HKU CS Tech Report TR-2011-02

Efficient Timestamped Event Sequence Anonymization

Reza Sherkat, Jing Li, Nikos Mamoulis

Department of Computer Science, University of Hong Kong
Pokfulam Road, Hong Kong
{rsherkat, jli, nikos}@cs.hku.hk

March 14, 2011

Abstract

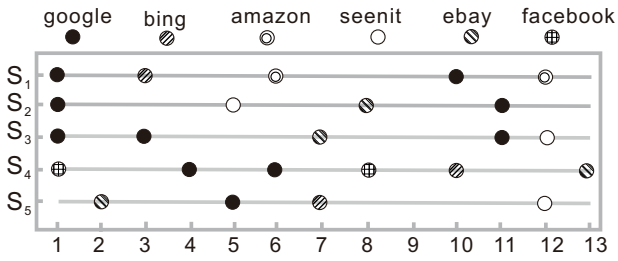
With the rapid growth of applications which generate timestamped sequences (click streams, RFID sequences, GPS trajectories), sequence anonymization becomes an important problem, if such data should be published or shared. Existing trajectory anonymization techniques disregard the importance of time or the sensitivity of events. This paper is the first, to our knowledge, thorough study on timestamped event sequence anonymization. We propose a novel and tunable generalization framework tailored to event sequences. We generalize timestamps 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; in this case, the receiver's background knowledge is confined to a set of timestamps and time generalization suffices, and (ii) sharing the data with multiple receivers; in this case, time generalization should be combined with event generalization. For both cases, we propose appropriate anonymization methods that prevent both user identification and event prediction. Extensive experiments confirm the efficiency and the scalability of our techniques and demonstrate the quality of the produced anonymizations.

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 Internet. Fig. 1 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 timestamps of user visits to their websites. For instance, Google can collect the timestamps shown in Fig. 1 (right), which correspond to Google visits of the click streams shown in Fig. 1 (left). Gaining access to the click stream data, collected by ISPs, enables a wide range of data analysis with social and commercial values. 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 [10]. The model can be used to better align services, e.g. search results and advertisements, towards identified emerging trends in users' interests [5, 11, 21]. However, sharing click streams can lead to serious privacy breaches, such as the notorious AOL privacy scandal [8], even if the identification information of individuals (e.g. IP address or name) are removed from the published data. We consider two practical scenarios in this paper:

Sharing with Single Receiver (SSR) - Assume that an ISP shares the click streams of Fig. 1 (left) with Google. Despite masking the real identifiers with random ones in Fig. 1 (left), the timestamps 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

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



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

Figure 1: Click streams of five users collected by an ISP (left) and the timestamps collected by Google (right)

