Efficient Time-Stamped Event Sequence Anonymization

REZA SHERKAT, JING LI, and NIKOS MAMOULIS, The University of Hong Kong

With the rapid growth of applications which generate timestamped sequences (click streams, GPS trajectories, RFID sequences), sequence anonymization has become an important problem, in that should such data be published or shared. Existing trajectory anonymization techniques disregard the importance of time or the sensitivity of events. This article is the first, to our knowledge, thorough study on time-stamped event sequence anonymization. We propose a novel and tunable generalization framework tailored to event sequences. We generalize time stamps using time intervals and events using a taxonomy which models the domain semantics. We consider two scenarios: (i) sharing the data with a single receiver (the SSR setting), where the receiver's background knowledge is confined to a set of time stamps and time generalization suffices, and (ii) sharing the data with colluding receivers (the SCR setting), where time generalization should be combined with event generalization. For both cases, we propose appropriate anonymization methods that prevent both user identification and event prediction. To achieve computational efficiency and scalability, we propose optimization techniques for both cases using a utility-based index, compact summaries, fast to compute bounds for utility, and a novel taxonomy-aware distance function. Extensive experiments confirm the effectiveness of our approach compared with state of the art, in terms of information loss, range query distortion, and preserving temporal causality patterns. Furthermore, our experiments demonstrate efficiency and scalability on large-scale real and synthetic datasets.

Categories and Subject Descriptors: K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Data sharing*

General Terms: Design, Algorithms, Performance, Experimentation

Additional Key Words and Phrases: Anonymity, privacy-preserving data publishing, user browsing history, time and URL generalization

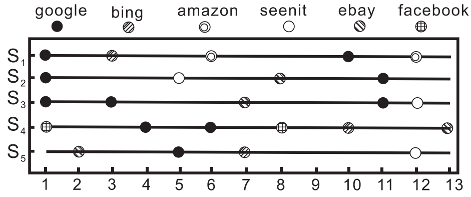
ACM Reference Format:
Sherkat, R., Li, J., and Mamoulis, N. 2013. Efficient time-stamped event sequence anonymization. *ACM Trans. Web* 8, 1, Article 4 (December 2013), 53 pages.
DOI:http://dx.doi.org/10.1145/2532643

1. INTRODUCTION

Consider an Internet Service Provider (ISP) who collects logs of HTTP requests sent by users, along with their IP addresses or any identifiers that users may provide to authenticate and to connect to the Internet. Figure 1(a) depicts a sample of such click streams, corresponding to the browsing history of five users in time interval [1–13]. For instance, click stream S_1 visits Google at times 1 and 10. At the same time, online stores and portals, which provide customized services (e.g., email, recommendations), can collect all time stamps of user visits to their websites. For instance, Google can collect the time stamps shown in Figure 1(b), which correspond to Google visits of the

This work was supported by grant HKU 715108E from Hong Kong RGC.
R. Sherkat is now affiliated with SAP Canada.
Authors' addresses: J. Li and N. Mamoulis, Department of Computer Science, The University of Hong Kong; R. Sherkat (corresponding author), SAP Canada; email: sherkat@gmail.com.

DOI:http://dx.doi.org/10.1145/2532643



(a)

ID	QIDs
Sue	1,10
Tom	1,11
Rob	1,3,11
Ted	4,6
Joe	5

(b)

Fig. 1. (a) Click streams of five users collected by an ISP, and (b) the time stamps that correspond to visits to Google.

click streams in Figure 1(a). Gaining access to the click stream data, collected by ISPs, enables a wide range of data analysis with social and commercial value. For instance, sharing click streams with a search engine allows the engine to model user temporal behavior using the aggregate frequency of visits to websites [Deshpande and Karypis 2004]. The model can be used to align services, for example, search results and advertisements, towards identified emerging trends in users' interests [Agichtein et al. 2006; Dupret and Piwowarski 2008; Koren 2010; Matthijs and Radlinski 2011]. However, sharing click streams can lead to serious privacy breaches, such as the notorious AOL privacy scandal [Barbaro and Zeller 2006], even if the identification information of individuals (e.g. IP address) is removed from the published data. We consider two practical scenarios in this article.

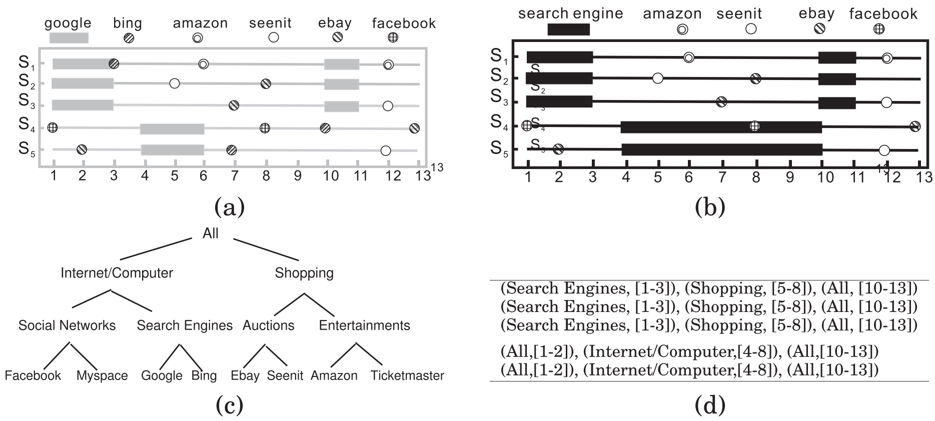
1.1. Motivating Examples

1.1.1. Sharing with Single Receiver (SSR). Assume that an ISP shares the click streams of Figure 1(a) with Google. Despite masking the real identifiers with random ones, the time stamps of Google visits can be used by Google to identify a user behind a sequence (*sequence identification*) and/or successfully guess some of the remaining events of a user's sequence (*event prediction*).[1] For instance, Google knows that only Sue and S_1 visited Google at time 1 and 10, thus S_1 can be associated to her. This association ensures that she visited Bing at 3 and Amazon at 6 and 12. Although the time points that Tom visited Google are not enough to associate him to one of S_2 or S_3, they provide enough evidence that he certainly visited eBay during time range [7–8].

Time stamps can be used to link a click stream provided by an ISP with a user ID held by the receiver. A privacy breach happens if the receiver matches a user ID with less than k click streams (k-anonymity [Samarati 2001]), or infers that the user visited a URL during any time interval of duration g with a probability over $\frac{1}{\ell}$. We call the latter requirement (g, ℓ)-diversity: an extension of ℓ-diversity [Machanavajjhala et al. 2006] to click streams. g represents the temporal aspect of sensitivity; intuitively, it makes sense to care about the certainty of an event within only a restricted time interval that makes this certainty statistically significant.

To prevent privacy breach by time stamp linkage, an ISP must anonymize data by replacing time points with intervals; we term this step *time generalization*. An anonymization for the click streams of Figure 1(a) is presented in Figure 2(a); only the time points of Google visits are generalized (i.e., replaced by time intervals) (e.g., S_1 visited Google during [1–3] and [10–11]). In addition to the uncertainty introduced in the time points, the *number* of visits to Google within each interval and each

[1] Disclaimer. We use company names to illustrate an example of possible adversaries in this article. We do not imply here that any real company has any malicious intentions for the use of their data.



(Search Engines, [1-3]), (Shopping, [5-8]), (All, [10-13])
(Search Engines, [1-3]), (Shopping, [5-8]), (All, [10-13])
(Search Engines, [1-3]), (Shopping, [5-8]), (All, [10-13])
(All,[1-2]), (Internet/Computer,[4-8]), (All,[10-13])
(All,[1-2]), (Internet/Computer,[4-8]), (All,[10-13])

Fig. 2. (a) Two-anonymity and (1, $\frac{3}{2}$)-diversity by time generalization in the SSR setting, when anonymized data is provided to Google, (b) event and time generalization in the SCR setting, when data is provided to Google and Bing, (c) a sample URL taxonomy, and (d) 2-anonymity by event and time generalization for an extreme SCR setting.

sequence is lost.[2] In Figure 2(a), Sue can be linked, with equal probability, to S_1, S_2, and S_3, as these sequences match with her regarding the time stamps of Google visits. Tom matches three sequences, two of them visited eBay during [7–8]. Thus, we infer that Tom visited eBay during [7–8] with probability $\frac{2}{3}$. Thus, Figure 2(a) satisfies 2-anonymity and (1, $\frac{3}{2}$)-diversity.

1.1.2. Sharing with Colluding Receivers (SCR). The time-generalization-based technique we described is meant for a single designated receiver. If the receiver decides to share its customized release with a third party, privacy can be threatened by the third party. For instance, Bing can identify the user behind click stream S_1 and learn all URLs (and time) s/he visited, if Google shares the data in Figure 2(a) with Bing. An ideal anonymization scheme must prevent attacks initiated by any subset of colluding parties. To achieve this goal, we propose reducing the certainty of both time stamps and URLs. We term this scheme *time and event generalization*.

The background knowledge of colluding receivers is a set of subsequences containing their common knowledge. For instance, if Google and Bing collude, the common background knowledge is at most[3] $\mathcal{B} = \{B_i\}_{i=1}^{5}$, where B_i is a sequence with all visits by S_i in Figure 1(a) to Google or Bing. Figure 2(b) demonstrates an anonymization which is provided to Google and Bing, assuming that they share data. Not only the time stamps of visits to Google and Bing are replaced by intervals, but also the URLs are replaced by *Search Engine* from the taxonomy in Figure 2(c). Any subsequence of $B_i \in \mathcal{B}$ matches with at least two sequences in Figure 2(b). In the extreme case, there is a risk that all parties collude, thus all time stamps and URLs need to be generalized, as shown in Figure 2(d). Each anonymized sequence is a set of (*category*, *time-interval*) pairs, where categories are derived from the taxonomy in Figure 2(c). Any subsequence of sequence S_i in Figure 1(a) matches with at least two sequences in Figure 2(d).

1.2. The Current State of the Art

In this section, we briefly review previous works that are directly related to our problem. An in depth survey of other related works appear in Section 9.

[2] The representation might be augmented to encode click statistics.

[3] This is the extreme case in which the colluding parties match the time stamps they collect with user IDs.

Recently, there has been a growing interest in anonymizing search engine query logs and browsing histories (e.g. [Götz et al. 2009; He and Naughton 2009; Jones et al. 2007; Korolova et al. 2009; Machanavajjhala et al. 2011; Pang et al. 2010; Terrovitis et al. 2008]). They transform queries using a taxonomy, augment queries, or suppress some keywords to achieve privacy. Surprisingly, the time dimension of click-streams has been totally ignored in all these works. In terms of data utility, the time stamps are important for extracting causality patterns [Koren 2010]. For instance, from the anonymized data in Figure 2(a), Google learns the probability that a user visits Amazon after (or before) five minutes of visiting Google (we study such queries in Section 8.3). Moreover, the time stamps make queries more specific. A combination of queries and time stamps can be used by adversaries to launch stronger attacks. For instance, in the AOL dataset (Section 8.1) an attacker can identify 56% of click streams if the attacker knows the visited URL. An attacker who knows a (URL, time stamp) pair can identify 68%, 93%, and 99% of click streams if the recorded time granularity is, respectively, an hour, a minute, and a second.[4] This article is the first work, to our knowledge, which studies the anonymization of time-stamped event sequences using both time and event generalization in order to resist against such a strong attacker.

In both SSR and SCR settings, one can find naive (legitimate) anonymizations, (e.g., (ALL, [1–13]) for Figure 1(a)). However, the naive solution reduces the utility of the anonymized data because the receiver does not learn any extra information beyond the global time interval. Our objective is to minimize the data distortion along time and event dimensions and at the same time protect privacy. For this reason, we adapt well-known anonymity models for microdata [Machanavajjhala et al. 2006; Sweeney 2002] to time-stamped event sequences. Like the itemset anonymization problem [He and Naughton 2009; Terrovitis et al. 2008; Xu et al. 2008], the key difference of anonymizing time-stamped event sequences and traditional relational data is the lack of a fixed-length schema. While in relational data, an attribute range (e.g., Age: [20–25]) blurs only one attribute (Age) of each tuple, in our setting there is more flexibility; an interval blurs a set of attributes in terms of (event, time) pairs. For instance, (Search Engines, [1–3]) in Figure 2(d) generalizes two clicks of S_1 (i.e., (Google, 1) and (Bing, 3)), but only one click of S_2. This, requires a tailored definition of information loss which takes this flexibility into account. Furthermore, the notion of sensitive attributes in our setting is defined in relation with time, whereas often an attribute (e.g., disease) is deemed sensitive. We propose a flexible privacy model which regards the sensitivity of event prediction along any time interval of specified length.[5] This distinguishes our work from set anonymity [He and Naughton 2009; Terrovitis et al. 2008] inspired by k-anonymity.

Our treatment of diversity is similar to the (h, k, p)-coherence model [Xu et al. 2008] for itemset anonymity. In fact, one may wish to *augment* an item taxonomy with time stamps and then specialize the suppression-based algorithm of Xu et al. [2008]. We review this model and explain why we did not use this approach to our problem. The (h, k, p)-coherence model requires that any combination of p items should appear in at least k transactions, and each sensitive item must not appear in more than h percent of these transactions. Otherwise, systematic item suppression is applied to achieve (h, k, p)-coherence. For a retail dataset, the Xu et al. [2008] report under 30% suppression for a wide range of values for parameter p (see Figure 8(b) of [Xu et al. 2008]). Unfortunately, their method brings severe suppression when item distribution is skewed. In particular, for the AOL dataset (explained in Section 8.1), even if the

[4] For the users that have at least one visit to Google. Many websites (e.g., [Google 2011]) record/keep visit time stamps in the granularity of a second.

[5] One can customize (g, ℓ)-diversity model to account only for the prediction probability of sensitive events.

time stamps of clicks were replaced with the corresponding month, it turns out that 68% of click streams have unique (URL, time) pairs and must be suppressed due to Xu et al. [2008], when p is as small as one. In this article, we locally generalize time stamp and URLs to avoid such a catastrophic information loss by global suppression.

Recently, differential privacy [Dwork et al. 2006] has received a growing interest in the database community. This model bounds the inference of a strong adversary who knows all records of database except one. Dwork et al. [2006] add well-designed Laplace noise to data to bound the inference power of a strong attacker. Achieving differential privacy by Laplace noise has become a recurring theme in many works (e.g., [Chen et al. 2012a; Ding et al. 2011; Götz et al. 2009; Korolova et al. 2009; Machanavajjhala et al. 2008; Xiao et al. 2010]). As shown in Section 9.5, considering time stamps with events makes data very sparse and the problem more challenging. Even when the time dimension is ignored, a severe data distortion is introduced, when data is sparse, to fulfill differential privacy (e.g., [Terrovitis et al. 2012]) or a *relaxed* version of it (e.g., [Götz et al. 2009]). In particular, our study of applying the algorithm of Götz et al. [2009] for click-streams (see Section 9.5) confirms a severe degree of suppression. Furthermore, mixing time stamps with URLs makes a large number of n-grams extracted from click streams by Chen et al. [2012a] approach unique, even for very small values of n. For instance, when the granularity of a recorded time stamp is one second, 99% of (event, time) pairs (i.e., 1-grams) were unique in our dataset. In Chen et al.'s [2012a] framework, this implies that a majority of nodes in the prefix tree constructed from the n-grams will be of depth one. Consequently, the synthetic dataset generated using the induced Markov model cannot preserve most of the sequential information inherent in click streams. An interesting research problem, beyond the scope of this article, is to extend techniques for achieving differential privacy to further retain the utility in the case of highly sparse time-stamped event sequences.[6]

Several works (surveyed in Section 9.7) have been proposed for trajectory anonymization to resist sequence identification attacks. These solutions (1) ignore the time points of sequences [Abul et al. 2008; Terrovitis and Mamoulis 2008], (2) use time points only for grouping sequences but only generalize location [Yarovoy et al. 2009], or (3) perform time and location generalization but apply point suppression [Nergiz et al. 2009]. As we experimentally show in Section 8.3, suppression reduces the utility of anonymized data in several aspects (e.g., preserving temporal causality patterns and range query distortion) and it may be critical when false negatives are not acceptable. Furthermore, the sensitivity with respect to the time dimension was not considered before. This renders existing methods vulnerable to event prediction attacks. We tackle this problem by directly including time in (g, ℓ)-diversity, which offers a tunable privacy model with respect to both time and events (i.e., URLs).

1.3. Contributions

We propose a measure for data distortion caused by time and event generalization and give a polynomial time algorithm (ANONYMIZE) to find an optimal set of intervals for each anonymization group. We extend traditional privacy models to event sequences and integrate time into the event diversity (the (g, ℓ)-diversity model). We propose a semisupervised clustering algorithm (BASELINE) to derive anonymization groups. This approach is akin to Utility-based Data Partitioning (UDP) [Xu et al. 2006]. UDP offers better overall utility vs. multidimensional partitioning (MP) (e.g., Mondrian [LeFevre et al. 2006]), as reported before [Abul et al. 2008; Ghinita et al. 2007; Xu et al. 2006].[7] However, MP scales better with database size. For our problem,

[6] One may apply time and/or event aggregation as a preprocessing step to reduce data sparseness.

[7] See Sections 3 and 9.3 for two different adaptations of Mondrian to our problem.

we propose algorithms to bridge the performance gap between MP and UDP. This way we can efficiently derive anonymized data with better overall utility. For this, we first focus on the SSR setting and propose an incremental pruning scheme (FASTNN) which uses an index structure and three lower bounds of utility. We then propose a hybrid partitioning scheme (HYBRID) and a fast grouping method based on the distribution of time points to further improve the performance. For the SCR setting, we propose a partition-and-refine paradigm (algorithm PR) which uses a novel taxonomy-aware distance (eventDist) and our optimizations for the SSR setting.

Experimental evaluation on large-scale real and synthetic data from diverse domains confirms the efficiency, quality, and scalability of our algorithms. In particular, As we show experimentally in Section 8.3, our methods achieve good utility not only in terms of our information loss measure, but two more general measures; the amount of distortion for range queries and the ability to preserve causality patterns.

2. DATA MODEL AND PROBLEM SETTING

Let E be the set of all possible *events*, for example, all URLs. The database $\mathcal{D}$ is a collection of time-stamped event sequences. Each sequence $S = \{(e_i, t_i)\}_{i=1}^{|S|}$ in $\mathcal{D}$ is a set of $|S|$ pairs. Each pair $(e, t) \in S$ denotes the participation of S in event $e \in E$ at time t. The background knowledge of a receiver is database $\mathcal{B}$ of time-stamped event sequences. We assume that $\mathcal{B}$ is subsumed by $\mathcal{D}$, that is, each $S_b \in \mathcal{B}$ must be a subsequence of (at least) one sequence in $\mathcal{D}$ but corresponds to only one sequence $S_i \in \mathcal{D}$. The sequences in $\mathcal{B}$ are used as quasi-identifiers (QIDs). We assume that only the time stamps of events in $E_r \subseteq E$ are collected by the receiver. More formally, $\forall (e, t) \in S_b \in \mathcal{B}, e \in E_r$.

Example 2.1. Consider the example in Section 1.1.1, where $E_r = \{\text{Google}\}$. The set $\mathcal{B}$ in this setting has five sequences: {(Google, 1), (Google, 10)}, {(Google, 1), (Google, 11)}, {(Google, 1), (Google, 3), (Google, 11)}, {(Google, 4), (Google, 6)}, and {(Google, 5)}.

Example 2.2. Consider the first example of Section 1.1.2, where $E_r = \{\text{Google}, \text{Bing}\}$. The set $\mathcal{B}$ in this setting has five sequences: {(Google, 1), (Bing, 3), (Google, 10)}, {(Google, 1), (Google, 11)}, {(Google, 1), (Google, 3), (Google, 11)}, {(Google, 4), (Google, 6), (Bing, 10)}, and {(Google, 5), (Bing, 7)}.

2.1. Generalization Model

To anonymize $\mathcal{D}$, we follow a partition-based approach [Sweeney 2002]. We divide $\mathcal{D}$ into non-overlapping partitions which are called *anonymization group* (AG). The set $\mathcal{P} = \{\mathcal{S}_1, \ldots, \mathcal{S}_{|\mathcal{P}|}\}$ is a partitioning of $\mathcal{D}$ if $\cup_{i=1}^{|\mathcal{P}|} \mathcal{S}_i = \mathcal{D}$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for any $1 \leq i < j \leq |\mathcal{P}|$. We generalize the sequences in each group independently and publish the generalized sequences as $\mathcal{D}^*_{\mathcal{P}}$. While forming the groups depends mostly on the privacy model (Section 2.3), the generalization step (Section 7) is often independent from the partitioning step and mostly focuses on the indistinguishability of data in the quasi-identifier (QID) space, and the amount of information loss in each AG. We generalize each time stamp using a time interval and each event using a category from a taxonomy (e.g., Figure 2(c)) only for event pairs $(e, t) \in S_b \in \mathcal{B}$.[8] To improve

[8] There are other variations of generalizations which we do not study in this article, for example, leave $(e, t) \in S_b \in \mathcal{B}$ intact but generalize (e, t) pairs not collected by a receiver, or selectively generalize subset of (e, t) pairs in $\mathcal{D}$ irrespective of $\mathcal{B}$. In this article, though, we decided to be consistent with the data privacy literature, where often quasi-identifier attributes are generalized and sensitive attributes remain intact. Examining other generalizations and their impact on the privacy model and utility is an interesting research direction.

data utility, we opt for local recoding; we do generalizations on the granularity of each anonymization group (AG). We represent a set of sequences $\mathcal{S}$ in the same AG using a set of intervals $\mathcal{I} = \{(c_i, [s_i - e_i])\}_{i=1}^{|\mathcal{I}|}$, where c_i is a category and $[s_i - e_i]$ is a time interval, $s_i \leq e_i$. We enforce the following property on $\mathcal{S}$ and $\mathcal{I}$.

PROPERTY 2.3 (STRONG COVERAGE (SC)). *Let the set of intervals $\mathcal{I}_{\mathcal{S}}$ represent the anonymization group $\mathcal{S}$. Each interval $(c_i, [s_i - e_i]) \in \mathcal{I}_{\mathcal{S}}$ must cover at least one (e, t) from each sequence in $\mathcal{S}$ if $e \in E_r$; event e is under category c_i and $t \in [s_i - e_i]$. There is no sequence in $\mathcal{S}$ with a pair (e, t) which is not covered by any interval $I \in \mathcal{I}_{\mathcal{S}}$ if $e \in E_r$.*

Example 2.4. In Figure 2(a), when the click-streams are shared with Google, we have $E_r = \{\text{Google}\}$. The interval set $I_1 = \{(\text{Google},[1–3]), (\text{Google}, [10–11])\}$ covers all Google visits to for all sequences in the anonymization group $\mathcal{S}_1 = \{S_1, S_2, S_3\}$. When $E_r = E$, that is, the set of all URLs, it is assumed that all receivers can share data with each other. For this setting, the interval set $I_2 = \{(\text{All}, [1–2]), (\text{Internet/Computer}, [4–8]), (\text{All}, [10–13])\}$ covers all visits for all sequences in the anonymization group $\mathcal{S}_2 = \{S_4, S_5\}$.

Violating SC has two drawbacks: (1) not generalizing (at least) one time stamp of sequence $S_b \in \mathcal{B}$ may cause a direct identification of S_b, and (2) having an interval which does not cover (at least) one time stamp of S_b indicates a fake event participation for S_b and therefore publishing wrong data. Note that the interval set $\{(\text{All}, [\min_{S\in\mathcal{D}} \mathsf{start}(S) - \max_{S\in\mathcal{D}} \mathsf{end}(S)])\}$ satisfies the SC property, where $\mathsf{start}(S)$ and $\mathsf{end}(S)$ are, respectively, the smallest and the largest time stamp of S. Thus, there is at least one set of intervals for each anonymization group that satisfies the SC property.

2.2. Information Loss Measure

The utility of anonymized data depends on the analysis of interest and the query workload. Because the query is not fixed ahead, it is impossible to find an anonymization which provides the most accurate result for every query. Thus, as it was practiced before [Bayardo and Agrawal 2005; LeFevre et al. 2006], we propose a data distortion measure as a surrogate for query answerability and find an anonymization which achieves the desired privacy with minimum information loss. We start by defining data distortion of a single interval I along the time and event dimensions. Then, we expand this measure to a set of intervals $\mathcal{I}$. Finally, we define a distortion measure for an anonymization corresponding to partitioning $\mathcal{P}$ of database $\mathcal{D}$.

