Chen, C. S. Jensen, J. R. Wen, and P. Kalnis, "Searching trajectories by regions of interest," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 7, pp. 1549–1562, Jul. 2017.
[2] Y. Li, Y. Yuan, G. Wang, L. Chen, and J. Li, "Semantic-aware location privacy preservation on road networks," in *Proc. DASFAA*, 2016, pp. 314–331.
[3] S. Shang, L. Chen, Z. Wei, C. S. Jensen, J.-R. Wen, and P. Kalnis, "Collective travel planning in spatial networks," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1132–1146, May 2016.
[4] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacypreserving ordinal response," in *Proc. CCS*, 2014, pp. 1054–1067.
[5] G. Fanti, V. Pihur, and Ú. Erlingsson, "Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries," *Proc. Privacy Enhancing Technol.*, vol. 2016, no. 3, pp. 41–61, 2016.
[6] A. Greenberg, "Apple's 'differential privacy' is about collecting your data—But not? Your data," *Wired*, 2016.
[7] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Proc. NIPS*, 2017, pp. 3571–3580.
[8] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *Proc. Theory Comput.*, 2015, pp. 127–135.
[9] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, "Heavy hitter estimation over set-valued data with local differential privacy," in *Proc. CCS*, 2016, pp. 192–203.
[10] S. Wang, L. Huang, Y. Nie, P. Wang, H. Xu, and W. Yang, "PrivSet: Set-valued data analyses with locale differential privacy," in *Proc. INFOCOM*, Apr. 2018, pp. 1088–1096.
[11] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.
[12] R. Chen, H. Li, A. K. Qin, S. P. Kasiviswanathan, and H. Jin, "Private spatial data aggregation in the local setting," in *Proc. ICDE*, 2016, pp. 289–300.
[13] C. Dwork, "Differential privacy: A survey of results," in *Proc. Theory Appl. Models Comput.*, 2008, pp. 1–19.
[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. TCC*, 2006, pp. 265–284.
[15] F. Mcsherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. FOCS*, Oct. 2007, pp. 94–103.
[16] F. Mcsherry and I. Mironov, "Differentially private recommender systems: Building privacy into the netflix prize contenders," in *Proc. KDD*, 2009, pp. 627–636.
[17] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *J. Amer. Statist. Assoc.*, vol. 60, no. 309, pp. 63–69, 1965.
[18] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *Proc. USENIX Secur. Symp.*, 2017, pp. 729–745.
[19] L. Bonomi and L. Xiong, "A two-phase algorithm for mining sequential patterns with differential privacy," in *Proc. CIKM*, 2013, pp. 269–278.
[20] S. Shang, L. Chen, Z. Wei, C. S. Jensen, K. Zheng, and P. Kalnis, "Trajectory similarity join in spatial networks," *Proc. VLDB Endowment*, vol. 10, no. 11, pp. 1178–1189, 2017.
[21] T. Wang, N. Li, and S. Jha. (2017). "Locally differentially private heavy hitter identification." [Online]. Available: https://arxiv.org/abs/1708.06674
[22] M. Terrovitis, N. Mamoulis, and P. Kalnis, "Local and global recoding methods for anonymizing set-valued data," *VLDB J, Int. J. Very Large Data Bases*, vol. 20, no. 1, pp. 83–106, 2011.
[23] R. Bassily, K. Nissim, U. Stemmer, and A. G. Thakurta, "Practical locally private heavy hitters," in *Proc. NIPS*, 2017, pp. 2288–2296.
[24] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong, "Publishing set-valued data via differential privacy," *Proc. VLDB Endowment*, vol. 4, no. 11, pp. 1087–1098, 2011.
[25] X. Xiao, G. Bender, M. Hay, and J. Gehrke, "iReduct: Differential privacy with reduced relative errors," in *Proc. SIGMOD*, 2011, pp. 229–240.
[26] N. Wang, X. Xiao, Y. Yang, Z. Zhang, Y. Gu, and G. Yu, "PrivSuper: A superset-first approach to frequent itemset mining under differential privacy," in *Proc. ICDE*, Apr. 2017, pp. 809–820.

[27] S. Shang, R. Ding, B. Yuan, K. X. Xie, K. Zheng, and P. Kalnis, "User oriented trajectory search for trip recommendation," in *Proc. EDBT*, 2012, pp. 156–167.
[28] S. Shang, L. Chen, Z. W. Wei, C. S. Jensen, K. Zheng, and P. Kalnis, "Parallel trajectory similarity joins in spatial networks," *VLDB J., Int. J. Very Large Data Bases*, vol. 27, no. 3, pp. 395–420, 2018.
[29] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith, "Composition attacks and auxiliary information in data privacy," in *Proc. SIGKDD*, 2008, pp. 265–273.
[30] J. Cao, P. Karras, C. Raïssi, and K. L. Tan, "ρ-uncertainty: Inference-proof transaction anonymization," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 1033–1044, 2010.
[31] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Proc. Secur. Privacy*, May 2008, pp. 111–125.
[32] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei, "Anonymization-based attacks in privacy-preserving data publishing," *ACM Trans. Database Syst.*, vol. 34, no. 2, p. 8, 2009.
[33] Y. Li, G. Wang, Y. Yuan, X. Cao, L. Yuan, and X. Lin, "*PrivTS*: Differentially private frequent time-constrained sequential pattern mining," in *Proc. DASFAA*, 2018, pp. 92–111.
[34] X. Zhang, X. Meng, and R. Chen, "Differentially private set-valued data release against incremental updates," in *Proc. DASFAA*, 2013, pp. 392–406.
[35] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *Proc. NIPS*, 2014, pp. 2879–2887.
[36] T. T. Nguyen, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin. (2016). "Collecting and analyzing data from smart device users with local differential privacy," [Online]. Available: https://arxiv.org/abs/1606.05053
[37] J. W. Kim, D.-H. Kim, and B. Jang, "Application of local differential privacy to collection of indoor positioning data," *IEEE Access*, vol. 6, pp. 4276–4286, 2018.
[38] N. Wang *et al.*, "PrivTrie: Effective frequent term discovery under local differential privacy," in *Proc. ICDE*, Apr. 2018, pp. 821–832.
[39] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proc. FOCS*, Oct. 2013, pp. 429–438.
[40] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?" *SIAM J. Comput.*, vol. 40, no. 3, pp. 793–826, Jun. 2011.

XIANGGUO ZHAO received the Ph.D. degree in computer science from Northeastern University, Shenyang, in 1996, where he is currently an Associate Professor. His main research interests include XML data management, LBSN, and machine learning.

YANHUI LI received the B.E. degree from the College of Computer Science and Technology, Northeastern University, China, in 2013. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Northeastern University. In 2017, she visited the University of New South Wales as an Academic Visitor. Her main research interests include data privacy, differential privacy, and big data management.

YE YUAN received the B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, China, in 2004, 2007, and 2011, respectively, where he is currently a Professor with the Department of Computer Science. His research interests include graph databases, probabilistic databases, and social network analysis.

XIN BI received the M.S. degree in computer science from Northeastern University, Shenyang, in 2011, where he is currently pursuing the Ph.D. degree. His main research interests include XML data management and distributed database systems.

GUOREN WANG received the B.Sc., M.Sc., and Ph.D. degrees from the Department of Computer Science, Northeastern University, China, in 1988, 1991, and 1996, respectively, where he is currently a Professor. He has published more than 100 research papers. His research interests include XML data management, query processing and optimization, bioinformatics, high-dimensional indexing, parallel database systems, and P2P data management.