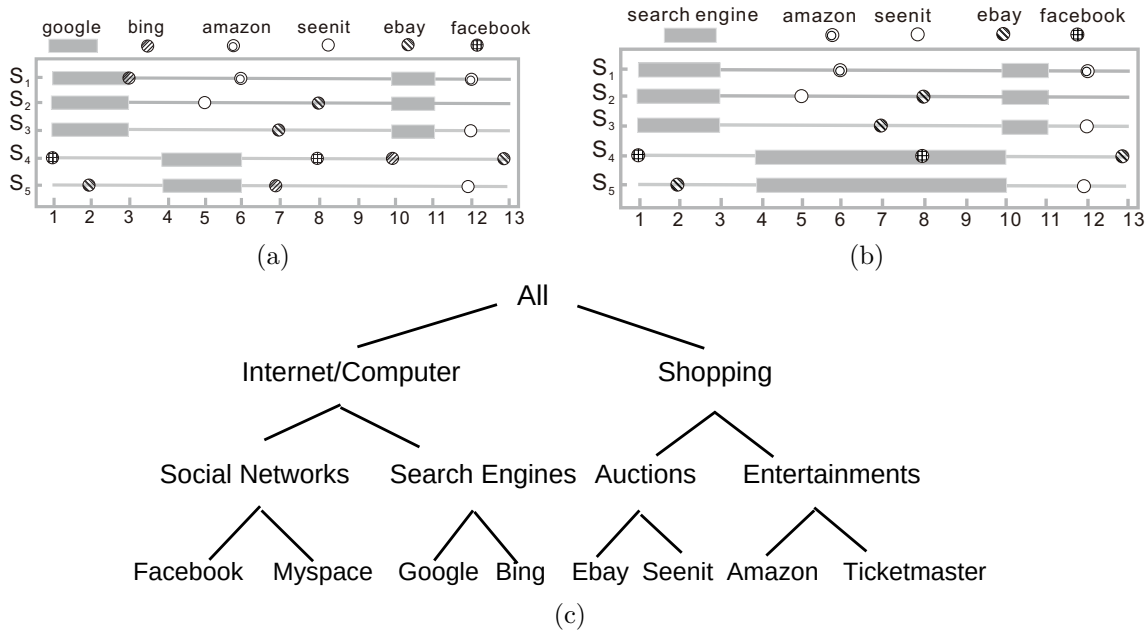


Figure 2: 2-anonymity and $(1, \frac{3}{2})$-diversity using time generalization (a) vs. event and time generalization (b). URL taxonomy (c)

only Sue and S_1 visited Google at time 1 and 10, thus sequence 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].

Timestamps can be used as *quasi-identifiers* (QIDs) 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 [30]), 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 privacy requirement (g, ℓ)-diversity: an extension of the ℓ-diversity model [24] 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 through timestamp linkage, an ISP can *anonymize* data by replacing time points with generalized intervals; we term this step *time generalization*. An anonymization for the click streams of Fig. 1 (left) is presented in Fig. 2(a); only the time points corresponding to Google visits are generalized (i.e., replaced by time intervals). For instance, sequence S_1 visited Google during [1–3] and

Table 1: 2-anonymity using event and time generalization

T_1 : (Search Engines, [1-3]), (Shopping, [5-8]), (All, [10-13])
T_2 : (Search Engines, [1-3]), (Shopping, [5-8]), (All, [10-13])
T_3 : (Search Engines, [1-3]), (Shopping, [5-8]), (All, [10-13])
T_4 : (All,[1-2]), (Internet/Computer,[4-8]), (All,[10-13])
T_5 : (All,[1-2]), (Internet/Computer,[4-8]), (All,[10-13])

[10–11]. In addition to the uncertainty introduced in the time points, the *number* of visits to Google within each interval and each sequence is lost[2]. In Fig. 2(a) Sue can be linked, with equal probability, to S_1, S_2, and S_3 because these sequences match with her regarding the timestamps 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 $\frac{2}{3}$. One can verify that Fig. 2(a) satisfies 2-anonymity and $(1, \frac{3}{2})$-diversity.

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 Fig. 2(a) with Bing. An ideal anonymization scheme must prevent attacks initiated by *any* subset of colluding receivers. To achieve this goal, we propose to reduce the certainty of both timestamps 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 Fig. 1 (left) to Google or Bing. Fig. 2 (b) demonstrates an anonymization which is provided to Google and Bing assuming that they share data. Not only the timestamp of visits to Google and Bing are replaced by intervals, but also the URLs are replaced by *Search Engine* from the taxonomy in Fig. 2(c). Any sub-sequence of $B_i \in \mathcal{B}$ matches with at least two sequences in Fig. 2(b).

In the extreme case, there is a risk that all parties collude, thus all timestamps and URLs need to be generalized, as shown in Table 1. Each anonymized sequence is a set of (*category*, *time-interval*) pairs, where categories are derived from the taxonomy in Fig. 2(c). Any sub-sequence of sequence S_i in Fig. 1 (left) matches with $\{T_j\}_{j=1}^{3}$ or $\{T_j\}_{j=4}^{5}$. Therefore, Table 1 is 2-anonymous.

In both SSR and SCR, one can find naive (legitimate) anonymizations, e.g. (ALL,[1-13]) for Fig. 1 (left). The naive solution reduces the utility of the anonymized data as 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 fulfill the desired privacy model.

This paper is the first work, to our knowledge, which studies the anonymization of timestamped event sequences using both time and event generalization. Although we adapt well-known anonymity models [30, 24] to timestamped event sequences, there are fundamental differences to problems studied in previous work, due to the lack of a fixed-length schema and a well-defined boundary between quasi-identifiers (QIDs) and sensitive attributes (SA). First, in relational data, an attribute range (e.g. Age:[20-25]) blurs only one attribute (Age) of each tuple, while 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 Table 1 generalizes two clicks of S_1 (i.e. (Google,1) and (Bing,3)) but one click of S_2. We provide a tailored definition of information loss which takes this flexibility into account. Second, the notion of SA in our setting is defined in relation with time points, whereas often an attribute (e.g. disease) is traditionally deemed sensitive. We propose a flexible privacy model which regards the sensitivity of event prediction along *any* time interval of a given length g.