Assume that we use interval $I = (c_i, [s_i - e_i])$ to represent a set of (event, time) pairs in time interval $[s_i - e_i]$ using the event category c_i. This adds uncertainty by time and event generalization. We use two measures IL_e and IL_t to quantify each loss separately. $IL_e(I)$ is the normalized distortion due to generalizing into event category c_i.

$$IL_e(I) = \begin{cases} 0, & \text{if } c_i \text{ is an event (not generalized),} \\ |c_i|/|E|, & \text{otherwise,} \end{cases}$$

where $|c_i|$ is the number of events e_j in the taxonomy under c_i. $IL_t(I)$ is the normalized distortion due to generalizing time stamps into interval $[s_i - e_i]$ and is defined as

$$IL_t(I) = \frac{(e_i - s_i)}{\max_{S\in\mathcal{D}} \mathsf{end}(S) - \min_{S\in\mathcal{D}} \mathsf{start}(S)}. \tag{1}$$

Both IL_e and IL_t are normalized and fall in the range [0–1], where 1 means the complete distortion of corresponding value. These two measures can be unified into one measure using a monotonically increasing function[9], such as *IDD*.

$$IDD(I) = \frac{w_t \cdot IL_t(I) \, + \, w_e \cdot IL_e(I)}{w_t + w_e}, \tag{2}$$

where the weights w_t and w_e, respectively, capture the relative importance of time and event uncertainty.[10] For instance, when the accuracy of events is more important than time in the anonymized data, we set $w_e > w_t$ to penalize event distortion more than time distortion; in the extreme case, $w_e \rightarrow \infty$ and Eq. (2) simplifies to $IL_e(I)$ and our anonymization target simplifies to the traditional query-log anonymization [Götz et al. 2009; Jones et al. 2007; Korolova et al. 2009]. In the most compact interval $I = (c_i, [s_i - e_i])$ for a sequence $S_b \in \mathcal{B}$, we must set $s_i = \mathsf{start}(S_b)$, $e_i = \mathsf{end}(S_b)$, and c_i to the *lowest common ancestor* of events $\{e \mid (e,t) \in S_b \, \wedge \, s_i \leq t \leq e_i\}$ in the taxonomy.

Example 2.5. Consider the dataset in Figure 1(a) and the taxonomy of Figure 2(c). For the interval I = (Internet/Computer, [8–10]), $IL_e(I) = \frac{4}{8}$ and $IL_t(I) = \frac{2}{12}$. The lowest common ancestor for the event sets $E_1 = \{\text{Ebay, Amazon}\}$, $E_2 = \{\text{Face, Myspace}\}$, and $E_1 \cup E_2$ are, respectively, Shopping, Social Networks, and All.

Eq. (2) combines the distortions along time and event dimensions into one measure in the range $[0, 1]$ with a smaller value more desirable and indicative of a tighter representation within the time range of I. For a set of sequences $\mathcal{S}$ and *a set of* intervals $\mathcal{I}$, we integrate *IDD* to quantify the distortion of generalizing sequences of $\mathcal{S}$ using $\mathcal{I}$.

$$IL(\mathcal{S}, \mathcal{I}) = \frac{\sum_{I \in \mathcal{I}} \sum_{S \in \mathcal{S}} |S_I| \cdot IDD(I)}{\sum_{S \in \mathcal{S}} |S|}, \tag{3}$$

where $S_I = \{(e,t) \in S | e \in E_r \, \wedge \, s_i \leq t \leq e_i\}$. The term $IDD(I)$ does not capture the *number* of (event, time) pairs that are approximated by interval I. Multiplying $IDD(I)$ by the term $|S_I|$ in Eq. (3) penalizes the distortion of each interval I by the number of (event, time) pairs it approximates. Next, we define our metric of *goodness* (homogeneity) for a set of sequences grouped in the same anonymization group $\mathcal{S}$. Intuitively, the are many possible set of intervals that can represent the same anonymization group. In order to measure the goodness of $\mathcal{S}$, we consider the interval set which has the smallest information loss to represent sequences in $\mathcal{S}$. Formally,

$$CP(\mathcal{S}) = \min_{\forall \mathcal{I}} \, \left\{ IL(\mathcal{S}, \mathcal{I}) \right\}. \tag{4}$$

In other words, *IL* depends on a specific set of intervals $\mathcal{I}$ used to represent $\mathcal{S}$, whereas *CP* depends on the information loss for the optimal set of intervals for $\mathcal{S}$. Therefore, a zero *CP* indicates that all sequences in $\mathcal{S}$ are equal. In Section 7, we show how to compute $CP(\mathcal{S})$ and the optimal set of intervals $\mathcal{I}$ which minimizes $IL(\mathcal{S}, \mathcal{I})$ for an anonymization group $\mathcal{S}$. We use *CP* to quantify the amount of uncertainty introduced by anonymizing database $\mathcal{D}$ using partitioning $\mathcal{P}$ as

$$NCP(\mathcal{D}, \mathcal{P}) = \frac{\sum_{\mathcal{S} \in \mathcal{P}} |\mathcal{S}| \cdot CP(\mathcal{S})}{|\mathcal{D}|}, \tag{5}$$

which is a normalized distortion measure in the range $[0 - 1]$.

[9] $f(x,y)$ is a monotonically increasing function, if $f(x_1,y) \leq f(x_2,y)$ for any $x_1 \leq x_2$ and $f(x,y_1) \leq f(x,y_2)$ for any $y_1 \leq y_2$.

[10] We assume that the weights are nonnegative ($w_t \geq 0$ and $w_e \geq 0$) and $w_t + w_e > 0$.

2.3. Privacy Model and Problem Setting

We consider two types of attacks, namely, sequence identification and event prediction. For the first attack, we directly use the traditional k-anonymity [Samarati 2001].

Definition 2.6 (*k-anonymity*). Partitioning $\mathcal{P}$ of database $\mathcal{D}$ is k-anonymous if no adversary that monitors events E_r can associate a sequence in $\mathcal{B}$ with less than k sequences in $\mathcal{D}^*_{\mathcal{P}}$.

In order to quantify the risk of event prediction attack, we extend the ℓ-diversity [Machanavajjhala et al. 2006] model, originally proposed for microdata, to event sequences. First, we must provide a measure for event prediction probability. For a set of sequences $\mathcal{S}$ and event set E_r, the probability of observing event $e \in E - E_r$ during time interval $\mathcal{T}$ in any sequence of $\mathcal{S}$ is $\Pr(e, \mathcal{T}|\mathcal{S}) = \frac{|\{S \in \mathcal{S} \mid \exists (e,t_i) \in S,\ t_i \in \mathcal{T}\}|}{|\mathcal{S}|}$. Formally, in an event prediction attack for sequence $S_b \in \mathcal{B}$, the adversary uses the pairs $(e, t) \in S_b$ to first find the partition in the anonymized data, which contains S_b. The adversary uses the partition found to make inference about the participation of S_b in events in $E - E_r$ during any time interval of interest. As an example of this attack, consider Figure 2(a) again, where E_r = {Google} and let S_b = {(Google, 1), (Google, 10)}. Here, S_b corresponds to sequence S_1 in Figure 1(a). An adversary can learn that S_b is a sequence inside $\mathcal{S} = \{S_1, S_2, S_3\}$, thus $\Pr(e, \mathcal{T}|\mathcal{S}) = \frac{1}{3}$, for e = Bing and $\mathcal{T}$ = [1–3].

To provide a bound on the inference power of an adversary, in terms of predicting event participation time for any sequence, we extend the traditional ℓ-diversity [Machanavajjhala et al. 2006] into (g, ℓ)-diversity as follows.

Definition 2.7 (*(g, ℓ)-diversity*). Let $E_r \subseteq E$ be the set of events monitored by the receiver(s) and $E_s = E - E_r$ be the events not monitored by receivers, thus regarded as sensitive events. A partitioning $\mathcal{P}$ of database $\mathcal{D}$ satisfies (g, ℓ)-diversity if $\Pr(e, \mathcal{T}_g|\mathcal{S}) \leq \frac{1}{\ell}$ for any event $e \in E_s$, any time interval $\mathcal{T}_g$ of length g, and any set $\mathcal{S} \in \mathcal{P}$.

Informally, no receiver, using the anonymized data can associate an event which is not monitored by the receiver to any sequence with a certainty over $1/\ell$ within any time interval of length g or shorter.[11] This definition, protects against sensitive event (URL) inference within any time interval of duration g or less. In this regard, Definition 2.7 is flexible; as long as for any time window $\mathcal{T}_g$, the adversary is confused for the event participation of a sequence, we consider the anonymized data privacy preserving. Thus, an event is not deemed sensitive only because it belongs to the set of sensitive events. If an adversary finds about an event but cannot not refine its time stamp to an interval shorter than g with a certainty more than $\frac{1}{\ell}$, we do not consider this a privacy breach. Of course, if event participation is considered sensitive regardless of its time, we can set $g = \infty$ and specialize (g, ℓ)-diversity to ℓ-diversity [Machanavajjhala et al. 2006].[12] Parameter g and ℓ control the resolution of event prediction. If g is set to a large value, our model is very similar to the simplified ℓ-diversity; the time stamps of events do not play any role on their sensitivity. If g is set to a very small value, our model insists on the diversity of events in short time intervals; all click-streams in an anonymization group may participate in event $e \in E_s$ but the participations time are forced to spread across the time dimension. A domain expert can help to select a suitable value for g.

[11] Trivially, (g, ℓ)–diversity does not imply $(g+1, \ell)$–diversity, as ℓ-diversity does not imply $(\ell+1)$–diversity, and k–anonymity does not imply $(k+1)$–anonymity.

[12] For instance, visiting porn URL might be considered sensitive. An alternative approach, not studied here, is to generalize sensitive URLs.

Definition 2.7 can be slightly modified such that E_s only includes a subset of events in $E - E_r$ which are considered as sensitive. On the other hand, and in the extreme case of the SCR setting, $E_r = E$ and E_s is empty. In this case, unknown receivers may exchange time stamps or perform matching based on the identifiers (e.g., IP addresses). In this case, there is no gain in anonymization beyond k-anonymity because the receivers already have complete sequences, in the extreme case. Otherwise, (g, ℓ)-diversity ensures that the certainty of an adversary about event participation for any event in E_s is bounded by $\frac{1}{\ell}$ in any time interval of duration g or less.

LEMMA 2.8 (MONOTONICITY OF (g, ℓ)-DIVERSITY). *For two non-overlapping sets of sequences $\mathcal{S}_1$ and $\mathcal{S}_2$, any event $e \in E - E_r$ and any time interval $\mathcal{T}_g$,*

$$\Pr(e, \mathcal{T}_g | \mathcal{S}_1 \cup \mathcal{S}_2) \leq \max\left\{ \Pr(e, \mathcal{T}_g | \mathcal{S}_1), \Pr(e, \mathcal{T}_g | \mathcal{S}_2) \right\}.$$

PROOF. Without loss of generality let $P_1 \geq P_2$, where

$$P_1 = \Pr(e, \mathcal{T}_g | \mathcal{S}_1) = \tfrac{n_1}{|\mathcal{S}_1|} \text{ and } P_2 = \Pr(e, \mathcal{T}_g | \mathcal{S}_2) = \tfrac{n_2}{|\mathcal{S}_2|}.$$

Thus $(n_1 + n_2) = \left(P_1 \cdot |\mathcal{S}_1| + P_2 \cdot |\mathcal{S}_2|\right) \leq \left(|\mathcal{S}_1| + |\mathcal{S}_2|\right) \cdot P_1$. Because $\mathcal{S}_1$ and $\mathcal{S}_2$ are non-overlapping, we have: $\Pr(e, \mathcal{T}_g | \mathcal{S}_1 \cup \mathcal{S}_2) = \frac{n_1+n_2}{|\mathcal{S}_1 \cup \mathcal{S}_2|} \leq P_1 = \max\{P_1, P_2\}$. □

PROBLEM 2.9. *Given the event set $E_r \subseteq E$ and database $\mathcal{D}$, find a partitioning $\mathcal{P}$ of $\mathcal{D}$ such that the information loss measure $NCP(\mathcal{D}, \mathcal{P})$ is minimized and $\mathcal{P}$ satisfies the desired privacy model (i.e., k-anonymity or (g, ℓ)-diversity).*

3. OVERVIEW OF OUR SOLUTIONS

Problem 2.9 is NP-hard; it reduces to the traditional microdata anonymity [Meyerson and Williams 2004] when all sequences have the same length (greater than one). We could solve this problem by adapting Mondrian [LeFevre et al. 2006], an algorithm which was proposed to anonymize multidimensional data.[13] Before this, we must transform $\mathcal{D}$ into a *relational* database $\mathcal{D}_R$. For this, as suggested by Abul et al. [2008], one column is required for each time stamp $t \in T$, the set of possible time stamps in $\mathcal{D}$. The value for column t for each record in $\mathcal{D}_R$ is equal to the URL visited by the corresponding sequence in $\mathcal{D}$. This way, we end up with a database with extremely large number of dimensions. Although Mondrian has a better performance, compared with *utility-based* top-down and bottom-up partitioning (e.g., [Xu et al. 2006]), Mondrian often produces anonymizations with larger information loss and inferior utility, specially for high dimensional data [Abul et al. 2008; Ghinita et al. 2007; Iwuchukwu and Naughton 2007; Xu et al. 2006]. On the other hand, utility-based partitioning techniques, unlike Mondrian, do no scale well with database size [Nergiz et al. 2009; Xu et al. 2006]. This is mainly because utility-based partitioning needs $O(|\mathcal{D}|^2)$ computations of information loss measure (i.e., utility) and $O(|\mathcal{D}|^2)$ disk access, which is daunting for large databases with billions of click-streams. Storing all data in main memory can alleviate the second performance bottleneck. However, still CPU cost is of paramount importance, because the information loss measure is often computationally expensive in our case (see Section 7). Therefore, we propose efficient algorithms that take advantage of the properties of our information loss measure (*CP* in Eq. (4)) in order to speed up utility-based partitioning from $O(|\mathcal{D}|^2)$ to $O(|\mathcal{D}| \cdot \log |\mathcal{D}|)$.

[13] See Section 9.3 for an alternative adaptation of Mondrian to our problem.

Table I. Definitions of Symbols

Symbol	**Definition**
E	The set of all events (URLs)
E_r	The set of events monitored by a receiver (e.g., {Google})
T	The set of all possible time points
$\mathcal{D}$	A database of sequences with $N = \lvert\mathcal{D}\rvert$ sequences
$\mathcal{B}$	A database of sequences collected by a receiver who Monitors all participations for the events in E_r
S, S_i	Sequence S, sequence S_i
start(S) (end(S))	The time stamp of the earliest (latest) event in sequence S
C, C_i	Cluster of sequences C, cluster of sequences C_i
start(C) (end(C))	The smallest (largest) time stamp among all sequences in cluster C
$\lvert C\rvert$	Number of sequences in cluster C
$(C_i, C_j) \in Con_{\neq}$	Merging C_i and C_j violates a privacy constraint (see Def. 4.1)
$\mathcal{S}$	Anonymization group (AG) $\mathcal{S}$, a group of sequences
$\mathcal{P}, \mathcal{P}_i$	Partitioning of database $\mathcal{D}$ into AGs $\mathcal{P}$ and $\mathcal{P}_i$
C	A cluster extent; represents a set of clusters (see Sec. 5.2.1)
$CP(\mathcal{S})$	Minimum information loss when sequences in $\mathcal{S}$ are generalized
$NCP(\mathcal{D}, \mathcal{P})$	Information loss of database $\mathcal{D}$ when AG $\mathcal{P}$ is applied
$D_{LB}, D_{fp}, D_{extent}$	Lower bounds for $CP(\mathcal{S})$ for group $\mathcal{S}$ at 3 levels of granularity
eventDist(A, B)	The loss of minimum generalization which makes the events of Two sequences A and B the same (Sec. 6.2)
$\lvert$region$A\rvert$ ($\lvert$region$B\rvert$)	The number of sequences in region A (B) in Sec. 5.3
[s–e]	The time interval starting from s and ending at e (both inclusive)

This bridges the performance gap between Mondrian and utility-based partitioning. Our techniques are presented as follows.

— In Section 4, we propose the BASELINE algorithm which effectively finds a partitioning for sharing with single receiver (SSR) or sharing with colluding receivers (SCR), for either of the privacy models. However, BASELINE is not scalable to large datasets. In Section 5, we propose optimizations for the SSR setting and time generalization.

— In Section 6, we consider the SCR setting with time and event generalization. Although BASELINE algorithm can be used here, but the optimizations proposed in Section 5 cannot be used directly in the SCR setting. In view of this, we propose a top-down iterative algorithm which integrates the optimizations for time generalization with a novel taxonomy-based distance function.

— In Section 7, we present our approach for computing data generalizations, after the partitioning phase is complete. This step is common to both SSR and SCR settings and orthogonal to the data partitioning step. The objective is to define the optimal set of intervals for publishing the anonymized data.

4. THE BASELINE ALGORITHM

The Utility-based Data Partitioning (UDP) approach was first introduced by Xu et al. [2006] for relational data and k-anonymity. In this section, inspired by the COP-COBWEB clustering algorithm of Wagstaff and Cardie [2000], we first propose a bottom-up UDP algorithm called BASELINE to achieve either (g, ℓ)-diversity or k-anonymity for time-stamped event sequences. While Xu et al. [2006] only consider utility (i.e., information loss measure) as a criteria of constructing anonymization groups (AGs), we also consider the privacy model as *instance-level constraints*. In BASELINE, the term *cluster* refers to a set of sequences and is equivalent to an AG.

We partition the database into a set of non-overlapping clusters such that the total information loss is minimized and each cluster satisfies the desired privacy model. To achieve this, BASELINE has a greedy conservative grouping followed by a merge. In each iteration of the greedy phase, it randomly selects a candidate cluster from the set of available clusters and merges it with its closest cluster.[14] Closeness is measured in terms of the information loss (*CP* in Eq. (4)) of the merged cluster. Unlike the bottom-up clustering algorithm of Xu et al. [2006], the conservative merge employs instance level constraints, as defined in Section 4.1. This is to ensure that at the end of phase one, only clusters with less than ℓ sequences (under-filled clusters) violate (g,ℓ)-diversity. In the second phase, under-filled clusters will be iteratively merged with their closest cluster until every cluster satisfies (g,ℓ)-diversity. The second phase of BASELINE always converges to a solution because of the monotonic property of (g,ℓ)-diversity.

4.1. Instance-Level Constraints

COP-COBWEB is a hierarchical clustering algorithm which supports instance-level constraints. For any pair of objects o_1 and o_2, the constraint does not allow two clusters C_1 and C_2 that contain these two objects to be merged. Similarly, BASELINE verifies privacy violation using instance-level constraints. We define an instance-level constraint for (g,ℓ)-diversity as follows.

Definition 4.1 (*Instance-Level Constraint for* (g,ℓ)*-diversity*). Let clusters C_1 and C_2 be two non-overlapping clusters, and let $|C_1| < \ell$ and $|C_2| < \ell$. We say that C_1 and C_2 violate an instance-level constraint, if there is an event $e \in E - E_r$ such that

$$\Big((e,t_1) \in S_1 \in C_1\Big) \bigwedge \Big((e,t_2) \in S_2 \in C_2\Big) \bigwedge \Big(|t_1 - t_2| \leq g\Big). \tag{6}$$

The violation of an instance level constraint for C_1 and C_2 is denoted by $(C_1, C_2) \in Con_{\neq}$, following the notation originally introduced in Wagstaff and Cardie [2000].

Example 4.2. Let $C_1 = \{S_1, S_2\}$ and $C_2 = \{S_3\}$ be two clusters, where S_1, S_2, and S_3 are the sequences in Figure 1(a). Then $C_{1,2} = C_1 \cup C_2$ violates (g,ℓ)-diversity when $g = 1$, $\ell = 3$, and $E_r = \{\text{Google}\}$, because $\Pr\big(\text{ebay}, [7\text{–}8] \mid C_{1,2}\big) = \frac{2}{3} > \frac{1}{3} = \frac{1}{\ell}$. For the same E_r and g but when $\ell = \frac{3}{2}$, $C_{1,2}$ satisfies (g,ℓ)-diversity although $\Pr\big(\text{Google}, [10\text{–}11] \mid C_{1,2}\big) = 1$ because only the prediction probability of the events that are not in the set E_r is important in (g,ℓ)-diversity model (see Definition 2.7).

Intuitively, one can verify that $(C_1, C_2) \in Con_{\neq}$ by Definition 4.1 implies that $\Pr\big(e, [t_1 - t_2] \mid C_1 \cup C_2\big) \geq \frac{2}{|C_1|+|C_2|} > \frac{1}{\ell}$ for at least one event $e \in E - E_r$ common to both clusters. This is indeed a violation of (g,ℓ)-diversity. However, if the condition in Eq. (6) is evaluated as false for every event e in cluster $C_{1,2} = C_1 \cup C_2$, then $C_{1,2}$ always satisfies (g,ℓ)-diversity if $|C_{1,2}| \geq \ell$, and violates (g,ℓ)-diversity if $|C_{1,2}| < \ell$. This is why BASELINE only merges under-filled clusters in the second step.

Note that although we mainly link the instance level constraints with a privacy model, the constraint can also incorporate a subset of domain-level requirements which may result into a privacy breach if violated. We clarify this using an example.

Example 4.3. Let cluster $C_1 = \{S_1\}$ and cluster $C_2 = \{S_3\}$ for the click streams of Figure 1(a). Assume that $E_r = \{\text{Google}\}$ and a domain-level constraint expressed as

[14] An alternative is to pick a pair of clusters with minimum pairwise distance. In our experiments, this approach increased running time but did not result in a remarkable improvement in information loss.

Algorithm 1: BASELINE (Dataset $\mathcal{D}$, Privacy model $\mathcal{M}$)

```
/* Step 0) verify if a solution exists for database $\mathcal{D}$ and privacy model $\mathcal{M}$ */
 1: if privacy model $\mathcal{M}$ is $k$-anonymity then
 2:     if $(|\mathcal{D}| \geq k)$ then set $\tau = k$ else return $\emptyset$          ▷ No anonymization exists
 3: if privacy model $\mathcal{M}$ is $(g, \ell)$-diversity then
 4:     if $\Pr(e, \mathcal{T}_g|\mathcal{D}) \geq \frac{1}{\ell}$ for an event $e \in E - E_r$ and time interval $\mathcal{T}_g$ shorter than $g$ then
 5:         return $\emptyset$          ▷ No anonymization exist
 6:     else set $\tau = \ell$
/* Step 1) find the list of clusters $\rightarrow \mathcal{P}$ */
 7: Set $C_i = \{S_i\}_{i=1}^{|\mathcal{D}|}$, set $C = \{C_i\}_{i=1}^{|\mathcal{D}|}$, and set $\mathcal{P} = \{\}$
 8: while $C$ is not empty do
 9:     Let $C_j = \mathrm{NN}(C_i, C, \mathcal{M}, \texttt{TRUE})$ for a random cluster $C_i \in C$
10:     if $C_j$ is empty then Remove $C_i$ from $C$ and add it to $\mathcal{P}$
11:     else
12:         Remove $C_j$ from $C$ and update $C_i = C_i \cup C_j$
13:         if $|C_i| \geq \tau$ then Remove $C_i$ from $C$ and add it to $\mathcal{P}$
/* Step 2) merging under-filled clusters */
14: for each cluster $C_i \in \mathcal{P}$ with less than $\tau$ sequences do
15:     repeat
16:         $C_j = \mathrm{NN}(C_i, \mathcal{P}, \mathcal{M}, \texttt{FALSE})$
17:         Remove $C_j$ from $\mathcal{P}$ and update $C_i = C_i \cup C_j$
18:     until $C_i$ satisfies the privacy model $\mathcal{M}$
19: return $\mathcal{P}$

20: function NN(cluster $C_i$, cluster set $C$, model $\mathcal{M}$, boolean verify)
21:     Set minDist = $\infty$ and $C_k = \{\}$
22:     for each cluster $C_j$ in $C$ do
23:         if verify=TRUE and $(C_i, C_j) \in Con_{\neq}$ then
24:             Skip cluster $C_j$          ▷ $C_i \cup C_j$ violates privacy model $\mathcal{M}$
25:         else if minDist $> CP(C_i \cup C_j)$ then
26:             Set $C_k = C_j$ and minDist = $CP(C_i \cup C_j)$
27:     return $C_k$
```

"a user visits only one URL at each time point." Merging the two clusters into $C_{1,2} = C_1 \cup C_2$ does not provide adequate privacy level for S_3; S_3 visited Google at time 3, an adversary can infer that the visit to Bing at time 3 in $C_{1,2}$ belongs to S_1.