Several works (surveyed in Sec. 2) have been proposed for trajectory anonymization to resist sequence identification attacks. The solutions (1) ignore the time points of sequences [31, 2], (2) use time points only

[2]The representation might be augmented to encode click statistics, e.g. the average number of clicks in each interval.

[3]This is the extreme case in which the colluding parties match the timestamps, which they collect, with real users.

for grouping sequences but only generalize location [37], or (3) perform time and location generalization but apply point suppression [27]. As we experimentally show, suppression reduces the utility of anonymized data 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 (temporal) 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). Our contributions are organized as follows:

- In Sec. 3, we propose a novel generalization scheme for event sequences: we apply both time generalization using time intervals and event generalization using event taxonomy. We propose a tunable information loss measure to quantify data distortion, based on the importance of time granularity or event diversity. We extend traditional privacy models to event sequences to model privacy for both sequence identification and event prediction attacks.
- In Sec. 4, we propose efficient algorithms to speed up anonymization for SSR setting using (1) cost-efficient lower bounds and compact summaries, (2) incremental index-based pruning, and (3) an effective hybrid approach to speed up our index based solution.
- In Sec. 5, we propose a partition-and-refine paradigm for SCR setting which uses: a novel taxonomy-aware distance function for event multisets, and our techniques proposed for SSR setting.
- In Sec. 6, we conduct extensive experiments to evaluate the quality, utility, efficiency, and scalability of our techniques on large-scale real and synthetic datasets from various domains.

Remark - In this paper, our discussion and examples are about click-stream data. However, our framework and techniques apply directly to other domains as well. For instance, click streams can be replaced with RFID sequences or trajectories, ISP with the Octopus company [1] or a central GPS tracker, and receiver with stores with loyalty card or an observer in a road network, respectively.

2 Related Work

Anonymizing event sequences is a relatively new topic and has been mostly studied for trajectories (see [14] for a brief survey).

Relational data anonymization - It has been shown that removing key identifiers from published data does not provide a comprehensive privacy protection [30]. 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, e.g. gossip. 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 [30], suppress some records [30], perform perturbation by adding noise to data [6], or obfuscate the association between quasi-identifiers and SA [35]. Several general models have been proposed to quantify data privacy. In k-anonymity [28], a tuple must be indistinguishable in quasi-identifier space, among a set of k (or more) tuples called anonymization group. However, k-anonymity is vulnerable to SA inference attack because there is no restriction on the distribution of the SA in each anonymization group. In ℓ-diversity [24], the values of the SA must be well-represented in each anonymization group. Partition-based anonymization may suffer from attribute inference attack if a machine learning approach is used to learn associations between quasi-identifiers and SA [19]. To bound the probability of attribute inference attacks, in t-closeness [23] the distribution of SA in each anonymization group must be close to the distribution of the SA in the original data.

Itemset anonymization - The concept of k-anonymity has been also extended to set-valued data [32, 18]. An adversary's background knowledge in k^m-anonymity [32] is confined to at most m items but no such limit is imposed in set k-anonymity [18]. The common goal of [32, 18] 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 contain (possibly sensitive) items not already included in the background knowledge. Furthermore, inside a support group, the distribution of the items *missing* from the background knowledge may diverge significantly from the global distribution of the same items in the published database, providing opportunities to missing item disclosure. This makes both approaches vulnerable to the infamous SA inference attack [24].

String anonymization - Aggarwal et al. [3] propose a condensation based method: they group similar strings and extract a statistical model from each group. The model captures the first and the second order distribution of characters in original data and it is used to produce - and consequently publish - pseudo-data with the same statistical distribution as the original strings. This property is desirable for aggregate data analysis, e.g. forming classifiers. However, the published strings are not truthful to the original strings. Due to favorable properties of sketches for sparse and high-dimensional data, sketch based techniques have been used [4] for privacy preserving data mining on text and transactional data. In particular, a family of sketches [7] has been applied to derive anonymous representation of data in [4]. A number of primitive data mining operations e.g. dot product and Euclidean distance can be estimated using sketches. However, estimating a complicated causality predicate can be quite complex on sketch representation.

Trajectory anonymization - Ref. [31] studies trajectory anonymization when the adversaries have disjoint and controlled sub-trajectories 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. [13] applied global suppression to produce anonymization with larger utility when a taxonomy is not available or using the taxonomy incurs a large information loss (e.g due to sparse quasi-identifiers).