4.2. The Algorithm

BASELINE, presented in Algorithm 1, regards all instance level constraints and makes a greedy decision to merge clusters based on the information loss incurred. First it examines the database to ensure that it can be anonymized to fulfill the desired privacy level (lines 1–6). Then, in the first step (Lines 7–13), each sequence is assigned to a single cluster inside clusters group C. While C is not empty, we pick a random cluster C_i from C. One of two possible cases can happen to C_i: if C_i violates an instance-level constraint with all other clusters in C, we move C_i to the set $\mathcal{P}$ to be processed later by a merge which ignores the instance-level constraints. Otherwise, we find a cluster $C_j \in C$ such that $CP(C_i \cup C_j)$ is minimum and C_j does not violate an instance-level constraint with C_i. This is in fact a nearest-neighbor search for C_i on C or $\mathrm{NN}(C_i, C, \mathcal{M}, \text{verify})$ for short; where verify indicates whether an instance-level constraint for privacy model $\mathcal{M}$ must be enforced. In the first step of BASELINE, the instance-level constraint is enforced. If merging C_i with C_j creates a cluster with at least ℓ sequences, we move the merged cluster to $\mathcal{P}$. Otherwise, we put the merged cluster back into C and continue until C becomes empty and we move to the second phase (Lines 14–19). At this point, the clusters in $\mathcal{P}$ with less than ℓ sequences (the under-filled clusters) definitely violate (g, ℓ)-diversity. We resolve the under-filled clusters by merging them in a way to minimize the increase in the total information loss. Each under-filled cluster C_i is iteratively merged with its nearest clusters in $\mathcal{P}$ until C_i meets (g, ℓ)-diversity. In this step, the instance-level constraint is not enforced to ensure convergence. The monotonicity of (g, ℓ)-diversity ensures that the second phase always finds a legitimate partitioning. If only grouping all sequences into a single group fulfills (g, ℓ)-diversity, this extreme grouping is found in the second phase.

4.2.1. Baseline for k-Anonymity. We need to replace the instance level constraint of (g, ℓ)-diversity with a proper definition. In k-anonymity, merging any two clusters creates a legitimate cluster which does not violate privacy. BASELINE achieves k-anonymity by setting verify=FALSE in Lines 9 and 16 of Algorithm 1. Because k-anonymity is monotone [Sweeney 2002], BASELINE always converges to a valid partitioning.

4.2.2. Time Complexity. Searching for the closest cluster to C_i is the bottleneck of BASELINE; an exhaustive search needs $O(|\mathcal{D}|)$ I/O costs for loading the (sequences to compute the information loss for merging with C_i. Thus, the total I/O complexity is $O(|\mathcal{D}|^2)$. However, computing information loss for a cluster is quadratic in the number of time points of the cluster and increases linearly with both the length of the shortest sequence and the height of taxonomy (Section 7 for a detailed analysis). As for CPU Cost, BASELINE needs $O(|\mathcal{D}|^2)$ computations of CP in either k-anonymity or (g, ℓ)-diversity.

5. OPTIMIZING BASELINE FOR THE SSR SETTING

In this section, we focus on the SSR setting which requires time generalization. Because NN is the bottleneck of BASELINE, we investigate two directions for performance improvement: (1) to speed up NN using an index, and (2) to reduce the number of NN calls using a heuristic. In the first direction, in Section 5.1, we propose summaries for clusters and fast-to-compute lower bounds for information loss. We then illustrate how the summaries and the lower bounds are used to boost NN in Section 5.2 using an index-based incremental pruning. In the second direction, in Section 5.3, we propose a heuristic method which uses the distribution of time points as a clue to (indirectly) reduce the number of NN calls. Because we consider the SSR setting in this section, only the time stamps of events monitored by the receiver(s) contribute to information loss. Thus, we extract summaries only from these time stamps. Naturally, for each sequence, the time stamps appear in ascending order, from the first (smallest) time stamp to the last (largest) one. The algorithms we propose in this section all take advantage of this order in connection with the information loss measure (i.e., CP). In Section 6, we use the techniques that we develop in this section as a module for the SCR setting.

5.1. Cluster Summaries and Lower Bounds

The direct approach to speed up NN in BASELINE is to reduce the number of times the information loss is computed. Naturally, in line 25 of Algorithm 1, if the lower bound of the information loss for $C_i \cup C_j$ is greater than minDist, evaluating the information loss for $C_i \cup C_j$ becomes redundant. In this section, we propose two fast-to-compute lower bounds; namely, D_{LB} and D_{fp}. The first one is more accurate but requires access to all sequences of the two clusters under comparison. The second lower bound relies on compact fingerprints that can be stored in main memory. Thus, besides reducing the number of information loss computations, it takes advantage of compact in-memory fingerprints to reduce the number disk accesses required for reading sequences of clusters to be pruned.

5.1.1. A Simple Lower Bound for Information Loss. Let C be a cluster of sequences. We want to find a lower bound for the information loss caused by anonymizing C using time generalization. First, assume that C has only two sequences S and R. For time stamp t and sequence R let $\mathsf{nearest}(t, R, E_r)$ be the time stamp of the closest event pair $(e_r, t_r) \in R$ to t such that $e_r \in E_r$. Intuitively, the distortion for time stamp t in S is not less than $|t - \mathsf{nearest}(t, R, E_r)|$. By induction, this idea can be applied when C has more than two sequences to find a lower bound of time generalization.

$$\mathsf{minLoss}(t, S, C) = \max_{R \,\in\, C - \{S\}} \big|t - \mathsf{nearest}(t, R, E_r)\big|,$$

where $C - \{S\}$ denotes all sequences in cluster C except for S. Aggregating $\mathsf{minLoss}$ over the time stamps of cluster C that are generalized yields our first lower bound.

LEMMA 5.1. *For event set E_r and cluster C, D_{LB} defined as*

$$D_{LB}(C) = \frac{\sum_{S \in C} \sum_{(e,t) \in S \wedge e \in E_r} \mathsf{minLoss}(t, S, C)}{\left(\max\limits_{S \in \mathcal{D}} \mathsf{end}(S) - \min\limits_{S \in \mathcal{D}} \mathsf{start}(S)\right) \cdot \sum\limits_{S \in C} |S|},$$

provides a lower bound for information loss $CP(C)$ under time generalization, in $O(n_s \cdot n_t)$ time, where n_s is the number of sequences in cluster C and n_t is the total number of (e, t) pairs in all sequences of C such that $e \in E_r$.

PROOF. (LOWER BOUNDING) For event pair (e, t) in sequence S, where $e \in E_r$, let $I_t = (c_t, [s_t - e_t])$ be the interval in the set $\mathcal{I}$ that covers time stamp t. Clearly, $(e_t - s_t) \geq \mathsf{minLoss}(t, S, C)$ by the definition of $\mathsf{minLoss}$. Also, there is no other interval that covers time stamp t due to the strong coverage property. Thus, replacing $\mathsf{minLoss}$ for IL_t provides a lower bound for IDD, for IL, and finally for CP.

(Time complexity) D_{LB} can be computed using a sliding window over $O(n_t)$ event pairs in C to be generalized. The window must keep two time stamps for each sequence, one which has been visited already and the next one to be visited. Because time stamps are ordered in each sequence, $\mathsf{minLoss}$ can be computed in $O(n_s)$ time for each time stamp by probing the next time stamp of n_s sequences. Thus, the overall computational complexity is $O(n_s \cdot n_t)$, linear in the number of time stamps to generalize. □

Example 5.2. Let $C_1 = \{S_1, S_2, S_3\}$ and $C_2 = \{S_4, S_5\}$ be two clusters of sequences in Figure 1(a). Let E_r = {Google}; only the time stamps of Google visits need generalization. We can verify that $\mathsf{minLoss}(1, S_1, C_1) = 2$ and $\mathsf{minLoss}(4, S_4, C_2) = 1$. The amount of data distortion for time points 1 and 4 are both equal to two since the former is generalized to the range [1–3] and the latter to [4–6] as shown in Figure 2(a). From the optimal time generalization for these two clusters shown in Figure 2(a), we can compute $CP(C_1) = \frac{4 \times 2 + 3 \times 1}{12 \times 14} = 0.065 = D_{LB}(C_1)$. Similarly, $CP(C_2) = \frac{3 \times 2}{12 \times 10} = 0.05$ but $D_{LB}(C_2) = \frac{3 \times 1}{12 \times 10} = 0.025$.

D_{LB} can be used effectively in BASELINE to reduce few (costly) information loss computations as follows. In Line 25 of Algorithm 1, we compute $CP(C_i \cup C_j)$ only if $D_{LB}(C_i \cup C_j)$ is less than $\mathsf{minDist}$. This pruning is effective but requires to read the sequences of C_j. Next, we propose our second lower bound which is designed to reduce disk access.

5.1.2. A Lower Bound for Information Loss Using Fingerprints. Each call to NN in BASELINE reads all clusters to find the nearest neighbor of C_i. We propose a compact structure, to store fingerprint of clusters in main memory and a lower bound for information loss. Our goal is to prune loading clusters and computing information loss (or D_{LB}).

Definition 5.3 (*Fingerprint*). A fingerprint is a bitmap which maps the time stamps of pairs (e, t) in sequences of a cluster, such that $e \in E_r$, to a binary bit-string with b buckets.

To derive the fingerprint of a cluster, we first divide the domain time span into b buckets of equal length. Let $|b|$ be the length of each time span. Let B_c be the fingerprint corresponding to cluster C. We use $B_c[r]$, $1 \leq r \leq b$, to refer to the bit at index r of B_c, we set this bit to one if there exists at least one pair (e, t) in any sequence of C such that $e \in E_r$ and $r \cdot |b| > t \geq (r-1) \cdot |b|$; otherwise, we set $B_c[r]$ to zero. At least one bit of each fingerprint must be set to one if the corresponding cluster is nonempty.

Example 5.4. Let $C = \{S_1, S_2, S_3\}$ be a cluster of sequences in Figure 1(a). Let $b = 12$ and E_r = {Google}. The domain time span is [1–13] and the length of each bucket is one and $B_c = 101111000110$; because there are three visits to Google in range [1–2), $B_c[1] = 1$ but there is no visit to Google during [2–3) in cluster C thus $B_c[2] = 0$.

Each bucket represents a time interval, and parameter b determines the *resolution* of each bucket in terms of the number and the distribution of time stamps that a bucket can capture. But each bucket does not capture frequency. An alternative and less compact representation, which we do not consider here, would store the frequency of time points in each bucket. We want to define a lower bound for information loss due to time generalization using the bucket that represents a cluster. The minimum generalization of each time stamp in a cluster can be estimated from the relative position of the bits that are set to one in the fingerprint corresponding to the cluster. For fingerprint B, we define the minimum generalization of the time stamps falling into bucket r as

$$\mathsf{mg}(B, r) = \begin{cases} 0, & \text{if } B[r] \text{ is set to one,} \\ \underset{1 \leq s \leq b\,,\, B[s]=1}{\arg\min} \; |r - s| - 1, & \text{otherwise.} \end{cases}$$

Intuitively, mg yields the number of bits set to zero between location r and the closest location s in B, which is also set to one. When $B[r]$ is set to one, the closest location to r is itself, and therefore the number of 0-bits is zero. The information loss of a time stamp mapped to location r can be estimated from mg and the bucket length as follows.

LEMMA 5.5. *Let B_1 and B_2 be, respectively, the fingerprints for clusters C_1 and C_2. For bucket $B[r]$, let $\overline{B[r]} = 1$ if the bucket is zero and $\overline{B[r]} = 0$ otherwise. D_{fp} defined as*

$$\frac{\sum_{r, B_1[r] \bigwedge \overline{B_2[r]}} \mathsf{mg}(B_2, r) + \sum_{r, \overline{B_1[r]} \bigwedge B_2[r]} \mathsf{mg}(B_1, r)}{b \;\cdot \sum_{S \in C_1 \cup C_2} |S|}$$

provides a lower bound for $CP(C_1 \cup C_2)$ in $O(b)$ time.

We give an example to motivate the idea, then we provide the proof of Lemma 5.5.

Example 5.6. In the click streams of Figure 1(a), let $C_1 = \{S_1\}$ and $C_2 = \{S_2, S_3\}$ be two clusters, E_r = {Google}, and $b = 12$. The fingerprints are $B_1 = 100000000100$ and $B_2 = 101000000010$, respectively. To find the information loss of $C_1 \cup C_2$, we must aggregate the generalizations of time stamps of Google visits. Because both clusters have Google visits at $t = 1$, the minimum generalization for this time stamp is zero. For $t = 3$, we find the closest time stamp to t in C_1, which is $t' = 1$. The minimum generalization is $|t - t'| = 2$. We estimate this difference using fingerprints. $B_1[1]$ is the closest bucket to $t = 3$ which is set to one. We use the number of empty buckets between $B_1[1]$ and $B_1[3]$, which is one, to estimate the minimum generalization as $1 \times (|b| = 1)$. For time points 10 and 11, the number of empty buckets between $B_1[10]$ and $B_2[11]$ is zero. Thus, we estimate the minimum time generalization for these time points as zero. In total, $D_{fp}(C_1 \cup C_2) = \frac{1}{12 \times 14} = 0.0059$ which is less than $CP(C_1 \cup C_2) = 0.065$.

PROOF. Let time stamp t_1 for an event pair in sequence $S \in C_1$ be mapped to bucket $B_1[r_1]$. Let time stamp t_2 be the closest time stamp to t_1 in sequence $R \in C_2$. We assume that the events corresponding to these time stamps are both in E_r. Let t_2 be mapped to bucket $B_2[r_2]$ and that there are $\mathsf{mg}(B_2, r_1)$ zero bits in B_2 between t_1 and the bucket of its closest time stamp. By the definition of minLoss,

$$\mathsf{minLoss}(t_1, R, C_{1,2}) = |t_2 - t_1| \geq |b| \cdot \mathsf{mg}(B_2, r_1), \tag{7}$$

where $C_{1,2} = C_1 \cup C_2$. To find the minimum information loss for generalizing t_2, we need to find the closest time stamp to t_2, which might be different from t_1 but still the same argument applies. If both $B_1[r]$ and $B_2[r]$ are set to one, the smallest generalization happens when $t_1 = t_2$ and thus $\mathsf{minLoss}(t_1, S, C_{1,2}) = \mathsf{minLoss}(t_2, R, C_{1,2}) = 0$. There are two possible cases when $r_1 = r_2 = r$. First, if both $B_1[r]$ and $B_2[r]$ are zero, there is not time stamp in these buckets to generalize. Second, when they are equal to one, the minimum generalization is zero. Only when the mapped buckets are different the information loss can be nonzero. Replacing $\mathsf{mg}(.)$ in the definition of D_{fp} with its upper bound from the left hand side of Eq. (7) produces an upper bound for $D_{fp}(B_1, B_2)$.

$$\frac{\sum_{S \in C_{1,2}} \sum_{(e,t) \in S \wedge e \in E_r} \mathsf{minLoss}(t, S, C_{1,2})}{b \ \cdot \ |b| \ \cdot \sum_{S \in C_{1,2}} |S|} = D_{LB}(C_{1,2}),$$

because $b \cdot |b| = \max_{S \in \mathcal{D}} \mathsf{end}(S) - \min_{S \in \mathcal{D}} \mathsf{start}(S)$. Since D_{LB} is a lower bound of CP, and D_{fp} is a lower bound for CP by transitivity. D_{fp} can be computed by one pass over the two fingerprints, thus $O(b)$ time complexity. □

Fingerprints can be stored in the main memory. To use the fingerprints and D_{fp} in BASELINE, Line 25 of Algorithm 1 needs to be modified as follows; we read sequences of cluster C_j and compute CP only if $D_{fp}(C_i, C_j) \leq \mathsf{minDist}$. Meanwhile, we can benefit from pruning extra CP computations using D_{LB}. In the next section, we propose an effective technique to prune a *group* of clusters using an index.

5.2. Indexing Clusters for Effective Pruning

One approach to speed up nearest-neighbor search in BASELINE is to index clusters. However, the information loss measure does not satisfy the triangle inequality. This is mainly due to two factors: (1) the constraint imposed by the strong coverage (SC) (Property 2.3), and (2) multiplying IDD in Eq. (3) by $|S_I|$. Unfortunately, relaxing the SC property by exempting a subset of sequences would have implications on either sequence identifications or publishing truthful data, whereas the term $|S_I|$ is important to make IL, and consequently CP, sensitive to not only the information loss of each interval per sequence, but also the number of (event, time) pairs blurred by each interval. Note that other utility-based partitioning algorithms (e.g., [Nergiz et al. 2009]) also have a nonmetric information loss measure. In this case, we cannot take advantage of algorithms for similarity search in metric space (e.g., [Ciaccia et al. 1997]) for the sake of performance. Instead, we extend the multistep nearest-neighbor search framework of Seidl and Kriegel [1998] to index clusters. First, we propose a new summary for clusters, which we term *cluster extent*. The summary records high-level information regarding the distribution of time points in the sequences of each cluster.

Then, we derive a proper lower bounds of information loss. Finally, we integrate all summaries and lower bounds that we have proposed by far in order to organize clusters inside an index structure and to support nearest neighbor search by incremental pruning.

5.2.1. Cluster Extent - A Summary for a Group of Clusters. A cluster extent extracts four dimensions from a set of clusters and represent them using a minimum bounding rectangle (MBR). The extent summarizes the distribution of time points in sequences of a cluster using four dimensions.

- s. The earliest time that any event in E_r is visited.
- e. The latest time that any event in E_r is visited.
- c. The total number of events visited in the cluster.
- l. The number of events visited after the latest start time and before the earliest end time by all sequences in the cluster.

For the cluster extent corresponding to a group of clusters, we keep the minimum and the maximum of each dimension in the extent. We use subindexes to refer to the minimum and the maximum of each dimension along all clusters represented by the extent. This way, s_1 (s_2) refers to the minimum (maximum) of earliest time along a set of clusters, e_1 (e_2) refers to the minimum (maximum) of latest time along a set of clusters, and so on. If an extent contains only one cluster, the minimum and the maximum of each dimension becomes the same.

Example 5.7. Consider Figure 1(a) and let E_r = {Google}. Consider two clusters $C_1 = \{S_1, S_2, S_3\}$ and $C_2 = \{S_4, S_5\}$. Then, for an extent that represents both C_1 and C_2.

- $\mathsf{s}_1 = 1$, $\mathsf{s}_2 = 5$. The minimum and maximum earliest visit to Google happen, respectively, in sequences $S_1 \in C_1$ and $S_5 \in C_2$.
- $\mathsf{e}_1 = 5$, $\mathsf{e}_2 = 11$. The minimum and maximum latest visit to Google happen, respectively, in sequences $S_5 \in C_1$ and $S_2 \in C_2$.
- $\mathsf{c}_1 = 3$, $\mathsf{c}_2 = 7$. The minimum and maximum number of visits to Google happen, respectively, in clusters C_2 and C_1.
- $\mathsf{l}_1 = 0$, $\mathsf{l}_2 = 1$. The minimum and maximum is computed for two clusters C_1 and C_2. For C_1, the number of visits to Google after its latest start time (i.e., 1) and before its earliest end time (i.e., 10) is 1. For C_2, this number is zero because both latest start visit to Google and earliest last visit to Google happen at the same time (i.e., 5).

In comparison with fingerprints, a cluster extent stores the *boundaries* of a cluster along with the number of time stamps in the range $[\mathsf{s}_2, \mathsf{e}_1]$. An interesting property of cluster extents is that they can be used for *batch pruning*; to skip information loss computation for a group of clusters represented by an extent. For this purpose, we next define a distance measure between a cluster and an extent. Then, we use this property to index clusters in Section 5.2.3.

5.2.2. A Lower Bound for Information Loss Using Extents. Let C_i be a cluster and the set of clusters C be represented by the extent $\big((\mathsf{s}_1, \mathsf{s}_2), (\mathsf{e}_1, \mathsf{e}_2), (\mathsf{c}_1, \mathsf{c}_2), (\mathsf{l}_1, \mathsf{l}_2)\big)$. In this section, we derive $D_{Extent}(C_i, \mathsf{C})$ and show that it lower bounds $CP(C_i \cup C_j)$ for any $C_j \in \mathsf{C}$. We define D_{Extent} based on the overlap of the range of time stamps of C_i with the time span of the extent of C. We consider two possible cases.

(1) No overlap if $\mathsf{l}_2 = 0$. Finding the minimum generalization of each time stamp in C_i and any cluster covered by the extent is straightforward. Let $\mathsf{start}(C)$ and $\mathsf{end}(C)$ denote the time stamp for the first and the last visit of a sequence in cluster C to an event in E_r. If $\mathsf{end}(C_i) \leq \mathsf{s}_1$, since each time stamp in C_i must be included in one interval that

covers a time stamp of a sequence in $C_j \in \mathsf{C}$, because of the strong coverage property, then the information loss of each time stamp in C_i is at least s_1-$\mathsf{start}(C_i)$ when all time stamps of C_j are at $\mathsf{start}(C_j)$. But the last time stamp of C_j is not before e_1. Thus, the smallest interval which covers all time points of C_i and C_j is the range [$\mathsf{start}(C_i)$, e_1]. By symmetry, a similar argument applies to the case where $\mathsf{start}(C_i) \geq \mathsf{e}_2$, in which case, the smallest interval is [s_2, $\mathsf{end}(C_i)$].

LEMMA 5.8. *For a set of clusters* C *represented by the extent* $\big((\mathsf{s}_1,\mathsf{s}_2),(\mathsf{e}_1,\mathsf{e}_2),(\mathsf{c}_1,\mathsf{c}_2),(\mathsf{l}_1,\mathsf{l}_2)\big)$ *and cluster* C_i*, when* $\mathsf{l}_2 = 0$,

$$D_{Extent}(C_i,\mathsf{C}) = \frac{\mathsf{minLoss}}{\max_{S\in\mathcal{D}} \mathsf{end}(S) - \min_{S\in\mathcal{D}} \mathsf{start}(S)}$$

provides a lower bound for $CP(C_i \cup C_j)$ *for any cluster* $C_j \in \mathsf{C}$ *in constant time, where* $\mathsf{minLoss}$ *is defined as*

$$\mathsf{minLoss} = \begin{cases} \mathsf{e}_1 - \mathsf{start}(C_i), & \mathsf{end}(C_i) \leq \mathsf{s}_1, \\ \mathsf{end}(C_i) - \mathsf{s}_2, & \mathsf{start}(C_i) \geq \mathsf{e}_2. \end{cases}$$

PROOF. We consider the case where $\mathsf{end}(C_i) \leq \mathsf{s}_1$; the other case is symmetric. All time stamps in $C_i \cup C_j$ are generalized to range [$\mathsf{start}(C_i)$, e_1]. The distortion of each time stamp is at least $\mathsf{minLoss}$ and at most $\mathsf{e}_2 - \mathsf{start}(C_i)$. D_{Extent} uses the minimum value for all points, and the rest is normalization by global time span. When $\mathsf{l}_2 = 0$, $D_{Extent}(C_i,\mathsf{C})$ can be computed in constant time. □

(2) Possible overlap if $\mathsf{l}_1 > 0$. For this case, we identify five ranges, two of which have a minimum of zero time generalization, while the other three ranges may produce nonzero distortion. Thus, we aggregate the loss over those three ranges to derive a lower bound for information loss. Before introducing the ranges, recall that for any time stamp t, the notation $\mathsf{nearest}(t,S,E_r)$ stands for the time stamp of a visit to an event in E_r by sequence S. We identify five ranges from the extent $\big((\mathsf{s}_1,\mathsf{s}_2),(\mathsf{e}_1,\mathsf{e}_2),(\mathsf{c}_1,\mathsf{c}_2),(\mathsf{l}_1,\mathsf{l}_2)\big)$, namely, $\mathsf{range}_0 = [-\infty,\mathsf{s}_1]$, $\mathsf{range}_1 = [\mathsf{s}_1,\mathsf{s}_2]$, $\mathsf{range}_2 = [\mathsf{s}_2,\mathsf{e}_1]$, $\mathsf{range}_3 = [\mathsf{e}_1,\mathsf{e}_2]$, and $\mathsf{range}_4 = [\mathsf{e}_2,+\infty]$. If timestamp t in a sequence of cluster C_i is in range_1, then $\mathsf{nearest}(t,S,E_r)$ could be the same as t for sequence S inside cluster $C_j \in \mathsf{C}$. This means that the minimum information loss in this range could be zero. The same argument holds for range_3. The ranges with possibly nonzero time generalizations are range_0, range_2, and range_4; we next derive the minimum total time generalization as, respectively, $\mathsf{minLoss}_0$, $\mathsf{minLoss}_2$, and $\mathsf{minLoss}_4$. In range_0, intuitively, every time stamp of C_i for events in E_r, will be generalized to s_1. Therefore,

$$\mathsf{minLoss}_0 = \big|\{t \mid t \in S \in C_i, t < \mathsf{s}_1\}\big| \cdot \big(\mathsf{s}_1 - \mathsf{start}(C_i)\big),$$

and by symmetry, the minimum total loss for range_4 is

$$\mathsf{minLoss}_4 = \big|\{t \mid t \in S \in C_i, t > \mathsf{e}_2\}\big| \cdot \big(\mathsf{end}(C_i) - \mathsf{e}_2\big).$$

In range_2, there are two possibilities for the overlap of timestamps of visit to E_r for cluster C_i and any cluster C_j in C. When the overlap is not empty (i.e., $\mathsf{e}_1 = 0$), the minimum loss is zero in the worst case. Otherwise, there is a nonzero information loss. However, the loss depends on the distribution of the time stamps relative to s_2 and e_1. Let $T^{\cap}_{i,2}$ denote the set of time stamp of cluster C_i (for visits to E_r) which falls in range range_2 if $\mathsf{s}_2 < \mathsf{e}_1$. Intuitively, a fraction of the time stamps in $T^{\cap}_{i,2}$ will be generalized to s_2, and the rest will be generalized to e_1. The fractions depend on the closeness of the time stamps in $T^{\cap}_{i,2}$ to these two end points. Because our goal is to find a lower bound for $\mathsf{minLoss}_2$, we take a conservative approach and find the distortion which corresponds to minimum possible information loss. For this purpose, for time stamps $t_r \in T^{\cap}_{i,2}$, we first define the amount of losses before and after t_r as follows.

$$\begin{aligned}\mathsf{minLoss}^b_2(t_r) &= |\{t \mid t \in T^{\cap}_{i,2} \wedge t \leq t_r|\} \cdot (t_r - \mathsf{s}_2),\\ \mathsf{minLoss}^a_2(t_r) &= |\{t \mid t \in T^{\cap}_{i,2} \wedge t \geq t_r|\} \cdot (\mathsf{e}_1 - t_r).\end{aligned}$$

Intuitively, each loss multiplies minimum loss by the number of generalized timestamp. With these two losses, we set $\mathsf{minLoss}_2 = 0$ if $\mathsf{e}_1 = 0$, otherwise,

$$\mathsf{minLoss}_2 = \min_{t_r \in T^{\cap}_{i,2}} \left\{\mathsf{minLoss}^b_2(t_r) + \mathsf{minLoss}^a_2(t_{r+1})\right\},$$

assuming that the set $T^{\cap}_{i,2}$ is sorted. Using the minimum time generalization for all ranges, we can finally define the lower bound of information loss for $\mathsf{l}_2 > 0$ case.

LEMMA 5.9. *For a set of clusters* C *represented by the extent* $\big((\mathsf{s}_1, \mathsf{s}_2), (\mathsf{e}_1, \mathsf{e}_2), (\mathsf{c}_1, \mathsf{c}_2), (\mathsf{l}_1, \mathsf{l}_2)\big)$ *and cluster* C_i *if* $\mathsf{l}_2 > 0$*, then* $D_{Extent}(C_i, \mathsf{C})$ *defined as*

$$\frac{\mathsf{minLoss}_0 + \mathsf{minLoss}_2 + \mathsf{minLoss}_4}{\big(\max\limits_{S \in \mathcal{D}} \mathsf{end}(S) - \min\limits_{S \in \mathcal{D}} \mathsf{start}(S)\big)\big(\mathsf{c}_2 + \sum\limits_{S \in C_i} |S|\big)} \tag{8}$$

lower bounds $CP(C_i \cup C_j)$ *for any cluster* $C_j \in \mathsf{C}$.

PROOF. We have already shown that the numerator of Eq. (8) is less than the total distortion due to time generalization (before normalization). Two terms appear in the denominator of Eq. (8). The first term (left) is to normalize time generalization. The second term (right) is an upper bound of the number of time stamps in $C_i \cup C_j$, since $\sum_{S \in C_i \cup C_j} |S| \leq \big(\mathsf{c}_2 + \sum_{S \in C_i} |S|\big)$. Replacing the numerator and the denominator of Eq. (3) with, respectively, the lower bound of total loss and upper bound of number of time points establishes the proof. $\mathsf{minLoss}_2$ can be computed in $O(n_i)$ time, where n_i is the number of events in E_r for cluster C_i. Both $\mathsf{minLoss}_0$ and $\mathsf{minLoss}_4$ can be computed in constant time. Thus, the total time complexity D_{Extent} is $O(n_i)$. □

A key property of cluster extents is that they can be organized using an *index* for incremental batch pruning. Next, we use extents and the lower bounds and summaries we proposed in Section 5.1 in a single algorithm to speed up NN search in BASELINE.

5.2.3. Indexing Clusters to Boost NN. We organize clusters using an R-Tree [Guttman 1984]. The internal nodes of the tree are cluster extents. The cluster extent for the root node represents the extent for all clusters organized by the index. A leaf node keeps the extents of clusters along with the list of identifiers of clusters covered by that node. We store compact cluster fingerprints in the main memory. We observed a negligible space overhead for fingerprints in our experiments (see Section 8.4.4). The overhead can be tuned to available memory by adjusting parameter b.

We extend the multistep nearest-neighbor search [Seidl and Kriegel 1998] as FASTNN. We maintain objects (clusters or summaries) using a priority queue. The priority of an object is determined by the information loss or the lower bound computed for the object when it is pushed into the queue. Until we find the closest cluster to a query, or the queue gets empty, we keep fetching objects from the queue. If the fetched object is a summary, we enqueue every object covered by the summary along with their distance. FASTNN is presented in Algorithm 2. A cluster C_i is provided as query and the goal is to find the closest cluster to C_i. It initializes an empty heap and

Algorithm 2: FASTNN(cluster C_i, cluster index R)

```
Initialize an empty heap $\mathcal{H}$
Push $(\mathcal{H}, \text{root}, D_{Extent}(C_i, \text{root}), \texttt{Extent})$    ▷ root of the index
while $\mathcal{H}$ is not empty do
    Pop an object from $\mathcal{H}$ into $p$    ▷ Pruning by clusters' start/end points
    if type of object $p$ is Extent then
        for each entry $q$ in cluster extent $p$ do
            if $p$ is a leaf node then Push$(\mathcal{H}, q, D_{fp}(C_i, q), \texttt{FP})$
            else Push $(\mathcal{H}, q, D_{Extent}(C_i, q), \texttt{Extent})$
    else if type of object $p$ is FP then    ▷ Pruning by clusters' internal time-points
        Load time sequences for cluster $p$ into $C_p$
        if $(C_i, C_p) \notin Con_{\neq}$ then
            Push $(\mathcal{H}, C_p, D_{LB}(C_i, C_p), \texttt{LB})$    ▷ More refined pruning than Extent and FP
    else if cluster $p$ is de-heaped for the $1^{st}$ time then
        Push $(\mathcal{H}, p,\ CP(C_i \cup p),\ \texttt{CP})$
    else return $p$    ▷ cluster $p$ is de-heaped for the $2^{nd}$ time
return { }    ▷ $(C_i, C_k) \in Con_{\neq}$ for all clusters $C_k$
```

pushes the extent of the root node of index with $D_{Extent}(C_i, \text{root})$ as distance. While the heap is not empty, an object p is deheaped to be processed.

- If p is a cluster extent, we push to the heap every cluster extent q covered by the node p with $D_{Extent}(C_i, q)$.
- If p is a leaf node, we push to the heap every fingerprint q covered by the node p with $D_{fp}(p, q)$.
- If p is a fingerprint, we load the sequences of cluster p into C_p. If merging C_i and C_p does not violate a privacy constraint, we push to the heap C_p with $D_{LB}(C_i \cup C_p)$.
- If p is deheaped as a cluster for the first time, we push it back to the heap, this time with $CP(C_i \cup p)$ as distance.
- If this is the second time that p is deheaped as an object of type cluster, it is returned as the closest cluster to C_i.

FASTNN can speed up BASELINE by reducing the number of computations of information loss inside NN in Algorithm 1 from $O(|C|)$ to $O(\log |C|)$. We should mention that FASTNN benefits from pruning at three resolutions: (1) start and end time points of clusters via extents and D_{Extent}, (2) *blurred* internal time points of clusters via fingerprints and D_{fp}, and (3) internal time point of clusters via D_{LB}. Next, we take a data oriented approach aiming at reducing the number of NN invocations in BASELINE.

5.3. Hybrid Partitioning to Reduce NN Calls

Indexing improves BASELINE by boosting NN search. We take an orthogonal approach in this section and propose a data-driven heuristic to reduce the number of NN calls. This approach trades quality for performance. However, we did not observe a significant increase in total information loss in our experiments (Section 8.4). The main idea, demonstrated in Algorithm 3, is to partition data into two independent regions, namely, region A and region B. We apply a fast partitioning method on region A and apply BASELINE (with FASTNN) on region B. We then integrate the resulting partitions to form anonymization groups. We need two properties to ensure that the utility of this approach, the HYBRID method in Algorithm 3, stays very close to BASELINE. (1) For each region, time sequences must be grouped with the time sequences in the same region, when either BASELINE or HYBRID is applied. (2) The fast partitioning method, used in region A, must produce anonymization groups with the same quality BASELINE.

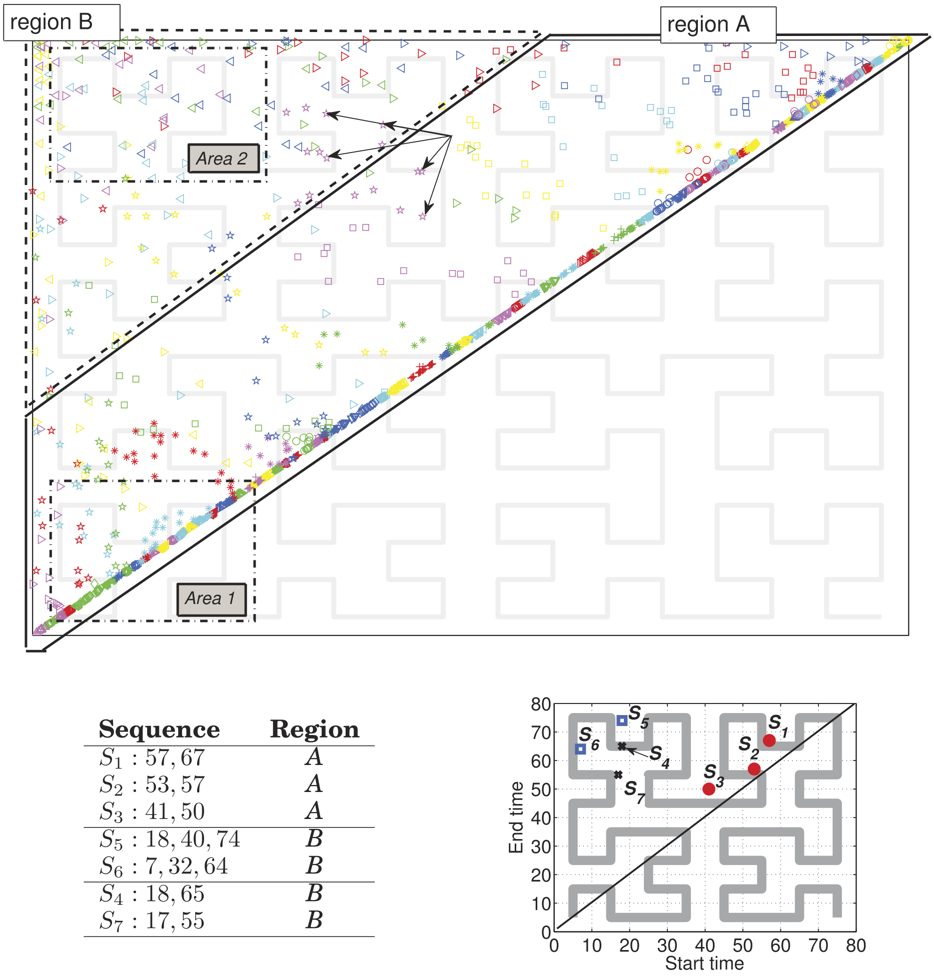
Our approach is demonstrated in Algorithm 3 and is based on observations on real datasets, as described in Section 5.3.1. We use Figure 3 (top) to motivate the main idea. In this figure, we use BASELINE to assign 1,000 time sequences into anonymization groups, when $w_e = 0$. In this figure, each time sequence is shown as a point in 2D space, the horizontal and vertical axis corresponds to, respectively, the first and the last time stamp of each sequence. For instance, time sequence S : 1,2,7,12 will be mapped to (1,12) in this space. We assigned the same color and marker to time sequences that fall in the same group. There are two regions we observe in Figure 3 (top). Region A is the area which is close to the diagonal line and region B is the area close to the upper-left corner (we describe how to assign time sequences to regions later). From Figure 3 (top), we observed that often, sequences in region A (region B) are assigned to groups with mostly sequences in the same region. One can verify this, for instance, by considering the sequences in Area 1 and Area 2 which belong to, respectively, region A and region B. Another property holds between the time sequences in region A and the Hilbert index assigned to each sequence (described later). We explain this property using a simpler example in Figure 3 with seven time sequences and two anonymization groups.

5.3.1. Data Partitioning in Hilbert Space. Figure 3 shows a mapping of time sequences into points in 2D space (recall that our focus is the SSR setting and time generalization). The coordinates correspond to the first and the last time stamp of each sequence. A Hilbert index (hIndex) is assigned to each sequence as follows: the space is divided into cells using a regular grid. The index of each cell on the Hilbert space filling curve [Moon et al. 2001] is assigned as hIndex(S) for any sequence S that has its end timestamps in the cell. A 2-anonymous grouping of sequences using BASELINE is shown in Figure 3 (left). We refer to the area close to the diagonal line in the 2D space as *region A* and the top-left corner of the 2D space as *region B*. We describe shortly how to assign sequences to regions. We made two observations through experiments on real and synthetic data.

OBSERVATION 5.10. *For sequences in* region *A (e.g., S_1, S_2, and S_3 in Figure 3), the difference between* hIndex *of two sequences is correlated with the information loss of the corresponding anonymization group. This is intuitive, because sequences in region A tend to have close start and end time points and a relatively small variance in time point values. As we get into region B, the end points of sequences get far apart, the variance of sequence length grows, and the diversity of time stamps increases. No consistent correlation is observed between the closeness of* hIndex *of two sequences and the information loss in region B. E.g.,* |hIndex(S_4)-hIndex(S_5)| < |hIndex(S_4)-hIndex(S_7)|, *but S_4 is grouped with S_7. Therefore, we can use a partitioning algorithm based on Hilbert index (and not* BASELINE *on complete time sequences) to efficiently and accurately assign the sequences in region A into anonymization groups.*

OBSERVATION 5.11. *A majority of sequences in region A are clustered with other sequences that are in the same region. Thus, an ordering of sequences in region A based on* hIndex *should bring closer all sequences that fall in the same cluster. Furthermore, partitioning sequences in region A independently from those time sequences in region B is not expected to significantly degrade the quality of produced anonymization groups.*

We propose HYBRID (Algorithm 3); it runs BASELINE (with FASTNN) for sequences in region B and cluster region A using a fast heuristic [Ghinita et al. 2007]. The key advantage of HYBRID is to avoid running BASELINE for time sequences that can be anonymized using an efficient technique which only considers the Hilbert index of sequences. In this regard, only short sequences actually favor from HYBRID. For long



Sequence	Region
S_1 : 57, 67	A
S_2 : 53, 57	A
S_3 : 41, 50	A
S_5 : 18, 40, 74	B
S_6 : 7, 32, 64	B
S_4 : 18, 65	B
S_7 : 17, 55	B

Fig. 3. Mapping sequences to 2D (start, end) space.

Algorithm 3: HYBRID(Dataset $\mathcal{D}$, Privacy model $\mathcal{M}$)

```
$(\mathcal{D}_A, \mathcal{D}_B)$ = ASSIGNREGIONS($\mathcal{D}$)                ▷ (Sec. 5.3)
$\mathcal{P}_A$ = FASTCLUSTER($\mathcal{D}_A$)                ▷ Using E.g. [Ghinita et al. 2007]
for any cluster $C$ in $\mathcal{P}_A$ that violates model $\mathcal{M}$ do
    Remove $C$ from $\mathcal{P}_A$ and add all its sequences to $\mathcal{D}_B$
$\mathcal{P}_B$ = BASELINE($\mathcal{D}_B$, $\mathcal{M}$)                ▷ Algorithm 1
return $\mathcal{P}_A \cup \mathcal{P}_B$                ▷ Merge $\mathcal{P}_A$ and $\mathcal{P}_B$
```

sequences, however, HYBRID directly inherits the performance improvements of BASELINE using FASTNN. We next describe two main components of HYBRID.

5.3.2. ASSIGNREGIONS *- Assigning Sequences to Regions.* We first sort sequences by their hIndex. For two consecutive sequences S_{i-1} and S_i in the ordered list, we compute the information loss of cluster $\{S_{i-1}, S_i\}$ as $CP(\{S_{i-1}, S_i\})$, and for S_1, we compute $CP(\{S_1, S_2\})$. We assign sequence S_i to region A if $CP(\{S_{i-1}, S_i\}) < \sum_{i=1}^{\mathcal{D}} \frac{CP(\{S_{i-1},S_i\})}{\mathcal{D}}$ or

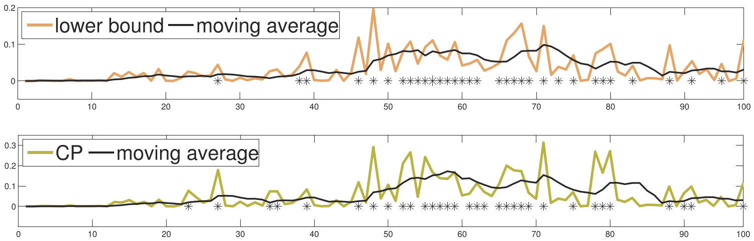


Fig. 4. Pairwise D_{LB} and CP for 100 time sequences sorted by their Hilbert index, the sequences assigned to region B are marked by *.

$|S_i| = 1$. Otherwise, S_i is assigned to region B. This heuristic needs $O(|\mathcal{D}|)$ evaluations of information loss, which is computationally expensive specially when the average length of sequences is large. Figure 4 shows CP and D_{LB} evaluated for 100 sequences in a real dataset, sorted by hIndex. The sequences which are assigned to region B are marked by star. We observe that not only CP is correlated with D_{LB}, but also the region assignment based on D_{LB} in Figure 4 seems to be correlated with the region assignment based on CP. Thus, we can use D_{LB} to speed up region assignment.

5.3.3. FASTCLUSTER - *Clustering Sequences in Region A.* We need to find a grouping of sequences such that each group has at least τ and at most $2\tau - 1$ members and the information loss for each group is minimized. Recall that $\tau = k$ in k-anonymity and $\tau = \ell$ in (g, ℓ)-diversity. Due to the correlation observed between the difference in the hIndex of sequences and the information loss in region A, we use a dynamic programming algorithm [Ghinita et al. 2007] to find an optimal grouping in region A in $O(\tau \cdot |\mathcal{D}_A|)$ time, where $\mathcal{D}_A$ is the sequences of $\mathcal{D}$ assigned to region A. In k-anonymity, all clusters found by dynamic programming can be finalized unless $|\mathcal{D}_A| < k$. In this case, we process all sequences of $\mathcal{D}_A$ with the sequences in region B ($\mathcal{D}_B$). Since during dynamic programming of [Ghinita et al. 2007], we only check the QID part of sequences, and not the sensitive part, (g, ℓ)-diversity may not hold for some clusters. For each cluster, the violation can be checked by moving a sliding window over all time and event pairs in the cluster. For each window there must be only one participation for each event type. This probe can be done in $O(g \cdot n_e)$ time, where n_e is the number of event pairs of the cluster under examination. If the cluster satisfies (g, ℓ)-diversity, it is finalized. Otherwise, the sequences of this cluster will be reassigned to region B to be handled with sequences in $\mathcal{B}$ by BASELINE (with FASTNN).

6. ANONYMIZATION IN THE SCR SETTING

Our algorithms in Section 5 focused on the SSR setting and time generalization. Our optimizations in Section 5 took advantage of the natural order between the time stamps of each sequence. In the SCR setting, sequences available to an adversary contain (event, time) pairs. In each sequence, time stamps are still sorted but events (i.e., URLs) are not visited by any prespecified order. The optimizations for the SSR setting can be used in the SCR setting when $w_e = 0$; in this case, any generalization of events (i.e., URLs) is acceptable. However, when $w_t > 0$ and $w_e > 0$, our optimizations for the SSR setting cannot be directly extended, because events in each sequence are not ordered.

In Section 6.1, for the SCR setting and $w_e = w_t$, we propose a top-down partitioning algorithm (PR) based on the construction step of kd-trees [Friedman et al. 1977]. PR is a round-robin divide and conquer method which recursively partitions data to reduce information loss along either the time dimension or the event dimension. For performance improvement, we use optimizations for the SSR setting (Section 5) as a

Algorithm 4: PR(Dataset $\mathcal{D}$, Privacy model $\mathcal{M}$)

```
1: Set $\tau$ to $k$ if $\mathcal{M}$ is $k$-anonymity and to $\ell$ if $\mathcal{M}$ is $(g, \ell)$-diversity
2: Set $\mathbb{C}$ to one cluster containing all sequences of $\mathcal{D}$
3: $\mathcal{P}$ = PARTITION( $\mathbb{C}$, $\tau$ )                    ▷ start the recursive drill down
/* resolve privacy violations of clusters */
4: for each cluster $C_i \in \mathcal{P}$ that violates privacy model $\mathcal{M}$ do
5:     repeat
6:         Find cluster $C_j \in \mathcal{P}$ with smallest $CP(C_i \cup C_j)$
7:         Remove $C_j$ from $\mathcal{P}$ and merge it into $C_i$
8:     until $C_i$ satisfies privacy constraints
9: return $\mathcal{P}$

10: function PARTITION( cluster $C$, $\tau$)
11:     if $C$ has less than $2\tau$ sequences then return $\{C\}$
12:     Pick a random sequence $S_1$ from $C$
13:     Find sequence $S_2 \in C$ with the largest distance from $S_1$ ▷ distance: in terms of eventDist
14:     Set cluster $C_1 = \{S_1\}$ and cluster $C_2 = \{S_2\}$
15:     for each sequence $S_i$ in $C$ do
16:         if $S_i$ is closer to $S_1$ than to $S_2$ then add $S_i$ to $C_1$     ▷ closer: in terms of eventDist
17:         else Add $S_i$ to $C_2$
18:     return REFINE($C_1$, $\tau$) ∪ REFINE($C_2$, $\tau$)

19: function REFINE( cluster $C$, $\tau$)
20:     if $C$ has less than $2\tau$ sequences then return $\{C\}$
21:     Set $\kappa = \lfloor |C|/2 \rfloor$
22:     $\{C_1, C_2\}$ = BASELINE( $C$, $\kappa$-anonymity )          ▷ use FASTNN or HYBRID to speed up
23:     return PARTITION($C_1$, $\tau$) ∪ PARTITION($C_2$, $\tau$)
```

module in PR while reducing the distortion along the time dimension. We introduce a fast-to-compute distance in Section 6.2 to efficiently partition data along the event dimension. In Section 6.3 we propose an extension of PR for more general SCR setting where $w_e \neq w_t$.

6.1. Top-Down Partition and Refinement

We give an overview of our approach (Algorithm 4). First, all sequences are assigned to a single cluster. This cluster is passed to PARTITION, which consequently triggers a series of recursive calls. (1) PARTITION splits the cluster into two smaller ones based on the similarity of events (described in Section 6.2), ignoring the time stamps. Each of the split clusters is then passed for further processing to REFINE. (2) REFINE splits each cluster into two smaller ones based on the similarity of time stamps, ignoring their events. Then, it calls PARTITION for each split cluster. We stop further partitioning (refinement) of a cluster if it has less than 2τ sequences, where $\tau = k$ in k-anonymity and ℓ in (g, ℓ)-diversity. In this case, we add the cluster to the set of clusters waiting to be finalized. We use the same heuristic as the second step of BASELINE to resolve the privacy violations of clusters, if any, by merge to finalize anonymization groups.

Intuitively, REFINE reduces information loss along the time dimension only. To achieve this goal, we use the BASELINE algorithm and the optimizations for the SSR setting. REFINE uses the time stamps to split clusters. To split cluster C into smaller clusters, we use a heuristic; we set $\kappa = \lfloor \frac{|C|}{2} \rfloor$ and use BASELINE (with FASTNN) to find κ-anonymous partitions (line 22 of Algorithm 4). The main benefit is that the number of distance computations for this step reduces from $O(|C|^2)$ to $O(|C| \cdot \log |C|)$. We then pass each smaller cluster for further refinements to PARTITION. PARTITION focuses on the event dimension. It splits a cluster C into two smaller ones C_1 and C_2 as follows. First, it picks a random sequence $S_1 \in C$ and finds sequence $S_2 \in C$ with the largest distance to S_1. Every sequence in C which is closer to S_1 (to S_2) is inserted into C_1 (to C_2). As a distance metric, we can use CP with $w_t = 0$ (our focus is event similarity). However, CP is expensive to compute (Section 7). Thus, we propose a fast to compute alternative which regards event taxonomy (Section 6.2). We invoke PARTITION or REFINE for each cluster only if it has at least 2τ sequences. At the end of partition and refinement, we examine each cluster regarding the desired privacy model. We use the same heuristic as the second step of BASELINE to resolve privacy violations.

6.2. eventDist - A Fast-to-Compute Taxonomy-Based Distance

In this section, we propose eventDist as a fast-to-compute estimation for information loss of a cluster along the event dimension. eventDist will be used to measure distance and closeness in lines 13 and 16 of Algorithm 4, respectively. To formally define eventDist, we first transform sequences into multisets and measure the cost of matching multisets.