The common focus of trajectory anonymization in [2, 37, 27] is to protect published data against *sequence identification* attacks. Ref. [2] 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 [2]. Thus, there is no protection against event prediction attack. In [37], 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 timestamps, which makes this approach vulnerable to event prediction attacks.

A combination of point generalization and suppression is practiced in [27]. 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 [34]. To address these two limitations, we relax the **exactly one** requirement of Ref. [27] into **at least one** (Strong Coverage property in Sec. 3.1). We achieve two important benefits from this relaxation, with no impact on privacy. (1) the optimal set of intervals for each AG can be found in polynomial time (Lemma 1), and (2) the generalization does not suffer from point suppression as each point will be always represented by one interval in our approach. Furthermore, we study both sequence identification and event prediction attacks in this paper.

3 Data Model and Problem Setting

Let E be the space of all possible events, e.g. the set of all URLs. The database $\mathcal{D}$ is a collection of timestamped event sequences. Each sequence $S = \{(e_i, t_i)\}_{i=1}^{|S|}$ in $\mathcal{D}$ is a set of $|S|$ timestamped events. Each pair $(e, t) \in S$ denotes the participation of S in event $e \in E$ at timestamp t. The background knowledge of a receiver is a database $\mathcal{B}$ of timestamped event sequences. We assume that $\mathcal{B}$ is subsumed by $\mathcal{D}$, i.e. 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 QIDs. We assume that only the timestamps of events in $E_r \subseteq E$ are collected by receiver(s). E.g. $E_r = \{\text{Google}\}$ in the SSR example of introduction and $E = E_r$ in the extreme case of the SCR setting. The set E_r imposes a constraint on each sequence $S_b \in \mathcal{B}$; $\forall (e,t) \in S_b, e \in E_r$.

3.1 Generalization model and information loss

To anonymize $\mathcal{D}$, we follow a partition-based approach [30]. We divide $\mathcal{D}$ into non-overlapping partitions $\mathcal{P}$[4] called anonymization groups (AGs). 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 (as described in Sec. 3.3), the generalization step is often independent from the partitioning step and it mostly focuses on (1) the indistinguishability of data in the QID space, and (2) the amount of information loss in each group.

3.1.1 Generalization model

We generalize each timestamp using a *time interval* and each event using a category from a taxonomy (e.g. Fig. 2(c)), only for event pairs (e,t) where $e \in E_r$. To improve data utility, we opt for *local recoding*; we do generalizations on the granularity of each 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 1 (Strong Coverage (SC)) *Let the set of intervals $\mathcal{I}_{\mathcal{S}}$ represent the AG $\mathcal{S}$. 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 $e \in E_r$; e is under category c_i and $t \in [s_i – e_i]$. There is no sequence in $\mathcal{S}$ with pair (e,t) which is not covered by any pair in $\mathcal{I}_{\mathcal{S}}$ if $e \in E_r$.*

Violating SC has two drawbacks: (1) not generalizing (at least) one timestamp of sequence $S \in \mathcal{S}$ may cause a direct identification of S, and (2) having an interval which does not cover (at least) one timestamp of S indicates a fake event participation for S, i.e. publishing wrong data. Note that the interval set $\mathcal{I}^n = \{(All, [\min_{S \in \mathcal{S}} start(S) - \max_{S \in \mathcal{S}} end(S)])\}$ satisfies the SC property, where $start(S)$ and $end(S)$ are, respectively, the smallest and the largest timestamp of S. Thus, there is at least one set of intervals for each AG that satisfies the SC property.

3.1.2 Information loss

We start by defining data distortion of a *single* interval along the time and event dimensions. Then, we expand this measure to a *set* of intervals. Finally, we define our information loss measure CP in Eq. 3. For sequence S we define the interval data distortion regarding the pair $I = (c_i, [s_i - e_i])$ as:

$$IDD(S,I) = \frac{w_t \cdot IL_t(S,I) \; + \; w_e \cdot IL_e(S,I)}{w_t + w_e}. \tag{1}$$

The weights w_t and w_e, respectively, capture the relative importance of time and event uncertainty. We assume that the weights are non-negative and $w_t + w_e > 0$. $IL_t(S,I)$ is the normalized data distortion due to generalizing the timestamps of (e,t) pairs in S using the time interval of I. $IL_t(S,I)$ is defined as:

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

$IL_e(S,I) \stackrel{def}{=} |c_i|/|E|$ is the normalized distortion due to generalizing event e_j using category c_i, $\forall (e_j, t_j) \in S$ s.t. $s_i \leq t_j \leq e_i$. $|c_i|$ is the number of events in taxonomy under c_i. IDD combines the distortions along time and event dimensions into a single measure with a maximum value of one. A smaller value of $IDD(S,I)$

[4]The set $\mathcal{P} = \{\mathcal{S}_1, \ldots, \mathcal{S}_{|\mathcal{P}|}\}$ is a partitioning of $\mathcal{D}$ if (1) $\cup_{i=1}^{|\mathcal{P}|} \mathcal{S}_i = \mathcal{D}$, and (2) $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for any $1 \leq i < j \leq |\mathcal{P}|$.

is more desirable and indicates a good representation for S within the time range of I. For a set of sequences $\mathcal{S}$, we integrate IDD to quantify the distortion due to generalizing sequences of $\mathcal{S}$ using $\mathcal{I}$:

$$IL(\mathcal{S},\mathcal{I}) = \frac{\sum_{I\in\mathcal{I}}\sum_{S\in\mathcal{S}}|S_I|\cdot IDD(S,I)}{\sum_{S\in\mathcal{S}}|S|} \tag{2}$$

where $S_I = \{(e,t) \in S | s_i \leq t \leq e_i\}$. Finally, we define our metric of *goodness* (homogeneity) for a set of sequences as follows:

$$CP(\mathcal{S}) = \min_{\forall\mathcal{I}} \ \big\{IL(\mathcal{S},\mathcal{I})\big\}. \tag{3}$$

Lemma 1 *$CP(\mathcal{S})$ can be computed in polynomial time.*

Proof 1 *Let $T_\mathcal{S} = \{t_i\}_{i=1}^{|T_\mathcal{S}|}$ be the set of timestamps in sequences of $\mathcal{S}$ and $i < j \Leftrightarrow t_i < t_j$. Let $n = |T_\mathcal{S}|$. For $t_i < t_j$, let $\mathcal{S}_{i,j}$ be the set of sequences induced from $\mathcal{S}$ by removing all (e,t) pairs in every sequence $S \in \mathcal{S}$ if $t < t_i$ or $t > t_j$. $[t_i$–$t_j]$ is* **not** *feasible for $\mathcal{S}$ ($[t_i,t_j] \nsim \mathcal{S}$), if for (at least) one sequence $S \in \mathcal{S}$, no (e,t) pair exists in S where $t_i \leq t \leq t_j$. We define Er:*

$$Er(\mathcal{S},\mathcal{I}) = \begin{cases} \infty & \exists\,(c_i,[s_i-e_i]) \in \mathcal{I},\ [s_i-e_i] \nsim \mathcal{S} \\ \sum_{I\in\mathcal{I}}\sum_{S\in\mathcal{S}}|S_I|\cdot IDD(S,I) & \textit{otherwise} \end{cases}$$

An interval $\mathcal{I}^*$ which minimizes Er also minimizes CP. Let $\mathcal{I}^*_{1,k}$ be the interval that (1) covers the time points $t_i|_{i=1}^k$ and (2) $Er(\mathcal{S}_{i,j},\mathcal{I}^*_{1,k}) = \min_{\forall\mathcal{I}} Er(\mathcal{S}_{i,j},\mathcal{I})$. One can verify that the following recursive equation holds for Er:

$$Er(\mathcal{S}_{1,n},\mathcal{I}^*_{1,n}) = \min_{1<k<n}\{Er(\mathcal{S}_{1,k},\mathcal{I}^*_{1,k}) + Er(\mathcal{S}_{k+1,n},\mathcal{I}^*_{k+1,n})\},$$

Therefore, Er (similarly CP) can be optimized efficiently using dynamic programming in $O\big(n^2 l_{min}(|\mathcal{S}|\log l_{avg} + \log n)\big)$ time using dynamic programming, where l_{avg} (l_{min}) is the avg. (min.) length of sequences in $\mathcal{S}$.