6.2.1. From Sequences to Multisets. We modify each sequence by removing the order of events. This produces a summary from the sequence which captures the diversity of events with respect to domain taxonomy, and the frequency distribution of events. Formally, from sequence S, we derive the multiset $M_S = \{(e_i, f_i)\}_{i=1}^{|M_S|}$, where $e_i \in E_r$ and $f_i = |\{t|(e_i, t) \in S\}|$ is the frequency of the appearance of event e_i in sequence S. For instance, the multiset of sequence S_1 in Figure 1(a) is {(Google,2), (Bing,1), (Amazon,2)}. To measure the similarity of two multisets, one option is to use measures proposed in the literature for sets (e.g., the Jaccard coefficient and the cosine measure). However, these measures ignore the semantics of an application domain modeled by a taxonomy. For instance, consider two multisets M_1 = {(Facebook,1), (Myspace,1), (Google,1), (eBay,1), (Seenit,1)} and M_2 = {(Bing,1), (Facebook,2),(Amazon,1)}. The similarity of M_1 and M_2 must capture not only the number of common events (i.e., Facebook) but also the number of events under the same category (i.e., Google and Bing which are both under category *Search Engines* in the taxonomy of Figure 2(c)). Therefore, motivated by edit distance, which is defined to compare two strings, we propose eventDist. As a distance measure between multisets, eventDist computes the minimum distortion caused by using the categories in the taxonomy to make the events of the two multisets equal.

Notations. Let $n(M, c_i)$ be the sum of the frequency of events in multiset M that are covered by category c_i. We set $n(M, c_i) = \infty$ if M has no event under category c_i. Let $|c_i|$ be the number of events under category c_i in the taxonomy.

Definition 6.1 (*Cost of Match*). The cost of matching multisets A and B, given the set of categories $\mathbb{C} = \{c_i\}_{i=1}^{|\mathbb{C}|}$ is

$$\mathsf{cost}(A, B, \mathbb{C}) = \frac{\sum_{c_i \in \mathbb{C}} |c_i| \cdot \big(n(A, c_i) + n(B, c_i)\big)}{|E| \cdot \big(\sum_{(e,f)\in A} f \ + \sum_{(e,f)\in B} f\big)}.$$

Example 6.2. Let $c_1 = \textit{Social Networks}$, $c_2 = \textit{Search Engines}$, $c_3 = \textit{Shopping}$, and $c_4 = \textit{Amazon}$ for the taxonomy of Figure 2(c). For the multisets M_1 and M_2 mentioned earlier in this section, we have: $n(M_1, c_1) = 2$, $n(M_1, c_3) = 2$, $n(M_1,\ c_4) = \infty$, and $n(M_2, c_1) = 2$. Let $\mathbb{C}_1 = \{c_i\}_{i=1}^{3}$ and Let $\mathbb{C}_2 = \{c_i\}_{i=1}^{4}$. Then, $\mathsf{cost}(M_1, M_2, \mathbb{C}_1) = \frac{25}{72}$, and $\mathsf{cost}(M_1, M_2, \mathbb{C}_2) = \infty$.

Algorithm 5: eventDist(Multiset A, Multiset B, Node π)

```
if both $A$ and $B$ are empty then return 0                                    ▷ Equal multisets
if either $A$ or $B$ is empty then return $\infty$                             ▷ Either one (not both)
if $\pi$ is a leaf node then return cost$(A, B, \pi)$                          ▷ Case 1 (Definition 6.1)
Set split point $sp$ = maximum event $Id$ in LeftSubTree$(\pi)$
Split $A$ using $sp$ into two non-overlapping multisets $A_1$ and $A_2$
Split $B$ using $sp$ into two non-overlapping multisets $B_1$ and $B_2$
if both $A_1$ and $B_1$ are empty then                                         ▷ Case 2.a
    return eventDist$(A_2, B_2,$ RightSubTree$(\pi))$
if both $A_2$ and $B_2$ are empty then                                         ▷ Case 2.b
    return eventDist$(A_1, B_1,$ LeftSubTree$(\pi))$
if any one of $A_1, A_2, B_1$, or $B_2$ is empty then                          ▷ Case 3
    return cost$(A, B, \pi)$                                                   ▷ Using Definition 6.1
$c_1$ = eventDist$(A_1, B_1,$ LeftSubTreeleft$(\pi))$
$c_2$ = eventDist$(A_2, B_2,$ RightSubTree$(\pi))$
return $(c_1 + c_2)$                                                           ▷ Case 4
```

Definition 6.3 (*Event Distance*). Let R and S be sequences and M_R and M_S be the multisets derived from R and S, respectively. The event distance between R and S is

$$\text{eventDist}(R, S) = \min_{\forall \mathbb{C} \in 2^H} \left\{ \text{cost}\left(M_R, M_S, \mathbb{C}\right) \right\},$$

where 2^H is the space of possible matchings from taxonomy H.

Computing eventDist is in fact an optimization problem. An exhaustive approach requires $2^{nodes(H)}$ examination for all possible subset of nodes in taxonomy H to find the best $\mathbb{C}$. However, some checkings are redundant and can be pruned by branch and bound. We first consider a binary event taxonomy (e.g., Figure 2(c)) and propose Algorithm 5 to compute eventDist. We then extend our algorithm to arbitrary event taxonomies.

6.2.2. Computing eventDist *(Binary Taxonomy).* Algorithm 5 reduces the search space for optimal $\mathbb{C}$ using (1) an ordered list of events in multisets, and (2) the structure imposed by taxonomy H. To compute eventDist, we first assign an *id* to each event by performing a depth-first traversal on H. The assignment of ids for the taxonomy of Figure 2(c) is demonstrated in Figure 5, in which the ids are the numbers in brackets in the leaf nodes (e.g., ID 7 is assigned to Amazon). We then re-present each multiset using an ordered list of IDs and frequencies. For instance, the multisets M_1 and M_2 introduced earlier in this section become {(1,1), (2,1), (3,1), (5,1), (6,1)} and {(1,2), (4,1), (7,1)}, respectively. We use Algorithm 5 to prune the search space for optimal set of nodes using the taxonomy and the order of event as a yardstick. We use a running example to convey the intuition:

Example 6.4. Figure 5 reproduces the same taxonomy of Figure 2(c), with node labels replaced by $\pi_{i,j}$ for brevity. The frequency of events are not shown for brevity. To find the distance of multisets M_1 and M_2, we start from the root node $\pi_{4,1}$ and split M_1 into $M_{1,1} = \{1, 2, 3\}$ and $M_{1,2} = \{5, 6\}$ using $sp = 4$ as the split point of $\pi_{4,1}$. This is because only the nodes in the left subtree of $\pi_{4,1}$ can be applied to *both* $M_{1,1}$ and $M_{2,1}$. Also, only the nodes in the right subtree of $\pi_{4,1}$ can be applied to both $M_{1,2}$ and $M_{2,2} = \{7\}$. Thus, we recursively call eventDist (Case 4 of Algorithm 5); once with $M_{1,1}$, $M_{2,1}$, and $\pi_{3,1}$ and once with $M_{1,2}$, $M_{2,2}$, and $\pi_{3,2}$.

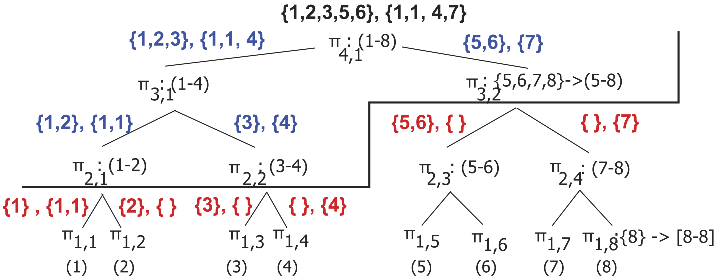


Fig. 5. Pruning taxonomy nodes for computing *eventDist*.

The first recursive call, has two branches. The first branch evaluates the distance between $\{1, 2\}$ and $\{1, 1\}$. There are two nodes under $\pi_{2,1}$. $\pi_{1,1}$ applies to splits $\{1\}$ and $\{1, 1\}$. Applying $\pi_{1,2}$ does not unify $\{1, 2\}$ and $\{1, 1\}$. Thus, we *roll back* and try the next closest map (i.e., $\pi_{2,1}$) which makes $\{1, 2\}$ and $\{1, 1\}$ equal. Thus, we commit the cost of applying $\pi_{2,1}$ in the total cost (Case 3) and roll back again. The second branch passes $\{3\}$ and $\{4\}$ to node $\pi_{2,2}$. Again, the split at $\pi_{2,2}$ level creates empty multisets (Case 3). Thus, we include the cost of applying $\pi_{2,2}$ on $\{3\}$ and $\{4\}$ and return.

The second recursive call considers the node $\pi_{3,2}$ and multisets $M_{1,2}$ and $M_{2,2}$ mentioned earlier. The split point of $\pi_{3,2}$ is 6 and splitting $M_{1,2}$ and $M_{2,2}$ at $\pi_{3,2}$ would generate at least one empty multiset on each branch. This means considering the branch under $\pi_{3,2}$ will not find any node which makes $M_{1,2}$ and $M_{2,2}$ equal and indeed $\pi_{3,2}$ is the *best* node. This ends the second round of recursive calls and returns with $\mathsf{cost}(M_{2,1}, M_{2,2}, \pi_{3,2})$, which consequently returns $\frac{25}{72}$ as $\mathsf{eventDist}(M_1, M_2)$. In summary, to find the optimal set of nodes $\pi^* = \{\pi_{2,1}, \pi_{2,2}, \pi_{3,2}\}$, we examined $\pi^* \cup \{\pi_{4,1}, \pi_{3,1}\}$ and pruned other redundant nodes (under the line in Figure 5). We return $\mathsf{cost}(M_1, M_2, \pi^*)$ as the eventDist of sequences corresponding to M_1 and M_2.

6.2.3. Computing eventDist *(Arbitrary Taxonomy).* Algorithm 5 can be extended when the taxonomy is not a binary tree and the fan-out of each node n_i is $d(n_i)$. Each node has $d(n_i) - 1$ ordered split points. We split A and B, respectively, into $A_i|_{i=1}^{d(n_i)}$ and $B_i|_{i=1}^{d(n_i)}$ splits. If for every $1 \leq i \leq d(\pi_i)$, either A_i and B_i are both empty or A_i and B_i are both nonempty, A and B must be split recursively (similar to Case 4 of Algorithm 5) and the total cost must be returned. In this case, if both A_i and B_i are not empty, the cost is evaluated by passing A_i, B_i, and the node corresponding to the ith child. Else, this is similar to Case 3, we use mapping n_i for A and B and return its cost (Line 12). For a leaf node, we return the cost of applying the leaf node (Case 1 of Algorithm 5). Algorithm 5 computes the distance in a time linear to the number of events of the multisets. In AOL dataset (Section 8), the taxonomy of URLs has only four levels and fits in main memory.

LEMMA 6.5 (COMPLEXITY OF eventDist). *For two sequences A and B, a taxonomy of height h with maximum fan-out d,* $\mathsf{eventDist}(A, B)$ *has $O(|A|+|B|+nhd)$ time complexity, where n is the maximum number of distinct events in A, B.*

PROOF. Algorithm 5 splits multisets using binary search. In the extreme case, it makes a recursive call at each step (Case 4) except for the last one. The length of each multiset is divided by two on average in each call. Thus, the number of binary search is $\sum_{i=1}^{\lfloor \log_2 |A| \rfloor} 2^i \cdot \log_2 \frac{|A|}{2^i} \approx 2|A| - \log_2(|A|) - 2$ which makes the search time $O\big(|A| + |B|\big)$. Let n be the maximum number of distinct events in A or B. In the extreme case,

Algorithm 6: BIASEDPR (Cluster $\mathbb{C}$, Privacy model $\mathcal{M}$)

```
Set $\tau$ to $k$ if $\mathcal{M}$ is $k$-anonymity and to $\ell$ if $\mathcal{M}$ is $(g, \ell)$-diversity
$\mathcal{P} = C$
repeat
    $\mathcal{P}_b = \mathcal{P}$ and $\mathcal{P}_a = \emptyset$
    $rnd$ = random number generated from uniform distribution in range [0–1]
    for each cluster $C$ in the set $\mathcal{P}$ do
        $\mathcal{P} = \mathcal{P} - C$
        if $rnd \leq \frac{w_e}{w_e+w_t}$ then $\mathcal{P}_a = \mathcal{P}_a \cup$ PARTITION( $C$, $\tau$ )    ▷ PARTITION in Algorithm 4
        else$\mathcal{P}_a = \mathcal{P}_a \cup$ REFINE( $C$, $\tau$ )    ▷ REFINE in Algorithm 4
    $\mathcal{P} = \mathcal{P}_a$
until $|\mathcal{P}_a| = |\mathcal{P}_b|$
return $\mathcal{P}$
```

Algorithm 5 assigns a leaf category (at height h) to each event, making the cost of traversing to $O(hdn)$; at each step, for a balanced taxonomy of height h and a maximum fan-out d, a loop is required to split $A(B)$ into $d(\pi_i) \leq d$ splits. The time complexity is $O(|A| + |B| + hdn)$. □

6.3. Extension of PR when $w_t \neq w_e$

Algorithm PR switches between partitioning along the time and the event dimensions. However, if $w_t \neq w_e$, the round-robin switch in PR would not account for relative importance of time over events or vice versa. One heuristic, to address this issue, is to invoke PARTITION (which refines events) more often than REFINE (which refines time) when w_e is larger than w_t. This policy is chosen to provide an increased chance to partition data along the event (i.e., URL) dimension in comparison with time, thus we expect it to find an anonymization with a better event generalization. Likewise, when w_t is larger we perform REFINE more often than REFINE to produce more compact time intervals. This approach, can be implemented by replacing the call to Partition at Line 3 of PR with BIASEDPR. BIASEDRR is depicted in Algorithm 6. During each iteration of the biased round-robin loop, PARTITION is invoked with probability $\frac{w_e}{w_e+w_t}$ and REFINE is invoked with probability $\frac{w_t}{w_e+w_t}$. Trivially, BIASEDPR simplifies to one of PARTITION or REFINE, respectively, when $w_t = 0$ or $w_e = 0$. In each iteration, BIASEDPR splits clusters in $\mathcal{P}$ to refine it along either time or event dimension. Eventually, when none of the clusters of $\mathcal{P}$ could be split, BIASEDPR terminates and returns a partitioning which is biased to the weights w_t and w_e. As we mentioned, BIASEDPR replaces Line 3 of Algorithm 4 when $w_t \neq w_e$. Those clusters, found by BIASEDPR, that violate a privacy constraint are merged, in Lines 4–9 of Algorithm 4, with other clusters to resolve privacy violations.

7. COMPUTING GENERALIZATIONS

In this section, we focus on each anonymization group and explain how the optimal time intervals and event generalizations are derived for each group. Recall that in both SSR and SCR setting, each group is represented by a set of intervals. In Section 7.1, we explain how to derive an optimal set of intervals. The Appendix discusses vulnerability issues of anonymized data in SSR and SCR settings.

7.1. Finding the Optimal Set of Intervals

Given a group of sequences G and the event set E_r, we find the optimal anonymization of sequences in G for release in three steps using ANONYMIZE (Algorithm 7). First, the group is divided into two sets $\mathcal{L}$ and $\mathcal{R}$. In the second step, we generalize event and time pairs in $\mathcal{L}$ only because these pairs can be used as quasi-identifiers in either the SSR or the SCR setting. In the third step, we combine the generalized sequences of $\mathcal{L}$ with the original sequences of $\mathcal{R}$ to produce the published anonymized data. The first step of ANONYMIZE (Lines 1–4) divides G into two groups, $\mathcal{L}$ and $\mathcal{R}$. Intuitively, $\mathcal{R}$ contains event participations that are not monitored by any potential receiver. Naturally, $\mathcal{R}$ is empty in the extreme setting of SCR. The set $\mathcal{L}$ contains all sequences which are available to the receiver and thus we focus on anonymizing $\mathcal{L}$ in the second step.

Notations. Let T_S be the set of time stamps in sequence S and let $T_{\mathcal{L}} = \cup_{S \in \mathcal{L}} T_S$ be the set of all time stamps in $\mathcal{L}$ and $n = |T_{\mathcal{L}}|$. Assume a total order on the set of time stamps $T_{\mathcal{L}}$, that is, for each t_i and t_j in $T_{\mathcal{L}}$, $t_i < t_j \Leftrightarrow i < j$. For any sequence S, let $S_{[t_i,t_j]}$ be a sequence derived from S by removing every time stamp (and its corresponding event) of S which is not included in time interval $t_{i,j} = [t_i - t_j]$. When S has no time stamp in interval $t_{i,j}$, then $S_{[t_i,t_j]}$ is empty.

Definition 7.1 (*Feasible Interval*). Given sequence S, interval $I = (c, [t_1 - t_2])$ is a feasible interval for S if $S_{[t_1,t_2]}$ is not empty and for any pair $(e, t) \in S_{[t_1,t_2]}$, event e is generalized by category c in the taxonomy. Interval I is a feasible interval for group $\mathcal{L}$ if I is feasible for every sequence in $\mathcal{L}$. Similarly, a set of intervals $\mathcal{I}$ is feasible for sequence S (group $\mathcal{L}$) if every interval $I \in \mathcal{I}$ is feasible for S ($\mathcal{L}$), resp.

Our goal is to find the set of feasible intervals $\mathcal{I}^*$ which is feasible for $\mathcal{L}$ and minimizes IL (Eq. (3)). The reason that we only consider feasible intervals is because they satisfy the strong coverage property; each (event, time) pair will be covered by exactly one feasible interval. We find $\mathcal{I}^*$ by induction as follows. Let $Opt(i, j)$ be the minimum information loss when all sequences in $\mathcal{L}$ before (including) t_i is approximated by an interval set with j intervals. For instance, $Opt(n, 1)$ denotes the minimum information loss when all sequences of $\mathcal{L}$ are approximated by an interval set with only one interval (recall, $n = T_{\mathcal{L}}$). Our quest for optimal set of feasible intervals ends when we find $Opt(n\ , m^*)$, where m^* is the cardinality of the largest set of feasible intervals on $\mathcal{L}$. Intuitively, when $t_{1,j} = [t_1 - t_j]$ is divided into two non-overlapping subintervals $t_{1,r}$ and $t_{r+1,j}$, the minimum distortion during $t_{1,j}$ is smaller than if the two intervals are generalized individually during any one of the two subintervals provided that both subintervals are feasible. Formally, $Opt(i, 1) = IL(\mathcal{L}, I^*_{1,i})$, and

$$Opt(i, m) = \min_{1 \le r < i-1} \left\{ Opt(r, m-1) + IL(\mathcal{L}, I^*_{r+1,i}) \right\},$$

where $I^*_{r,i} = (c_{r,i}, [t_r - t_i])$ is the optimal feasible interval on $\mathcal{L}$ and $c_{r,i}$ is the *lowest common ancestor* (LCA) category in the taxonomy for all events of $\mathcal{L}$ during $[t_r - t_i]$. If there is no feasible interval on $\mathcal{L}$ during $[t_r - t_i]$, $IL(\mathcal{L}, I^*_{r,i}) = \infty$. This property (the definition of Opt) is due to the strong coverage property which imposes a constraint on the feasibility of intervals and implies that Opt satisfies the principle of optimality. Thus, Opt and the optimum set of intervals can be computed by dynamic programming (step 2 of ANONYMIZE). The last step creates the anonymized data. Because sequences in G must be indistinguishable regarding their events in E_r, we concatenate each sequence in $\mathcal{R}$ with the optimal set of intervals of $\mathcal{L}$ (step 3 of ANONYMIZE).

Time Complexity. The information loss for single interval generalizations are constructed in Lines 6–7. The maximum number of feasible intervals m^* is bounded by l_{min}, the length of the shortest sequence in $\mathcal{L}$. If constructing m non-overlapping feasible intervals is impossible for the set of time stamps $\{t_1, \ldots, t_{f_m - 1}\}$, then it is impossible to construct $m + 1$ feasible intervals for the set of time stamps $\{t_1, \ldots, t_{f_m}\}$ as well. ANONYMIZE uses this heuristic to prune Opt computations for $t_i \le t_{f_{m-1}}$. The algorithm

Algorithm 7: ANONYMIZE (Partition G, Event set E_r)

```
/* step 1) divide $G$ using $E_r$ into 2 parts */
Set $\mathcal{L} = \{\}$ and $\mathcal{R} = \{\}$
for each sequence $S$ in $G$ do
    Set $S_L = \{(e,t) \in S | e \in E_r\}$, $S_R = \{(e,t) \in S | e \notin E_r\}$
    Add $S_L$ to $\mathcal{L}$ and $S_R$ to $\mathcal{R}$
/* step 2) find optimal set of intervals $\mathcal{I}$ */
Set $T_{\mathcal{L}} = \{t_1\}|_{i=1}^{n}$ to the set of timestamps in $\mathcal{L}$, $i < j \Leftrightarrow t_i < t_j$
for $i = 1$ to $n$ do
    $Opt(i, 1) = IL\big(\mathcal{L}, (c_{1,i}, [t_1 - t_i])\big)$
for $m = 2$ to $l_{min}$ do                                    ▷ $l_{min} = \min_{S \in \mathcal{L}} |S|$
    $Opt(i, m)|_{i=1}^{n} = \infty$
    $f_{m-1} = \arg\min_j Opt(j, m-1)$
    for $i = f_{m-1} + 1$ to $n$ do
        for $r = f_{m-1}$ to $i - 1$ do
            $Opt(i, m) = \min_r \{Opt(r, m-1) + IL\big(\mathcal{L}, I^*_{r+1,i}\big)\}$
            $prev(i, m) = r$ that minimizes $Opt(i, m)$
    if $Opt(n, m) = \infty$ then
    $m^* = m$
Set $\mathcal{I} = \{\}$, $e = n$, and $m = m^*$
while $m > 1$ do
    $b = prev(e, m)$
    $\mathcal{I} = \mathcal{I} \cup \{(c_{b+1,e}, [t_{b+1} - t_e])\}$              ▷ add feasible interval to $\mathcal{I}$
    $e = b$                                ▷ end for the next feasible interval (if exists)
    $m = m - 1$
$\mathcal{I} = \mathcal{I} \cup \{(c_{1,e}, [t_1 - t_e])\}$
/* step 3) anonymize $G$ to publish */
for each sequence $S$ in $\mathcal{R}$ do
    Publish $\mathcal{I}$ and $S$                ▷ same generalizations for all sequences in $\mathcal{L}$
```

computes at most $n \times l_{min}$ entries of Opt array. The computation of each entry involves at most n evaluations of IL. Computing $IL(\mathcal{L}, I^*_{i,j})$ requires two steps. First, checking the feasibility of $t_{i,j}$ for every sequence in $\mathcal{L}$ and then compute LCA of events of $\mathcal{L}$ in this interval. Because time stamps are ordered, each check needs $O(\log l_{avg})$ time for a sequence of average length l_{avg}, or $O(n_{\mathcal{L}} \log l_{avg})$ in average for all $n_{\mathcal{L}}$ sequences in $\mathcal{L}$. Computing LCA requires $O(h)$ for any two pair of events, h is the height of the taxonomy, or $O(h \log n)$ for all n events if LCA is computed by tournament selection. For any $t \in T_{\mathcal{L}}$, let $cf(t) = \sum_{S \in \mathcal{L}} |S_{[t_1,t]}|$ be the cumulative frequency of time stamps. The information loss for a feasible interval $I_{i,j}$ can be computed in $O(\log n)$ time by searching for $cf(t_i)$ and $cf(t_j)$ in a sorted array with n pairs of $(t, cf(t))$. Thus, computing $IL(\mathcal{L}, I_{i,j})$ has $O(n_{\mathcal{L}} \log l_{avg} + h \log n)$ time complexity and the time complexity of ANONYMIZE is $O\big(n^2 l_{min}(n_{\mathcal{L}} \log l_{avg} + h \log n)\big)$. The total space requirement is $O(l_{min} n)$: (i) $O(n)$ to keep n pairs of $(t, cf(t))$ and two consecutive rows of Opt array, and (ii) $O(l_{min} n)$ to keep the path in $prev$ array which is used to find the optimal set of intervals in $O(l_{min})$ time.

Example 7.2. Assume the SCR setting, where $E_r = \{\text{Google, Bing}\}$. Table II shows a database divided into the sets $\mathcal{L}$ and $\mathcal{R}$ by the first step of ANONYMIZE. For brevity, sequences are summarized (e.g., Bing: [1, 11] stands for (Bing, 1) and (Bing, 11)). Running BASELINE to achieve 2-anonymity creates three partitions; $\mathcal{P}_1 = \{S_1, S_2, S_3\}$, $\mathcal{P}_2 = \{S_4, S_5\}$, and $\mathcal{P}_3 = \{S_6, S_7\}$. We call ANONYMIZE on $\mathcal{P}_1$ to create the anonymized version of sequences in $\mathcal{P}_1$. Intuitively, the optimal set of intervals have at most two

Table II. A Click-Stream Database

ID	$\mathcal{L}$	$\mathcal{R}$
S_1	Google: [3], Bing[7,10]	Ebay: [1]
S_2	Google: [2], Bing:[9]	Myspace: [6]
S_3	Google: [3], Bing: [1,11]	Ebay: [5]
S_4	Google: [7]	Facebook: [9]
S_5	Bing: [8]	Amazon: [10]
S_6		Twitter: [1]
S_7		Youtube: [10]

Table III. 2-Anonymous Published Data

Search Engine: [1-3], Bing: [7-11], Ebay: [1]
Search Engine: [1-3], Bing: [7-11], Myspace: [6]
Search Engine: [1-3], Bing: [7-11], Ebay: [5]
Search Engine: [7-8], Facebook: [9]
Search Engine: [7-8], Amazon: [10]
Twitter: [1]
Youtube: [10]

feasible intervals, as limited by the length of the shortest sequence in $\mathcal{L}$ for $\mathcal{P}_1$ which is two (the section of S_2 in $\mathcal{L}$). The end point of the first feasible interval must be at least at time stamp 3, before this time point the strong coverage property does not hold. Similarly, the last time point of the first feasible interval cannot be at or after 9. Within time range [1–3], we next compute the information loss at each time point in $Opt(i, 1)$. We assume $w_t = w_e = 1$. The distortion for optimal interval during [1–3] is $Opt(3, 1) = 0.5 * (\frac{(3-1)}{10} + \frac{2}{8})$ using Eq. (2); the first term is for generalizing time interval to [1–3] and the second term is to replace Google and Bing by their lowest common ancestor in Figure 2(c) (Search Engine) which covers two out of $|E| = 8$ URLs. In summary, the optimal set of feasible intervals for $\mathcal{P}_1$ is $\mathcal{I}^* = \{$ (Search Engine, [1–3]), (Bing, 7–11) $\}$ with an information loss equal to $Opt(3, 1) + 0.5 * \frac{(11-7)}{10}$; no event generalization is required in the second interval of $\mathcal{I}^*$. Table III presents the anonymized data after applying ANONYMIZE on partitions $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$.

8. EXPERIMENTAL EVALUATION

We present the result of an empirical evaluation of our algorithms. Table IV summarizes the methods and Table V summarizes the datasets. We only present a representative subset of our results. Sections 8.1 and 8.2 present datasets used and the settings. Section 8.3 compares our algorithms with related work. Sections 8.4 and 8.5 investigate the usefulness of our optimizations for SSR and SCR settings. Section 8.4.4 studies the scalability of our methods and measures the resource overhead of our optimizations.

8.1. Datasets

8.1.1. AOL. Contains the query log of users during three months in 2006. This dataset has been mainly used to study the anonymization of query logs using keyword generalization [He and Naughton 2009]. Each record indicates either a search or a URL visit. After data cleaning, there were 521,692 click streams. We considered two scenarios: First, AOL shares data with a search engine or a website (SSR setting). We picked Yahoo and Flickr as candidates because the average number of visits to these websites per sequence is among the largest and the smallest values, respectively. Thus, $E_r = \{\text{Yahoo}\}$ and $E_r = \{\text{Flickr}\}$, respectively, in datasets Yahoo and Flickr. Second, AOL

Table IV. Methods Used in Experimental Study

Method	Description
TA	Trajectory Anonymization algorithm [Nergiz et al. 2009]
SA	Symmetric Anonymization algorithm [Yarovoy et al. 2009]
HN	BASELINE
HI	BASELINE using FASTNN
HY	HYBRID using BASELINE with FASTNN
HL	HY with all sequences are assigned to region A
ET	PR

Table V. Summary of Datasets Used for Evaluations

Property	Flickr	Yahoo	Google	Mixed	Oldenburg	Worldcup
$\|\mathcal{D}\|$	6,919	33.4K	84.5K	243.4K	11,571	19.4K
$\|$region$A\|$	6,074	25.4K	63.8K	159.1K	8,060	15.6K
Max. len.	54	867	1,895	8.6K	186	685
Avg. len.	1.2	3.8	3.1	4.9	7.0	14.3
(e, t) pairs	1.5M	2.5M	5.6M	14.8M	0.5M	6.2M
$\|E\|$	37.6K	451.8K	812K	1.4M	635	19K

Note: $|$region$A|$ is the number of sequences in region A, see Figure 3 for a demonstrative example.

shares data with colluding receivers (i.e. SCR). We extract the Mixed dataset from AOL for E_r = {Google, Yahoo, MySpace, eBay, Amazon, Wikipedia}. We use Mixed for evaluating the SCR setting (Section 8.5) and for scalability evaluations (Section 8.4.4).

8.1.2. Worldcup98. Contains the access log to WorldCup 1998 website[15] with 2,140,622 click streams. We considered sharing click streams with a party that collects time stamps of visiting one random URL in the last week of the tournament during which the site experienced a large number of visits. We excluded URLs visited by almost all users, as these pages may correspond to the homepage or navigational links and we did not regard them as sensitive. We used this dataset for evaluation of SSR.

8.1.3. Oldenburg. Contains synthetic data generated using Brinkhoff's traffic data generator [Brinkhoff 2003] for the city of Oldenburg with parameters set to their default values. We discretized the city map using a uniform grid with 1,024 cells and assigned an identifier to each cell. If a trajectory contains a point in a grid cell we consider a visit to the corresponding cell at the time stamp of visit. For each cell, we compute the number of trajectory visits to the cell and consider the scenario where a store, with several branches in the city, records visit times of customers. We considered a controller that monitored the top 10 locations in terms of the number of visits. Out of 25K trajectories created by the data generator, we picked the trajectories that visited any of the monitored locations. We used this dataset for evaluation of SSR setting.

8.2. Settings

We implemented algorithms in C and conducted experiments on a machine with a dual core 3Ghz CPU, 2GB RAM, running Linux Fedora 12. The index for clusters was implemented using R-Tree[16] with a fan-out of 100 and 4K pages. We stored sequences in a B-Tree index with sequence ID as key. Each fingerprint was 4 bytes. In SSR experiments, we set $w_e = 0$ and in SCR experiments, we set $w_e = w_t = 1$. We extracted URL categories

[15] http://ita.ee.lbl.gov/html/contrib/WorldCup.html

[16] http://www2.research.att.com/∼marioh/spatialindex/

in AOL dataset by sending queries to Alexa[17] in October 2009. One of the three cases happened for each URL: (1) Alexa had the category of the URL[18], (2) Alexa returned a list of similar URLs, and (3) Alexa had no entry. We only consider the first four categories from the longest category returned by Alexa for each URL. We organized URLs, based on their categories, into a taxonomy. In case 2, the URL was assigned to the category which is shared among its similar URLs. In case 3, the URL was assigned to the Miscellaneous category, which contains typos and un-popular URLs. This way, we could build a taxonomy of height four, 45,383 nodes, maximum fan-out of 128 (avg. 3), and 884K categorized URLs (case 1 or 2) from over 1.6M URLs. We only used the Mixed dataset for the evaluation of the SCR setting and for scalability experiments.

8.3. Comparison with Related Work

We compared HN (i.e., the BASELINE algorithm) with the two comparable state-of-the-art *syntactic* anonymization methods for sequence databases; the Trajectory Anonymization [Nergiz et al. 2009] and the Symmetric Anonymization [Yarovoy et al. 2009], which we refer to in the rest, respectively, as TA and SA.

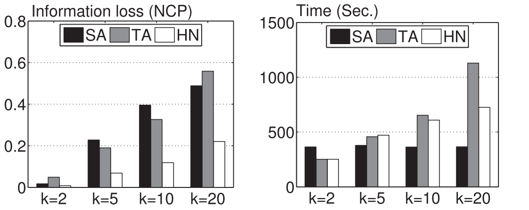
- TA clusters sequences into groups of size at least k to achieve k-anonymity. This is similar to our approach. However, TA does not enforce Strong Coverage (Property 2.3). Instead, TA applies suppression, when sequences that fall in the same group are of different length. Like HN, TA applies location and time generalization.
- SA requires a prespecified set of quasi-identifiers (QIDs). It optimizes a *time independent* distortion measure. For each trajectory, if performs an approximate nearest neighbor search, based on the location of the trajectory at its QIDs. The closeness is measured in terms of the aggregate difference of the Hilbert index extracted from locations. Finally, SA enforces a *symmetric* property over the attack graph, which may result into generating anonymization groups with more than $2k$ trajectories.

Several other works studied syntactic anonymization of sequence data. However, the problem setting of SA and TA are the most similar to ours, thus we compared our approach with them. Following TA and SA, syntactic sequence anonymization has been studied in different settings. Huo et al. [2012] extend Abul et al. [2008] by generalizing only the *stay points* (check-in places, credit card transaction places, etc.). It is not clear how to define stay points for our problem. Monreale et al. [2010] extend SA by finding a tessellation of geographical data into sub-areas. It is not clear how this technique can be applied on non-geographical data. Mahdavifar et al. [2012] consider publishing moving objects, where each has a different privacy requirements. No systematic approach was proposed to elicit each trajectory's privacy requirement. When a sensitive attribute is attached to each trajectory (e.g., patient's trajectory and their disease), Chen et al. [2013] propose (K,C)L privacy model. In our setting, there is no separation between sensitive and nonsensitive (event, time stamp) pairs.

We stress that the anonymized data produced by HN conforms with the symmetric requirement of SA; each anonymization group (AG) of HN is a complete bipartite graph, thus the induced attack graph [Yarovoy et al. 2009] by HN is symmetric. To assure the fairness of our comparisons, we compare TA, SA, and HN in the following setting. We set $E_r = E$, the set of all possible events. Thus, HN generalizes all (event, time) pairs, TA clusters complete sequences, and SA assumes all (event, time) pairs

[17] `http://www.alexa.com`

[18] E.g., for Facebook Alexa returned Computers/Internet/On the Web/Online Communities/Social Networking as the longest category.



(a) Information loss. (b) Anonymization time.

Fig. 6. k-anonymity in Oldenburg dataset, varying k.

as quasi-identifiers.[19] Thus, HN, SA, and TA all anonymize complete sequences in this experiment. Except TA and SA, our approach is not comparable to previous works (e.g., [Abul et al. 2008] and [Terrovitis and Mamoulis 2008]) mainly because they all ignore time points of trajectories, both in forming AGs and during generalization.

We compare running time and the quality of the anonymized data generated by these methods with respect to data distortion and workload based utility measures. Because SA requires a predefined set of quasi-identifiers (QIDs) for trajectories, we picked the set of all time points as QIDs. This is to provide a common ground for fair comparison. This setting provides an equivalent privacy level for the three methods. Furthermore, to make the information loss independent from the employed taxonomy, we slightly changed the definition of data distortion in Eq. (2) by setting IL_e to the sum of distortion along the spatial coordinates of trajectories. This was motivated by the observation of Sherkat and Rafiei [2008] that minimizing the sum was shown to be superior to minimizing the volume in terms of preserving quality. Finally, we picked Oldenburg dataset as TA and SA perform geometric transformations on spatial coordinates.

8.3.1. Information Loss. Figure 6(a) shows that information loss increases monotonically with k for all three methods. This is because the anonymization groups get larger in order to fulfill the desired privacy level. HN incurs the smallest distortion and there is no winner between SA and TA; for $k = 2, 20$ the former has a smaller distortion but TA performs better when $k = 5$ and 10. A key reason behind high information loss of TA is that each group is biased by the selection of the first trajectory in each group. The group is formed in TA by finding k-1 nearest neighbor to the seed, each time merged with a new sequence to create a new seed. While SA benefits from preserving the exact value of time points (it only generalizes location) but from the other side, it may group more than $2k$ points to ensure that attack graph is symmetric. Naturally, since HN targets optimizing the information loss, a smaller distortion is observed in HN.

8.3.2. Running Time. The running time for all three methods increases with k (Figure 6(b)); the amount of increase is smallest in SA compared to the other two methods (from 479 sec. to 541 sec). Although both TA and HN do hierarchical clustering, but at each step TA merges a cluster with a sequence whereas HN merges clusters. Thus, HN may require, in total, a smaller number of cluster comparisons and information loss computations, specially when k is large. TA and HN evaluate computationally expensive information loss measures to form clusters but SA uses a

[19] SA [Yarovoy et al. 2009] requires that the set of quasi-identifiers (as time points) to be known ahead of time. But no method was proposed by the authors to find the set of quasi-identifiers.

Table VI. Range Query Distortion Changing k (Oldenburg)

	TA [Nergiz et al. 2009]			**HN**	**SA [Yarovoy et al. 2009]**
k	**Suppression**	f_n	f_p	f_p	f_p
2	6.29 %	0.03	0.18	0.23	0.37
5	25.76 %	0.07	0.25	0.54	0.66
10	32.74 %	0.09	0.25	0.55	0.73
20	37.56 %	0.09	0.27	0.74	0.74

score which is linear in length of sequences. Thus, its running time is less than the other two methods except at $k = 2$, where the overhead of maintaining the symmetric attack graph is larger than the saving from computing information loss.

8.3.3. Range Query Distortion. We report false positive (f_p) and false negative (f_n) ratios for range queries as a metric for utility of the anonymized data. Such queries can be modules of complex spatiotemporal pattern queries [Hadjieleftheriou et al. 2005]. Each query q has a spatial range s_q and a temporal range t_q. Let $nHits(q, \mathcal{D})$ be the number of trajectories in dataset $\mathcal{D}$ that pass over the region s_q during time t_q. For anonymized dataset $\mathcal{D}^*$, the locations and time points are generalized to intervals. The intervals may overlap partially or completely with s_q and t_q. Let $nHits(q, \mathcal{D}^*)$ be the number of trajectories in $\mathcal{D}^*$ that overlap with query q. If there is no suppression, $nHits(q, \mathcal{D}^*)$ overestimates $nHits(q, \mathcal{D})$. For query workload Q, we define the ratios as

$$f_p(\mathcal{D}^*) = \frac{\left|\{q \mid q \in Q,\, nHits(q, \mathcal{D}^*) > nHits(q, \mathcal{D})\}\right|}{|Q|},$$

$$f_n(\mathcal{D}^*) = \frac{\left|\{q \mid q \in Q,\, nHits(q, \mathcal{D}^*) < nHits(q, \mathcal{D})\}\right|}{\left|\{q \mid q \in Q,\, nHits(q, \mathcal{D}) > 0\}\right|}.$$

Both f_p and f_n are in range $[0, 1]$ and a smaller value is favorable for both ratios. In our evaluation, we divided the time, x, and y coordinates into 100 intervals of equal length. A combination of temporal and spatial dimensions provided the query workload Q which has $100 \times 100 \times 100$ queries. We observed that $f_n = 0$ for both HN and SA. In Table VI, we observe that TA has the smallest f_p ratio, because it has compact intervals; each interval has one time point from each sequence. Still, TA has the drawback of false negatives due to suppression. SA has the largest f_p ratio even though it does not perform time generalization for two reasons. (1) In SA, each AG is formed initially by finding $k - 1$ closest sequences to a single sequence in QID space. The AG can get biased towards a single sequence. In HN each AG is found iteratively: it is initialized to a single sequence and in an iterative process, it is merged with the clusters closest to the current AG. (2) The symmetric merging in SA generates larger AGs in an attempt to maintain the attack graph symmetric. This results into more data distortion, reflected in part as larger f_p ratio compared to HN. HN is the best approach, among the three, when false negatives are not acceptable.

8.3.4. Preserving Causality Patterns. We next evaluate the utility of anonymized data for causality queries. Unlike range queries in the previous section, which consider hits within a time interval and spatial region, causality queries measure how well the anonymized data can preserve causality patterns. In this work, we consider spatiotemporal prediction queries, as a case of causality queries. The goal (i.e., utility) is to predict the location of a trajectory at time $t + w$ given its location at time t. These queries are useful for mining temporal causality patterns (e.g., [Mannila et al. 1997]). Assume a relational schema as (tid, loc, time) for the Oldenburg dataset, where each

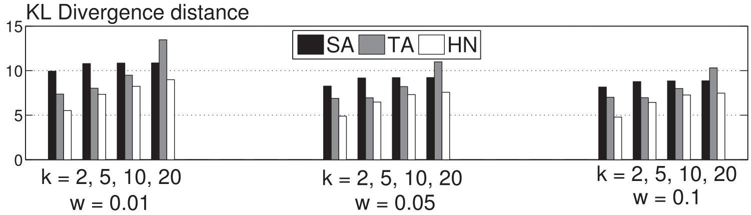


Fig. 7. KL Divergence, varying k and w.

tuple records the location *loc* at time stamp *time* for trajectory with identifier *tid*. A causality query can be written using the following SQL statement.

```
SELECT A.loc, B.loc, COUNT(DISTINCT A.tid)
FROM Oldenburg A, Oldenburg B
WHERE A.tid = B.tid
AND A.time <= B.time <= A.time + w
GROUP BY A.loc, B.loc;
```

Intuitively, for each location pair (x, y) the query finds the number of trajectories that make a visit to these two locations (in order) during a time interval of length w. Normalizing the count in this query induces a probability distribution $P^w_{x,y}$, which is similar to the transition probability of a Markovian process model [Papoulis and Pillai 2002]. We can derive $Q^w_{x,y}$ as an estimate of $P^w_{x,y}$ using the anonymized dataset. Let LOC be the set of all locations[20] and let Oldenburg* be the anonymized data with the schema (tid, [loc1-loc2], [time1-time2]). The causality query can be written as

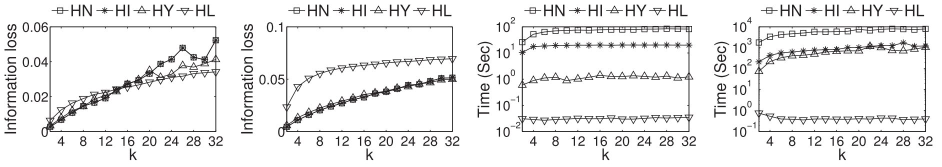
```
SELECT A.loc, B.loc,
 (SELECT count(Distinct C.tid)
  FROM Oldenburg* C, Oldenburg* D
  WHERE C.tid = D.tid
  AND C.loc1 <= A.loc <= C.loc2
  AND D.loc1 <= B.loc <= D.loc2
  AND C.time2 <= D.time1 <= C.time2 + w)
FROM LOC A, LOC B;
```

We use the Kullback-Leibler Divergence [Kullback and Leibler 1951] to measure the closeness of $P^w_{x,y}$ and $Q^w_{x,y}$, as it was used before as a representative metric in the data anonymization literature [Kifer and Gehrke 2006].

$$D_{KL}(Q^w_{x,y}, P^w_{x,y}) = \sum_{\forall x,y \in LOC} P^w(x, y) \log \frac{P^w(x,y)}{Q^w(x,y)}.$$

D_{KL} is zero for identical distributions and a smaller value denotes a more distribution preserving anonymization. Figure 7 shows D_{KL} for SA, TA, and HN when k and w increases. We set w to 1%, 5%, and 10% of the entire time span. We notice that for all methods, D_{KL} increases monotonically with k, which is expected because the anonymization groups get larger and it becomes harder to predict trajectory locations in each group. We also observe that the distribution of $Q^w_{x,y}$ gets closer to $P^w_{x,y}$ when w increases; this is because the amount of time generalization does not increase with w but the sensitivity of D_{KL} to anonymization decreases with w. We consistently observe a smaller (better) D_{KL} for HN. A large percentage of point suppression for TA in

[20] In this experiment, we divided the map into $1,024 \times 1,024$ cells of equal area to derive the set of all locations LOC.



(a) NCP, Flickr (b) NCP, Yahoo (c) Time, Flickr (d) Time, Yahoo

Fig. 8. k-anonymity changing k.

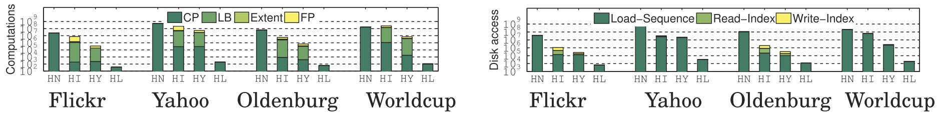
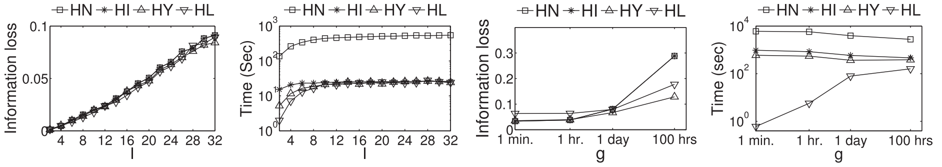


Fig. 9. Computations and disk accesses, k-anonymity ($k = 16$).



(a) NCP, Oldenburg. (b) Time, Oldenburg. (c) NCP, Yahoo. (d) Time, Yahoo.

Fig. 10. (g, ℓ)-diversity changing ℓ (g=1 hour) in (a), (b) and changing g (ℓ=16) in (c), (d).

Table VI appears as large D_{KL} when k is large; although the suppression in TA does not produce a significant amount of false negatives (less that 10%), it has a relatively large impact (up to 30%) on the accuracy of causality queries when $k = 20$.

8.4. Evaluation of the SSR Setting

8.4.1. k-Anonymity. As we observe in Figure 8, the information loss is very small for all methods (less than 6%).[21] The running time increases monotonically with k as more comparisons and disk accesses are needed to form larger anonymization groups. In Flickr, HL is very close to HN in terms of information loss as the average length of QIDs is relatively small (1.2 clicks/user) and the percentage of sequences that fall into region A is high. Thus, a fast heuristic in 2D space produces results with the same quality as HN. HY (in Flickr) benefits from partitioning and runs ten times faster than HI (Figure 8(c)) without a significant drop in information loss. In Yahoo, HL runs significantly faster than other methods. The average length of QIDs in Yahoo is larger than Flickr. We observe a large difference between information loss of HL and HN in Yahoo. However, HY makes a balance; it provides an improvement in running time over HI and the information loss of HY is very close to HN. In both Flickr and Yahoo, HI runs up to an order of magnitude faster than HN as the index and FastNN reduce the number of computations by almost four orders of magnitude and the number of disk accesses by more than one order of magnitude (Figure 9), which explains the observed improvement of running time for HY (Figures 8(c) and 8(d)). HI and HY have two overheads: creating index (I/O) and evaluating lower bounds (CPU). The lower bounds are very efficient to compute and both HI and HY benefit from the index. HL

[21] Since the global time-span is 91 days, an information loss of 6% is equal to a time distortion of five days (in average) for each time stamp.

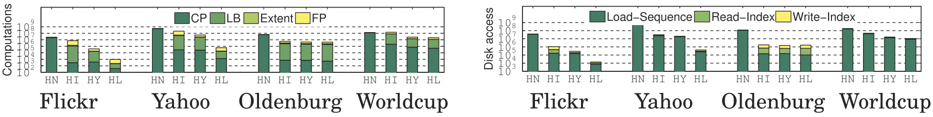


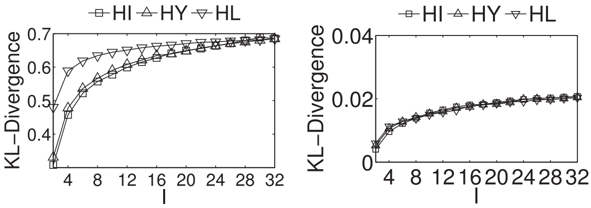
Fig. 11. Computations and disk accesses, (g, ℓ)-diversity, $\ell = 16$, $g = 1$ hour.

scans the data once for sorting and once for finding AGs at an I/O cost which is three to four orders of magnitude smaller than HN. Overall, the index and our heuristics are effective to reduce the anonymization time. HI has the same distortion as BASELINE but HY reduces the time for HI with a small increase in data distortion.

8.4.2. (g, ℓ)-Diversity. The information loss is very small for all datasets (Figure 10). The anonymization cost grows with ℓ due to an increase in group size, which is directly related to the increase of information loss. We observed a very similar pattern that we had observed before in k-anonymity, in terms of information loss and relative performance of HN, HY, and HL, in all datasets except for Oldenburg in which we see a unification of HI, HL, and HY (Figure 10(a) and Figure 10(b)). We noticed that the distribution of locations is dense in Oldenburg. Many locations, mostly in the center of the map, were visited by a large number of sequences during a small time interval. These sequences are candidates for (g, ℓ)-diversity violations as they visited the same location during a close time interval. In HL and HY, these sequences are handled by HI. Figure 11 confirms this pattern; HL and HY make the same number of computations as HI.

Increasing g adds more grouping constraints. This has two impacts in Figure 10(b): (1) in HN and HI, the number of candidate sequences to merge with a cluster reduces when g increases. This reduces the number of required computations and hence the running time. (2) For HY and HL, there will be more (g, ℓ)-diversity violations in larger intervals; more clusters become infeasible and will be passed to HI. Thus, the running time of HL and HY converge to HI. In terms of information loss (Figure 10(c)), it seems hat greedily checking constraints locally (as in HN and HI) does not always yield the best anonymization group. The index and optimizations are effective and reduce the number of disk accesses and computations, as shown in Figure 11, which results into reduction of the running time; HI has the same distortion of HN whereas HL and HY have improved running time with a small increase in data distortion.

8.4.3. Utility. To measure the usefulness from the perspective of data receiver, we considered the accuracy of causality queries. Inline with the queries and the accuracy measure (i.e., D_{KL}) introduced earlier in Section 8.3 for causality patterns, we assume that a receiver that collects time stamps of URL A wants to compute the distribution of visits for click streams within w time units of visits to URL A. For instance, Google may want to know the probability that one visits Amazon after five minutes of visiting Google, or conversely, the probability that one visits Bing five minutes before visiting Google. In Figure 12, we observe that D_{KL} is relatively small for AOL dataset, compared to that reported in Figure 7. The reason is that here we generalize only the section of each sequence that are collected by the corresponding receiver (i.e., the set $\mathcal{L}$ in Section 7). D_{KL} monotonically increases with ℓ; this is expected because AGs get larger and it becomes harder to predict the event participation—(url,time) pair—for each user in a group due to the added diversity. These trends are consistent with the trend observed for information loss (e.g., Figure 10(a)). Again, HL has the largest D_{KL} among other methods and D_{KL} is the same for HN and HI. As shown in Table VII, increasing w reduces D_{KL}; this implies that the anonymized data is more useful for predicting long term trends.

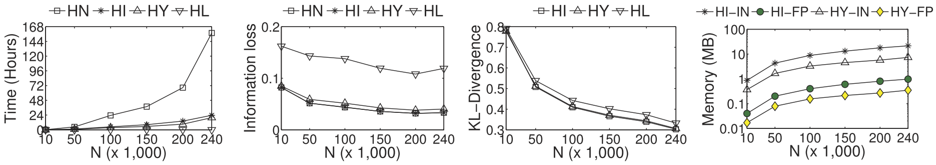


(a) Yahoo. (b) Oldenburg.

Fig. 12. KL-Divergence changing ℓ.

Table VII. KL-Divergence Changing w

Dataset	1 minute	1 hour	1 day	1 week	1 month
Flickr	1.680	0.670	0.327	0.147	0.049
Yahoo	2.100	0.628	0.316	0.102	0.027
Oldenburg	1.179	0.018	0.017	0.017	0.017
Worldcup	0.407	0.058	0.010	0.002	0.002



(a) Time. (b) NCP. (c) D_{KL}. (d) Memory usage.

Fig. 13. Scalability of k-anonymity in Mixed ($k = 16$).

8.4.4. Scalability. Figure 13 reports scalability evaluations for k-anonymity on Mixed dataset. All our algorithms scale better than HN (Figure 13(a)), which has a quadratic scalability. Except for HL, the information loss is relatively small (under 10%) for all methods (Figure 13(b)). HL is the fastest but it has a larger information loss. HI is an order of magnitude faster than HN and HY is 10%–20% faster than HI with a slightly larger information loss but almost equal D_{KL} (Figure 13(c)). HI offers the same quality as HN but runs significantly faster. Data density increases with the number of sequences. Thus, the sequences in each group become more similar and the information loss decreases monotonically for all methods (Figure 13(b)). This trend is consistent with the trend in D_{KL}; the distribution of events in anonymized data gets closer to the general distribution. HL is the fastest method, but has a larger information loss. HY and HI have the same data distortion as HN in Figure 13(b). We next measure the overhead of HI and HY (Figure 13(d)). HI and HY both require an index for clusters and memory to store the fingerprints. In Figure 13(d), we use HI-IN and HY-IN to refer to the size of index (on disk) for HI and HY, respectively. Similarly HI-FP and HY-FP refer to the storage space (in main memory) for fingerprints for HI and HY, respectively. HY has a smaller space requirement than HI, because HY only indexes sequences in region B. The space for fingerprints is under 1MB for 240K sequences with over 14M (event,time) pairs. This shows that HI and HY are practical for large datasets with a small overhead.

8.5. Evaluation of the SCR Setting

8.5.1. Time Generalization vs. (Event, Time) Generalization. We compared HI with ET (PR in Algorithm 4) on a sample of AOL dataset with 5K sequences. We set E_r to the set of all URLs to simulate the extreme setting of the SCR model, where all receivers may

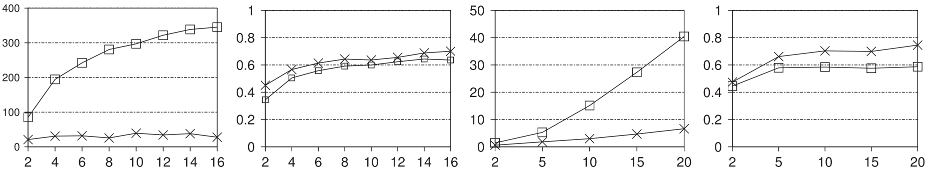
Table VIII. Time Generalization (HI) vs. Event and Time Generalization (ET) for a Sample of AOL, $w_t = w_e = 1$

	Time (Sec)				**Information loss**			
	$k = 2$	$k = 5$	$k = 10$	$k = 20$	$k = 2$	$k = 5$	$k = 10$	$k = 20$
HI	63.7	145.5	234.3	372.7	0.72	0.84	0.88	0.90
ET (PR)	121.1	263.8	398.5	565.9	0.81	0.89	0.92	0.94

collude. In this case, there is no gain beyond k-anonymity, because the receivers already have complete sequences. Thus, we do not consider (g, ℓ)-diversity in this experiment. The average length of sequences is 72 and we generalize URLs and time points. Table VIII reports information loss and running time for k-anonymity. Note that time generalization (HI) does not provide the same privacy level as ET because events can be used as QIDs to breach privacy. We include HI in Table VIII only to compare the information loss and running time of the two approaches. Since ET provides a more strong privacy coverage, it naturally incurs a larger information loss. Besides, it takes longer to anonymize data using ET. As ET generalizes every (time, event) pair of sequences, we observe a relatively larger information loss vs. HI (compare information loss with Flickr/Yahoo in Figure 8). Another factor for the large information loss in ET is the sporadic nature of click times combined with the large diversity of URLs. Even though ET uses progressive clustering, its running time is higher than HI due to PARTITION. Because ET is comparable with HN, in terms of the privacy of anonymized data, we next perform an in-depth comparison of HN and ET.

8.5.2. k-Anonymity. Figure 14 compares HN as our baseline approach (Algorithm 1) with Partition and Refine (PR) approach proposed in Algorithm 4. When k increases (Figure 14(a)), the overall running time of both methods grow for two reasons. First, the time complexity of calculating information loss grows linearly with the average number of sequences in each anonymization group (Section 7.1), which is correlated with k. Second, the number of under-filled clusters grows with k, which demands for more time to resolve under-filled clusters in both HN and PR. PR has a better running time, compared to HN, because not only computing *eventDist* is more efficient than *NCP*, but also refining over the *time* dimension (REFINE) is effective and takes advantage of the optimizations of HN for time dimension. The information loss of HN and PR are very close (Figure 14(b)); both grow with k. This is because as the size of each anonymization group grows, so does the number of (url, timestamp) pairs that need to be generalized together. This translates into larger distortion on both time and event dimensions. Using heuristics in PR does not have a great impact on the total running time of PR for a wide range of k. In the next experiment, we compared HN and PR by limiting the length of sequences in the dataset (varying *maxLen*) from 2 to 20 (Figures 14(c), (d)). For this experiment, each time we only used the first (i.e., earliest) *maxLen* (url, time stamp) pairs of each sequence. When *maxLen* is small, the difference between total information loss of the two methods is negligible (Figure 14(d)). This is mainly because the *eventDist* is not sensitive to the order of events, and as the sequence length grows, the partitioning places sequences with very similar events but possibly visited in different order in the same group. This, at the same time, reduces the running time of PR (Figure 14(c)) because partitioning across the event dimension is effective, and refining over the time dimension can take advantage of SSR optimizations.

8.5.3. Changing Weights w_t and w_e. We next compare HN and PR with the biased version of PR (Algorithm 6) when the relative importance of time (w_t) over event (w_e) decides the number of times REFINE or PARTITION is invoked. In this experiment, we start with a *small* $\frac{w_t}{w_e}$, which means preserving information about events is more

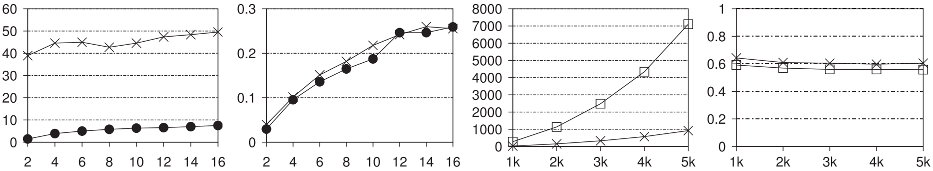


(a) Time(Sec.), changing k. (b) NCP, changing k. (c) Time(Sec.), changing $MaxLen$. (d) NCP, changing $MaxLen$.

Fig. 14. k-anonymity HN (-□-) (Algorithm 1) and PR (-x-) (Algorithm 4). Setting: $E_r = E$ and $|D|$ = 1K.

Table IX. Impact of Changing the Weights on Information Loss of Event and Time Stamp. Setting: $E_r = E$ and $|D|$ = 1k

	Avg. information loss (**event**)			Avg. information loss (**time**)		
w_t / w_e	PR	HN	Biased PR	PR	HN	Biased PR
0.01	0.9749	0.8557	0.9790	0.9668	0.8493	0.9724
0.1	0.9768	0.8551	0.9748	0.9012	0.7964	0.9078
1	0.9739	0.8714	0.9748	0.5537	0.4871	0.5602
10	0.9849	0.9352	0.9909	0.2105	0.1392	0.1660
100	0.9852	0.9858	0.9877	0.1350	0.0604	0.0937



(a) Time(Sec.), varying k. (b) NCP, varying k. (c) Time(Sec.), varying $|D|$. (d) NCP, varying $|D|$.

Fig. 15. (a, b) k-anonymity with time generalization. Methods: PR (-x-) (Algorithm 4) and HI (-•-) with our optimizations for time dimension. Settings: $w_e = 0$, $w_t = 1$, and $|D|$ = 1K. (c, d) Scalability of k-anonymity with increasing the number of sequences. Methods: HN (-□-) (Algorithm 1) and PR (-x-) (Algorithm 4). Setting: $E_r = E$ and k = 8.

important than time stamps. We increase $\frac{w_t}{w_e}$ up to a point where preserving temporal information is more preferred. For each value of $\frac{w_t}{w_e}$ we report the average information loss, once projected on the time dimension and once projected on the event dimension in Table IX. In HN, the amount of event distortion *increases* monotonically when $\frac{w_t}{w_e}$ grows and the average temporal distortion *decreases*. A similar pattern is observed in PR and Biased PR. We observe the main advantage of Biased PR over PR when $\frac{w_t}{w_e}$ is large ($\geq$ 10). In this range, the former method has smaller information loss across the time dimension. In the same region, Biased PR has a larger loss across the event dimension. In Table IX, the information loss projected on the event dimension is larger than that on the time dimension. This is attributed to a relatively small cardinality of event sets, compared to the space of possible time stamps. Because the domain taxonomy in our experiments was four levels, event generalization on such a shallow taxonomy turns out to be more coarse-grained, compared with time generalization.

Next, we compare our top-down PR algorithm with the bottom-up HI in a different setting; $w_e = 0$ and $w_t = 1$. In this setting, PR only performs time generalization and therefore it becomes comparable with HI in terms of the privacy level of the published data. As Figure 15(a), (b) present, HI is faster than PR and offers anonymized data with smaller information loss. This is mainly because PR produces several partitioning of the dataset until it forms groupings with $2k$ or less sequences. In each iteration, PR invokes Baseline optimized for the time dimension for partition sizes which are often *greater* than k (κ in Line 21 of Algorithm 4). Therefore, for the SSR setting, the algorithms optimized for time dimension are more effective than adapting the algorithm for SCR (i.e., PR, both considering the total running time and the utility).

8.5.4. Scalability. We set $k = 8$ and compared HN and PR when the size of database ($|D|$) grows (Figure 15(c) and 15(d)). The total information loss decreases for both methods (Figure 15(d)) because the data density grows with the size of database. While the information loss of HN remain very close to PR as size increases, the difference between the running time grows rapidly with size (Figure 15(c)). The difference is mainly attributed to the optimizations of SSR setting during REFINE in PR and the efficiency of computing *eventDist*, instead of performing $O(|D|^2)$ computations of information loss in HN.

8.6. Time and Event Generalization for the SSR Setting

Our algorithms for the SSR setting are based on time generalizations for events in E_r. We next investigate the idea of reducing information loss by generalizing events. As a motivating example, let E_r = {Google}, and consider two sequences S_1 = {(Google,1), (eBay,5), (Google,10)} and S_2 = {(Bing,1), (Bing,2), (Google,10)}. The SSR setting would publish the sequences as {(Google, [1–10]), (eBay, 5)} and {(Bing,1), (Bing, 2), (Google,[1–10])}, resp. An alternative approach to achieve 2-anonymity publishes

{(Search engine, 1), (eBay, 5), (Google, 10)} and {(Search engine, 1), (Bing, 2), (Google, 10)},

to preserve more of temporal content (in the above example, the original temporal information) by selectively generalizing some events in $E - E_r$. Note that this generalization is different from the SCR setting, because SCR does not generalize any event in $E - E_r$. To derive this new generalization, we have to modify Algorithm 7 as follows. Instead of considering the sequences in $\mathcal{L}$ for computing $IL(\mathcal{L}, I^*_{i,j})$ in Line 13, we need to consider a subset of (event, time stamp) pairs in $\mathcal{R}$ as well. Therefore, the computational complexity of finding $IL(\mathcal{L}, I^*_{i,j})$ is multiplied by the number of possible subsets of (event, time stamp) pairs in $\mathcal{R}$ that appear in range $[t_i - t_j]$. Thus, computing *NCP* and finding the optimal generalization of a set of sequences become more expensive compared to applying time-generalization only (i.e., the traditional SSR setting). Table X compares the results achieved by the modified algorithm ($\mathbb{ET}$) with the SSR setting with and without our performance optimizations ($\mathbb{T}_{HI}$ and $\mathbb{T}_{HN}$, respectively). We ran experiments once for E_r = {Google} and once for E_r = {Yahoo}. For each setting, we used different dataset sizes and values of k for k-anonymity. Each time that *NCP* is computed in $\mathbb{ET}$, we also compute *NCP* in the traditional SSR setting, and report the number of times that generalizing events improves (i.e., reduces) information loss. As we observe, for a very small number of cases (e.g., 34 in 10.3 million *NCP* computation for $|D|$ = 5K and k = 2) generalizing events in the SSR setting improves *NCP*. However, in the great majority of cases the *NCP* does not improved under the new policy. This is due to the fact that the cardinality of the temporal dimension is much larger than the number of events, in our experiments where the taxonomy of events has four levels. In this setting, one level of event generalization causes a large drop in utility, compared to generalizing time stamps to a wide time interval. On the other hand, the computational cost of $\mathbb{ET}$ is one to two order of magnitudes higher than $\mathbb{T}_{HN}$ and $\mathbb{T}_{HI}$. This is due to the computational complexity injected by the subset selection problem.

Table X. SSR; *time* ($\mathbb{T}$) vs. *event and time* ($\mathbb{ET}$) generalization. Setting: $maxLen = 20$

			E_r = {Google}			E_r = {Yahoo}		
$\lvert D \rvert$	k	method	*NCP*	$\frac{\#\text{better } NCP}{\#\text{total } NCP}$	time (sec.)	*NCP*	$\frac{\#\text{better } NCP}{\#\text{total } NCP}$	time (sec.)
5K	2	$\mathbb{ET}$	0.014	**34** / 10.3M	183.4	0.011	**2** / 10.3M	396.4
		$\mathbb{T}_{HN}$	0.014	-	12.3	0.011	-	11.0
		$\mathbb{T}_{HI}$	0.014	-	5.6	0.011	-	5.8
	8	$\mathbb{ET}$	0.068	**111** / 12.4M	1,010.5	0.055	**1** / 12.4M	1,144.2
		$\mathbb{T}_{HN}$	0.068	-	28.5	0.055	-	24.8
		$\mathbb{T}_{HI}$	0.068	-	11.9	0.055	-	11.4
10K	2	$\mathbb{ET}$	0.011	**120** / 41.5M	1,415.1	0.009	**6** / 41.6M	1,561.7
		$\mathbb{T}_{HN}$	0.011	-	50.2	0.009	-	42.6
		$\mathbb{T}_{HI}$	0.011	-	25.4	0.009	-	23.0
	8	$\mathbb{ET}$	0.054	**258** / 49.6M	4,080.7	0.044	**2** / 49.6M	4,630.6
		$\mathbb{T}_{HN}$	0.054	-	117.8	0.044	-	95.8
		$\mathbb{T}_{HI}$	0.054	-	54.2	0.044	-	46.2

9. RELATED WORK

9.1. Relational Data Anonymization

It has been shown that removing key identifiers from published data does not provide a comprehensive privacy protection [Sweeney 2002]. An attacker can join a subset of the attributes of data, termed as quasi-identifiers, with published data to reveal the existence of an individual in the published database or to find the value of sensitive attributes (SA) for an individual. An attacker can gain access to quasi-identifiers from public databases (e.g., voters registration list) or other sources of information. To protect privacy, a data provider transforms a database before making it available to third parties. The transformation step might generalize values into less specific values [Samarati 2001; Sweeney 2002], suppress some records [Sweeney 2002], perform perturbation by adding noise [Agrawal and Srikant 2000], or obfuscate the association between quasi-identifiers and SA [Xiao and Tao 2006]. In k-anonymity [Samarati 2001], a tuple must be indistinguishable in quasi-identifier space, among a set of k (or more) tuples called anonymization group (AG). In ℓ-diversity [Machanavajjhala et al. 2006], the values of the SA must be well-represented in each AG.

Partition-based anonymization may suffer from attribute inference attack if machine learning is used to learn associations between quasi-identifiers and SA [Kifer 2009]. To bound the probability of attribute inference attacks, in t-closeness [Li et al. 2007] the distribution of SA in each anonymization group must be close to the distribution of the SA in the original data. In minimality attack [Wong et al. 2007], it is assumed that the adversary knows about the privacy model, the anonymization algorithm, and the criterion being optimized (e.g., information loss). Wong et al. show that this adversary can breach privacy for a number of models including ℓ-diversity, t-closeness. For sensitive attributes which encodes either a "positive" or "negative" value, they propose a randomized algorithm to mitigate method based attacks (for binary $\ell-$diversity) by randomized grouping and sensitive value generalization. Cormode et al. [2010] analyze method-based attacks and define three properties that can make an anonymization algorithm vulnerable to minimality attacks: "being deterministic in operation, making asymmetric grouping choices, and jointly identifying the identifying and sensitive attributes." They show that symmetric partitioning (e.g., k-anonymity) and algorithms that only consider sensitive attributes (e.g., Anatomy [Xiao and Tao 2006]), can resist minimality attacks. For an algorithm designed deliberately to be vulnerable to minimality attacks, they show analytically that an attacker's confidence for the sensitive attribute of an individual may increase by a small constant. They conclude that the impact of such attacks can be minimized in practice.

9.2. Itemset Anonymization

The concept of k-anonymity has been also extended to set-valued data [He and Naughton 2009; Terrovitis et al. 2008, 2012]. An adversary's background knowledge in k^m-anonymity [Terrovitis et al. 2008] is confined to at most m items but no such limit is imposed in set k-anonymity [He and Naughton 2009]. Because a click-stream is a set of (URL, time) pairs, one can directly use k^m-anonymity [Terrovitis et al. 2008] (similarly k-anonymity [He and Naughton 2009]) to our problem by using an *augmented* taxonomy. The new taxonomy is a combination of the original event hierarchy with a time generalization tree. However, the common goal of Terrovitis et al. [2008] and He and Naughton [2009] is to ensure that after item generalizations any itemset known to an adversary is supported by at least k itemsets in the published data. The supporting itemsets may lack diversity, which makes Terrovitis et al. [2008] and He and Naughton [2009] vulnerable to SA inference attack.

To offer diversity, Xu et al. [2008] have proposed (h, k, p)-coherence. In this model, the itemsets which contain up to p public items, must have cardinality k (or more) and at most h percent of these itemsets contains a common private item. Our approach is different from Xu et al. [2008], due to two main properties of browsing histories: (1) temporal sensitivity, and (2) high data sparsity. First, each itemset can have more than one sensitive attribute, each corresponding to a different visit time. We define sensitivity not just in terms of visited URLs, but also visit time relative to other sequences that visited the same URL. Second, Xu et al. [2008] apply global item suppression to achieve (h, k, p)-coherence. However, unlike the Retail dataset used in Xu et al. [2008], the AOL dataset which contains browsing histories is very sparse; 68% of sequences have at least one unique (URL, time) pair, if we generalize time to the granularity of a month. Applying suppression to achieve (h, k, p)-coherence on browsing histories causes severe information loss, even when $p = 1$. We locally generalize time stamp and URLs to avoid catastrophic data distortion.

9.3. Multidimensional Anonymization

Multidimensional partitioning is the main impetus for Mondrian [LeFevre et al. 2006] and spatial partitioning [Iwuchukwu and Naughton 2007]. These greedy algorithms find a data transformations which supports k-anonymity by data partitioning. They start with one partition that contains the database as d-dimensional objects. They iteratively select a partition and divide it into two (or more) subpartitions, such that each subpartition has k or more objects. The algorithms end when there is no partition to split. LeFevre et al. [2006] provide a bound on the quality of anonymized data regarding the size of generated equivalence classes. To apply these works to our problem, we must first represent our dataset as multidimensional vectors. We map each click-stream into $(|E| \cdot |T|)^m$-dimensional binary vector, m being the maximum number of $(event, time)$ pairs in each click-stream. Thus, the database is transformed into a binary matrix. There are two issues when Mondrian is applied to anonymize the binary matrix. First, it is difficult to find a criteria to select a dimension to split (*choose-dimension* in [LeFevre et al. 2006]) because each dimension gets a value of one for a single record and zero for all other records. This is mainly due to the large sparsity of the click-stream datasets. Second, even if a criteria were defined to form anonymization groups, all that would be left after local-recoding each partition is a set of zeros (negative associations) and suppressed dimensions (i.e., 0 and 1 generalized to *) which poses a large amount of data distortion. Iwuchukwu and Naughton [2007] uses a spatial index structure (R-Tree) to maintain partitions. This approach suffers from the curse of dimensionality as $|E|$ and $|T|$ are very large in practice.

9.4. String Anonymization

Aggarwal and Yu [2007a] propose a condensation based method: they group similar strings and extract a statistical model for each group. The model captures the first and the second order distribution of characters in original data and it is used to produce—and publish—pseudodata with the same statistical distribution as the original strings. This property is desirable for aggregate data analysis (e.g., forming classifiers), but the published strings are not truthful to the original strings. Sketch based techniques have been used by Aggarwal and Yu [2007b] for privacy preserving data mining on text and itemsets. Sketches [Alon et al. 1996] have been applied to anonymize data [Aggarwal and Yu 2007b]. A number of primitive data mining operations (e.g., dot product and Euclidean distance can be estimated using sketches). But, estimating other general causality predicates can be quite complex over sketches.

9.5. Differential Privacy for Private Release of Query Logs

One way to anonymize click-streams is to represent them using a *histogram* and apply random perturbation techniques (e.g., [Götz et al. 2009; Korolova et al. 2009; Machanavajjhala et al. 2008; Xiao et al. 2010]). These techniques stem from the original work of Dwork et al. [2006]. To achieve provable privacy against a strong adversary, Dwork et al. [2006] add random noise to the cells in the histogram which is extracted from microdata. It is assumed that the adversary knows all records except one that belongs to a victim. The goal is to ensure that the adversary cannot make inference about the victim record with high confidence. Formally, for any two tables T_1 and T_2 that differ in one tuple and for any output O of randomized algorithm $\mathcal{A}$, the $\epsilon-$differential privacy ($\epsilon-$DP for short) requires that $\Pr\{\mathcal{A}(T_1) = O\} \leq e^{\epsilon} \cdot \Pr\{\mathcal{A}(T_2) = O\}$. Here, following Ref. [Dwork et al. 2006], we replace $\mathcal{A}$ by a transformation that maps the click-stream database into a histogram and adds random noise to the cells of the histogram. The noise is often generated using the Laplace distribution [Dwork et al. 2006; Götz et al. 2009; Korolova et al. 2009; Xiao et al. 2010] but other distributions can be used (e.g., the Multinomial distribution with a Dirichlet prior [Machanavajjhala et al. 2008]).

Rigorous privacy guaranty and analytical bounds on accuracy, for a specific measure of accuracy defined in Götz et al. [2009], are two desirable properties of perturbation based techniques that offer $\epsilon-$DP. To apply these techniques for our problem, we first simplify our data model and assume for now that each click-stream contains at most m $(event, time)$ pairs. This model is in fact the click model of Götz et al. [2009] augmented with time stamp of events. We map the database $\mathcal{D}$ of click-streams into a histogram $\mathcal{H}_m$ with $(|E| \cdot |T|)^m$ cells. Each click-stream in $\mathcal{D}$ contributes to exactly one cell of $\mathcal{H}_m$. It is straightforward to show that one can reconstruct $\mathcal{D}$ from $\mathcal{H}_m$ and that $\mathcal{H}_m$ can be used to answer all queries that can be answered by $\mathcal{D}$. Thus, the problem of private publication of $\mathcal{D}$ is equivalent to the private publication of $\mathcal{H}_m$.

As proposed in Korolova et al. [2009] and Götz et al. [2009], we add random noise to the cells of $\mathcal{H}_m$ to create a private release that offers privacy guarantees *close* to ϵ-DP. In fact, Götz et al. [2009] proves the infeasibility of ϵ-DP in search log publication; an algorithm that provides ϵ-DP is inferior to a naive algorithm that always outputs an empty set in two aspects: (1) the ability to retain very frequent terms and (2) the ability to filter-out very infrequent terms. Refs. [Machanavajjhala et al. 2008] and [Korolova et al. 2009] offer probabilistic variants of ϵ-DP, namely (ϵ, δ)-DP and (ϵ, δ)-Indistinguishability. Götz et al. [2009] prove that (ϵ, δ)-DP implies (ϵ, δ)-Indistinguishability but the converse is not true. They conclude that (ϵ, δ)-DP offers a

Table XI. Probability of Suppressing (i.e., Removing) a (URL,Time) Pair from a Click-Stream, when (ϵ, δ)-Differential Privacy is Applied to Anonymize AOL Dataset

ϵ	δ	λ	τ'	Pr(suppression)
2	$3.2 * 10^{-3}$	4.0	80.28	0.999
10	$1.3 * 10^{-37}$	0.8	80.20	1.000

stronger privacy compared to (ϵ, δ)-Indistinguishability. For this reason, in the rest of this section, we focus on (ϵ, δ)-DP and the threshold-based algorithm termed ZEALOUS [Götz et al. 2009]. The algorithm has three parameters: the thresholds τ, τ' and the variance of Laplace additive noise (λ). Given the histogram $\mathcal{H}_m$, ZEALOUS first filters (i.e., suppresses) each cell of $\mathcal{H}_m$ which is supported by less than τ records in $\mathcal{D}$. Then, it adds a Laplace noise to the support of each of the remaining cells of $\mathcal{H}_m$. Only cells with a noisy support of at least τ' are published (other cells are suppressed). For $\tau = 1$, an optimal setting for λ and τ' was proposed in Götz et al. [2009].

We present the result achieved by applying ZEALOUS to AOL dataset (refer to Section 8.1 for specification of this dataset). First, we built the histogram $\mathcal{H}_m$ for $m = 2$. Let $\chi = \max_i(\mathcal{H}_m[i])$ be a random variable that denotes the maximum count in the histogram. Because $\mathcal{D}$ is not empty, it must hold that $\chi \geq 1$. Clearly, $\Pr(\chi = 1) = 1 - \Pr(\chi > 1)$, where $\Pr(\chi > 1)$ is the probability that at least two click-streams are exactly equal. In practical settings, not all possible click-streams appear in a dataset and $|\mathcal{D}| \ll (|E| \cdot |T|)^m$. Therefore $\Pr(\chi > 1)$ is very small. For instance, in AOL dataset, $|E| = 812,031$, and $|T| = 2,906,507$, but $|D| = 650,000$. When we set $m = 2$ (less than the average length of click-streams which is 4), we have $\Pr(\chi > 1) = 1.7 * 10^{-25}$. Thus, a large fraction of entries in the histogram is either zero or one and $\tau = 1$ seems to be a reasonable choice. Table XI shows the probability that a click-stream is suppressed to support (ϵ, δ)-DP in the AOL dataset, where the probability of suppression is computed as follows: for each histogram cell $\mathcal{H}_m[i]$, $1 \leq i < (|E| \cdot |T|)^m$, with nonzero support, we compute the suppression probability $\Pr(\eta < \tau' - \mathcal{H}_m[i])$ directly, since η is generated by a Laplace kernel with variance λ. The high probability of suppression is the result of the inevitable uniqueness of click-streams in the AOL dataset.

A remedy, proposed in Machanavajjhala et al. [2008] for a similar problem is to shrink the domain using a privacy-preserving clustering algorithm that is compatible with differential privacy. However, this does not typically yield a bound on quality which is in fact a desirable property of the differential privacy framework. Ref. [Machanavajjhala et al. 2008] suggests to select initial partitions for shrinking from a publicly available data or a previous similar release. This data can reduce the sparsity problem for synthetic data generation problem, which is different from our problem.

Recently, Chen et al. [2012a] propose a variant of Götz et al. [2009], which instead of adding noise to the frequency of fixed length sequences, considers *variable length* subsequences. They use a prefix tree (similar to [Chen et al. 2012b]) for bookkeeping substrings and their frequencies. If a substring is not frequent enough, it is marked as leaf node and further expanding of it is terminated. Substrings induced from leaf nodes of the prefix tree suggest a Markov model to construct synthetic data. This work, while novel because of considering variable length substrings, has two limitations. First, for sparse data, a very large fraction of tree nodes, in the first level, could be marked as leaf nodes when time stamp comes into the picture. The MSNBC dataset which was used in Chen et al. [2012a] is not sparse for subsequences of length 5 (or less) for two reasons: (1) it has URL "category" instead of actual URLs, and (2) it does not have time stamps. Because 99% of (URL,time stamps) are unique in AOL dataset (see Section 1.2), except for the root node, a large fraction of the nodes of the prefix tree could be marked as leaf node in depth one. Sequential information will be lost in a synthetic dataset produced by the proposed algorithm of Chen et al. [2012a] when a large fraction of the nodes in prefix tree contains depth-1 leaf nodes. This issue may be avoided by employing a preprocessing step, similar to the method proposed in Machanavajjhala et al. [2008], to shrink the domain of events and time stamps, and consequently reduce data sparsity. Furthermore, the error measures used in Chen et al. [2012a] are not sensitive to the suppression of a relatively large number of not-so-frequent substrings, and the reported utility measure does not capture the suppressed subsequences that fall into the long tail of subsequence distribution. From an application point of view, publishing frequent substrings and itemsets is essential for index caching and query substitution (e.g., [Korolova et al. 2009; Zeng et al. 2012]). However, there are other applications (e.g., studying customer's purchase behavior [Goel et al. 2010]), with sparse *not-so-frequent* high dimensional data that fall in the long tail of distribution. An interesting problem, beyond the scope of this article, is to extend techniques for differential privacy (or at least relax the privacy model) to these domains to reduce the amount of suppression.

9.6. Privacy Preserving of Location-Based Services (LBS)

Our problem is loosely related to privacy protection in LBS, where the *location* of each user is required to offer personalized service (reply) to their request. To prevent user identification by location, several techniques have been proposed ([Chow and Mokbel 2011]). Existing solutions can be classified into three categories: (1) those that augment each request with *fake* (locations, requests) pairs (e.g., [Kido et al. 2005]), (2) those that cloak locations (e.g., [Cheng et al. 2006; Mokbel et al. 2006]), and (3) those that encrypt locations and rely on private information retrieval (e.g., [Ghinita et al. 2008]). Unlike our setting, these algorithms are applied in an online setting to protect privacy at the time of submitting requests. In contrast, our focus in this article is the offline setting; to protect privacy when sequences of requests are published.

9.7. Trajectory Anonymization

Anonymizing event sequences is a relatively new topic and has been mostly studied for trajectories (see [Ghinita 2009] and [Bonchi et al. 2011] for surveys). Ref. [Terrovitis and Mamoulis 2008] studies trajectory anonymization when the adversaries have disjoint and controlled (and known ahead) subtrajectories of the trajectories in the database to publish. One release is provided to all receivers. A privacy breach occurs if an adversary infers the location for a trajectory with a certainty above a threshold. This is similar to our (g, ℓ)-diversity model when the time gap g takes an infinitely large value. For RFID sequences, Fung et al. [2009] applied global suppression to produce anonymization with larger utility when a taxonomy is not available or using the taxonomy incurs a large information loss.

The main focus of trajectory anonymization [Abul et al. 2008; Nergiz et al. 2009; Yarovoy et al. 2009] is to protect published data against *sequence identification* attacks. Ref. [Abul et al. 2008] clusters trajectories that are similar along their entire time span. A regular sampling rate is assumed for all trajectories, which are generalized by spatial translation. Since all points of trajectories are regarded as quasi-identifiers, there is no notion of sensitive location in Abul et al. [2008]. Thus, there is no protection against event prediction attack. In Yarovoy et al. [2009], the quasi-identifiers of trajectories are a set of time points for each trajectory and each trajectory can have a different set of quasi-identifiers (QIDs). The adversary may have any subset of time points designated as QID for each trajectory. Anonymization groups (AGs) are formed by imposing a symmetric constraint on the attack graph to prevent sequence identification attack using location generalization. However, the trajectories in the same group may pass via the same location at close time stamps, which makes this approach vulnerable to event prediction attacks.

A combination of point generalization and suppression is practiced in Nergiz et al. [2009]. Each AG is generalized as a set of spatial and temporal intervals. Multiple sequence alignment is employed to find the optimal set of intervals for each AG. There are two limitations to this approach. First, each interval, extracted from a set of trajectories in the same AG, must cover exactly one point of each trajectory in the group. Thus, point suppression is inevitable if trajectories in the same group are of different length. Second, finding the optimal generalization for each anonymization group requires multiple sequence alignment which is NP-complete [Wang and Jiang 1994] and impractical even for small anonymization groups with long trajectories. To address these two limitations, we relaxed the exactly one requirement of Ref. [Nergiz et al. 2009] into at least one (Strong Coverage in Property 2.3). We achieve two important benefits from this relaxation, with no impact on privacy. (1) The optimal set of intervals for each anonymization group can be found in polynomial time (Lemma 1), and (2) the generalization does not suffer from point suppression.

10. CONCLUSIONS AND FUTURE WORK

We extended traditional privacy models to event sequences and proposed a framework based on time and event generalization. For the SSR setting, we proposed incremental index-based algorithms, in-memory summaries, and effective lower bounds. We also proposed a hybrid approach that takes advantage of sequence distribution to achieve a better performance without a significant drop in utility. For the SCR setting, we proposed a natural distance function for multisets which uses domain semantics modeled by a taxonomy. Our experiments on real and synthetic data show the efficiency of our algorithms, the scalability of our algorithms to large datasets, as well as the quality of our generated anonymizations compared to state-of-the-art methods for syntactic anonymization of sequence databases, in terms of our proposed information loss measure, range query distortion, and preserving causality patterns.

Previous results [Ghinita et al. 2007; Xu et al. 2006] show that Utility-based Data Partitioning (UDP) offer anonymizations with better quality vs. Multidimensional Partitioning (MP), such as Mondrian [LeFevre et al. 2006]. However, MP scales better with database size; $O(|\mathcal{D}| \cdot \log |\mathcal{D}|)$ vs. $O(|\mathcal{D}|^2)$ in UDP. In this article, for the first time, we took advantage of the properties of the utility measure and proposed algorithms to bridge the performance gap between UDP and MP. Given the rich literature of distance-based indexing techniques, our results suggest that UDP anonymization can be implemented as efficiently as MP for data types other than time-stamped sequences. As observed in our experiments, the performance improvement comes from two sources; reduced disk access and CPU cost; even if data is completely stored in main memory, still our optimizations remain effective as it reduces the number of expensive computations.

There are a lot of promising research directions left as future work. First, we plan to study event inference in a setting where an association between time and sensitive events can be learned from published data. Second, since generalization can retain data utility but suppression removes utility due to the skewness of data distribution, future work can investigate applying our methods as a preprocessing step for differential privacy and suppression-based algorithms (e.g., [Xu et al. 2008] and [Chen et al. 2012a]). Third, evaluating the utility of the anonymized data for applications (e.g., personalizing web search [Matthijs and Radlinski 2011]) is another interesting research direction. Finally, there are many challenging issues when user browsing histories are anonymized in a streaming scenario. A naive approach would be to slide a window over the browsing histories and run our algorithms on each split of data corresponding to each window. Data privacy and the utility for queries on causality patterns are important issues that depend on the size of the window and their possible overlap.

APPENDIX: VULNERABILITY OF ANONYMIZED DATA

We now investigate the vulnerability of anonymized data in the SCR setting to attacks aimed at recovering the (e, t) pairs for $e \in E_r$. For instance, let $E_r = \{\text{Google, Bing}\}$ and the Google and Bing share the original time stamps of their visits with each other. Having access to published data in Figure 2(b), Google learns that a user in the first partition visited Bing at 1, but does not know the user that makes this visit, nor (exactly) other events visited by that user. We do not consider learning about (e, t) pairs for $e \in E_r$ a privacy breach in the SCR settings. This is because in the extreme (and inevitable) setting, the colluding receivers may exchange the time stamps they collect along with users' identifiers (e.g., IP address). Even in this case, our anonymization guaranty in the SCR setting holds; the receivers cannot identify sequences using (e, t) pairs they exchange for $e \in E_r$. Sharing can lead to privacy breach in the SSR setting, as discussed in Section 1. Both sequence identification and event prediction attacks are possible if the main assumption of the SSR setting (i.e., not sharing time stamp) is violated by receiver(s) that engages in sharing data. However, this breach is not unique to click-streams and is inline with the general practice of privacy preserving data publications. Often, reasonable assumptions are made on the data generation process or the knowledge of an adversary, and there is a privacy risk if these assumptions are violated (e.g., [Ganta et al. 2008] studies the risk of privacy breach when data from multiple sources are shared). The impossibility of providing privacy and utility without making any assumption about the data generation process has been reported for differential privacy [Kifer and Machanavajjhala 2011].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments. Thanks to Nergiz et al. [2009] for providing the code of the trajectory anonymization (TA) algorithm, to Chen et al. [2012a] for providing the code of privacy-preserving publication of sequences using variable-length n-grams, and to Yilun Cai for implementing the symmetric anonymization (SA) algorithm [Yarovoy et al. 2009].

REFERENCES

Abul, O., Bonchi, F., and Nanni, M. 2008. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE)*.

Aggarwal, C. C. and Yu, P. S. 2007a. On anonymization of string data. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*.

Aggarwal, C. C. and Yu, P. S. 2007b. On privacy-preservation of text and sparse binary data with sketches. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*.

Agichtein, E., Brill, E., Dumais, S., and Ragno, R. 2006. Learning user interaction models for predicting Web search result preferences. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Agrawal, R. and Srikant, R. 2000. Privacy-preserving data mining. *SIGMOD Records 29,* 2, 439–450.

Alon, N., Matias, Y., and Szegedy, M. 1996. The space complexity of approximating the frequency moments. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*.

Barbaro, M. and Zeller, T. 2006. A face is exposed for AOL searcher no. 4417749. *The New York Times*. August 9.

Bayardo, R. and Agrawal, R. 2005. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*.

Bonchi, F., Lakshmanan, L. V. S., and Wang, W. H. 2011. Trajectory anonymity in publishing personal mobility data. *SIGKDD Explor. 13,* 1, 30–42.

Brinkhoff, T. 2003. Generating traffic data. *IEEE Data Eng. Bullet. 26,* 2, 19–25.

Chen, R., Acs, G., and Castelluccia, C. 2012a. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the ACM Conference on Computer and Communications Security*.

Chen, R., Fung, B. C. M., Desai, B. C., and Sossou, N. M. 2012b. Differentially private transit data publication: A case study on the montreal transportation system. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 213–221.

Chen, R., Fung, B. C. M., Mohammed, N., Desai, B. C., and Wang, K. 2013. Privacy-preserving trajectory data publishing by local suppression. *Inf. Sci. 231*, 83–97.

Cheng, R., Zhang, Y., Bertino, E., and Prabhakar, S. 2006. Preserving user location privacy in mobile data management infrastructures. In *Proceedings of the 6th Workshop on Privacy Enhancing Technologies*.

Chow, C.-Y. and Mokbel, M. F. 2011. Trajectory privacy in location-based services and data publication. *SIGKDD Explor. 13,* 1, 19–29.

Ciaccia, P., Patella, M., and Zezula, P. 1997. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on VLDB (VLDB)*.

Cormode, G., Srivastava, D., Li, N., and Li, T. 2010. Minimizing minimality and maximizing utility: Analyzing method-based attacks on anonymized data. *Proc. VLDB Endow. 3,* 1–2, 1045–1056.

Deshpande, M. and Karypis, G. 2004. Selective Markov models for predicting web page accesses. *ACM Trans. Internet Technol. 4,* 2, 163–184.

Ding, B., Winslett, M., Han, J., and Li, Z. 2011. Differentially private data cubes: Optimizing noise sources and consistency. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

Dupret, G. E. and Piwowarski, B. 2008. A user browsing model to predict search engine click data from past observations. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Theory of Cryptography Conference (TCC)*. Lecture Notes in Computer Science, vol. 3876. Springer, Berlin Heidelberg. 265–284.

Friedman, J., Bentley, J. L., and Finkel, R. A. 1977. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw. 3*, 209–226.

Fung, B., M. Cao, M., Desai, B., and Xu, H. 2009. Privacy protection for RFID data. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*.

Ganta, S. R., Kasiviswanathan, S. P., and Smith, A. 2008. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*.

Ghinita, G. 2009. Private queries and trajectory anonymization: A dual perspective on location privacy. *Trans. Data Privacy 2,* 1, 3–19.

Ghinita, G., Karras, P., Kalnis, P., and Mamoulis, N. 2007. Fast data anonymization with low information loss. In *Proceedings of the 33rd International Conference on VLDB (VLDB)*.

Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., and Tan, K.-L. 2008. Private queries in location based services: Anonymizers are not necessary. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

Goel, S., Broder, A. Z., Gabrilovich, E., and Pang, B. 2010. Anatomy of the long tail: Ordinary people with extraordinary tastes. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM)*.

Google. 2011. Google privacy FAQ. http://www.google.com/privacy/faq.html.

Götz, M., Machanavajjhala, A., Wang, G., Xiao, X., and Gehrke, J. 2009. Publishing search logs—A comparative study of privacy guarantees. *IEEE Trans. Knowl. Data Eng. 24*, 3, 520–532.

Guttman, A. 1984. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

Hadjieleftheriou, M., Kollios, G., Bakalov, P., and Tsotras, V. J. 2005. Complex spatio-temporal pattern queries. In *Proceedings of the 31st International Conference on VLDB (VLDB)*.

He, Y. and Naughton, J. F. 2009. Anonymization of set-valued data via top-down, local generalization. *Proc. VLDB Endow. 2,* 1, 934–945.

Huo, Z., Meng, X., Hu, H., and Huang, Y. 2012. You can walk alone: Trajectory privacy-preserving through significant stays protection. In *Proceedings of the 17th International Conference on Database Systems for Advanced Application (DASFAA)*. Lecture Notes in Computer Science, vol. 7238, Springer, Berlin Heidelberg. 351–358.

Iwuchukwu, T. and Naughton, J. F. 2007. K-anonymization as spatial indexing: Toward scalable and incremental anonymization. In *Proceedings of the 29th International Conference on VLDB (VLDB)*.

Jones, R., Kumar, R., Pang, B., and Tomkins, A. 2007. “I know what you did last summer”: Query logs and user privacy. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM)*.

Kido, H., Yanagisawa, Y., and Satoh, T. 2005. An anonymous communication technique using dummies for location-based services. In *Proceedings of the International Conference on Pervasive Services*.

Kifer, D. 2009. Attacks on privacy and definetti’s theorem. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

Kifer, D. and Gehrke, J. 2006. Injecting utility into anonymized datasets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

Kifer, D. and Machanavajjhala, A. 2011. No free lunch in data privacy. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

Koren, Y. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM 53,* 4, 89–97.

Korolova, A., Kenthapadi, K., Mishra, N., and Ntoulas, A. 2009. Releasing search queries and clicks privately. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*.

Kullback, S. and Leibler, R. A. 1951. On information and sufficiency. *Ann. Math. Stat. 22,* 1, 79–86.

LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. 2006. Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*.

Li, N., Li, T., and Venkatasubramanian, S. 2007. T-closeness: Privacy beyond K-anonymity and L-diversity. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*.

Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. 2006. ℓ-diversity: Privacy beyond K-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*.

Machanavajjhala, A., Kifer, D., Abowd, J. M., Gehrke, J., and Vilhuber, L. 2008. Privacy: Theory meets practice on the map. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*.

Machanavajjhala, A., Korolova, A., and Sarma, A. D. 2011. Personalized social recommendations - accurate or private? *Proc. VLDB Endow. 4,* 7, 440–450.

Mahdavifar, S., Abadi, M., Kahani, M., and Mahdikhani, H. 2012. A clustering-based approach for personalized privacy preserving publication of moving object trajectory data. In *Proceedings of the 6th International Conference on Network and System Security*. 149–165.

Mannila, H., Toivonen, H., and Inkeri, V. A. 1997. Discovery of frequent episodes in event sequences. *Data Mining Knowl. Discov. 1,* 3, 259–289.

Matthijs, N. and Radlinski, F. 2011. Personalizing Web search using long term browsing history. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM)*.

Meyerson, A. and Williams, R. 2004. On the complexity of optimal K-anonymity. In *Proceedings of the 23rd Symposium on Principles of Database Systems (PODS)*.

Mokbel, M. F., Chow, C.-Y., and Aref, W. G. 2006. The new casper: Query processing for location services without compromising privacy. In *Proceedings of the 28th International Conference on VLDB (VLDB)*.

Monreale, A., Andrienko, G., Andrienko, N., Giannotti, F., Pedreschi, D., Rinzivillo, S., and Wrobe, S. 2010. Movement data anonymity through generalization. *Trans. Data Privacy 3,* 2, 27–31.

Moon, B., Jagadish, H. V., Faloutsos, C., and Saltz, J. H. 2001. Analysis of the clustering properties of the Hilbert space-filling curve. *IEEE Trans. Knowl. Data Eng. 13,* 1, 124–141.

Nergiz, M. E., Atzori, M., Saygin, Y., and Güç, B. 2009. Towards trajectory anonymization: A generalization-based approach. *Trans. Data Privacy 2,* 1, 47–75.

Pang, H., Ding, X., and Xiao, X. 2010. Embellishing text search queries to protect user privacy. *Proc. VLDB Endow. 3,* 1–2, 598–607.

Papoulis, A. and Pillai, S. U. 2002. *Probability, Random Variables and Stochastic Processes*. McGraw Hill.

Samarati, P. 2001. Protecting respondents’ identities in microdata release. *IEEE Trans. Knowl. Data Eng. 13,* 6, 1010–1027.

Seidl, T. and Kriegel, H.-P. 1998. Optimal multi-step k-nearest neighbor search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

Sherkat, R. and Rafiei, D. 2008. On efficiently searching trajectories and archival data for historical similarities. *Proc. VLDB Endow. 1,* 1, 896–908.

Sweeney, L. 2002. K-anonymity: A model for protecting privacy. *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst, 10,* 5, 557–570.

Terrovitis, M. and Mamoulis, N. 2008. Privacy preservation in the publication of trajectories. In *Proceedings of the 9th International Conference on Mobile Data Management (MDM)*.

Terrovitis, M., Mamoulis, N., and Kalnis, P. 2008. Privacy-preserving anonymization of set-valued data. *Proc. VLDB Endow. 1,* 1, 115–125.

Terrovitis, M., Liagouris, J., Mamoulis, N., and Skiadopoulos, S. 2012. Privacy preservation by disassociation. *Proc. VLDB Endow. 5,* 10, 944–955.

Wagstaff, K. and Cardie, C. 2000. Clustering with instance-level constraints. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*.

Wang, L. and Jiang, T. 1994. On the complexity of multiple sequence alignment. *J. Comput. Biol. 1,* 4, 337–348.

Wong, R. C.-W., Fu, A. W., Wang, K., and Pei, J. 2007. Minimality attack in privacy preserving data publishing. In *Proceedings of the 29th International Conference on VLDB (VLDB)*.

Xiao, X. and Tao, Y. 2006. Anatomy: Simple and effective privacy preservation. In *Proceedings of the 28th International Conference on VLDB (VLDB)*.

Xiao, X., Wang, G., and Gehrke, J. 2010. Differential privacy via wavelet transforms. In *Proceedings of the 26th International Conference on Data Engineering (ICDE)*.

Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., and Fu, A. W.-C. 2006. Utility-based anonymization using local recoding. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*.

Xu, Y., Wang, K., Fu, A. W.-C., and Yu, P. S. 2008. Anonymizing transaction databases for publication. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*.

Yarovoy, R., Bonchi, F., Lakshmanan, L. V. S., and Wang, W. H. 2009. Anonymizing moving objects: How to hide a MOB in a crowd? In *Proceedings of the 12th International Conference on Expanding Database Technology: Advances in Database Technology (EDBT)*.

Zeng, C., Naughton, J. F., and Cai, J.-Y. 2012. On differentially private frequent itemset mining. *Proc. VLDB Endow. 6,* 1, 25–36.

Received September 2012; revised April 2013, July 2013; accepted September 2013