We use CP to quantify the amount of uncertainty which is introduced by anonymizing $\mathcal{D}$ using partitioning $\mathcal{P}$ into $\mathcal{D}^*_\mathcal{P}$:

$$NCP(\mathcal{D},\mathcal{P}) = \frac{\sum_{\mathcal{S}\in\mathcal{P}}|\mathcal{S}|\cdot CP(\mathcal{S})}{|\mathcal{D}|} \tag{4}$$

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

3.2 Privacy model and problem setting

We consider two types of attacks, sequence identification and event prediction. For the first attack, we use the traditional k-anonymity [30] as follows: no adversary can associate any sequence in $\mathcal{B}$ with less than k sequences in $\mathcal{D}^*_\mathcal{P}$. To quantify the risk of event prediction attack, we extend the ℓ-diversity model [24] to event sequences. First, we provide a formal definition for event prediction probability. For a set of sequences $\mathcal{S}$, the probability of observing event $e \in E - E_r$ during time interval $\mathcal{T}$ in any sequence of $\mathcal{S}$ is defined as: $\mathbf{P}(e,\mathcal{T}|\mathcal{S}) = \frac{|\{S\in\mathcal{S}\,|\,\exists\,(e,t_i)\in S,\ t_i\in\mathcal{T}\}|}{|\mathcal{S}|}$.

Example 1 *In Fig. 2(a), let $\mathcal{S}_1 = \{S_i\}_{i=1}^3$ and $\mathcal{S}_2 = \{S_j\}_{j=4}^5$. $\boldsymbol{P}(ebay,[6-9]\,|\,\mathcal{S}_1) = \frac{2}{3}$ and $\boldsymbol{P}(ebay,[1-3]\,|\,\mathcal{S}_2) = \frac{1}{2}$.*

Definition 1 ((g,ℓ)-diversity) *Let $E_r \subseteq E$ be the set of events monitored by the receiver(s). Let $\mathcal{P}$ be a partitioning of $\mathcal{D}$. $\mathcal{P}$ satisfies (g,ℓ)-diversity if $\mathbf{P}(e,\mathcal{T}_g|\mathcal{S}) \leq \frac{1}{\ell}$ for any event $e \in E - E_r$, 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. Parameter g gives time an equal importance as event identifiers when modeling the sensitivity of events. Thus, an event is not deemed sensitive *only* because it is in a set of sensitive events. If an adversary finds about an event but does not know its approximate timestamp, we do not consider this a privacy leakage. Def. 1 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. In the extreme case, $E_r = E$, un-known receivers may perform matching based on the identifiers (thus $\mathcal{B} = \mathcal{D}$). In this case, there is no gain in anonymization beyond k-anonymity. Otherwise, (g, ℓ)-diversity ensures that the certainty of an adversary about event participation for any event in $E - E_r$ is bounded by $\frac{1}{\ell}$ in any time interval of duration g or less.

Lemma 2 (Monotonicity of (g, ℓ)-diversity) *For two non-overlapping sets $\mathcal{S}_1$ and $\mathcal{S}_2$, any event e and time interval $\mathcal{T}_g$:*

$$\mathbf{P}(e, \mathcal{T}_g | \mathcal{S}_1 \cup \mathcal{S}_2) \leq \max\big\{\mathbf{P}(e, \mathcal{T}_g | \mathcal{S}_1), \mathbf{P}(e, \mathcal{T}_g | \mathcal{S}_2)\big\}.$$

Proof 2 *Without loss of generality let $P_1 \geq P_2$ where:*

$$P_1 = \boldsymbol{P}(e, \mathcal{T}_g | \mathcal{S}_1) = \tfrac{n_1}{|\mathcal{S}_1|}, \quad \textit{and} \quad P_2 = \boldsymbol{P}(e, \mathcal{T}_g | \mathcal{S}_2) = \tfrac{n_2}{|\mathcal{S}_2|}.$$

Thus $(n_1 + n_2) = \big(P_1 \cdot |\mathcal{S}_1| + P_2 \cdot |\mathcal{S}_2|\big) \leq \big(|\mathcal{S}_1| + |\mathcal{S}_2|\big) \cdot P_1$. Because $\mathcal{S}_1$ and $\mathcal{S}_2$ are non-overlapping, the following holds:

$$\boldsymbol{P}(e, \mathcal{T}_g | \mathcal{S}_1 \cup \mathcal{S}_2) = \frac{n_1 + n_2}{|\mathcal{S}_1 \cup \mathcal{S}_2|} \leq P_1 = \max\{P_1, P_2\}.$$

Problem 1 *Given the event set $E_r \subseteq E$, find a partitioning $\mathcal{P}$ of database $\mathcal{D}$, such that:*

- *The information loss (i.e. $NCP(\mathcal{D}, \mathcal{P})$) is minimized, and*
- *$\mathcal{P}$ satisfies the desired privacy model (i.e. k-anonymity or (g, ℓ)-diversity).*

3.3 Baseline: A constraint-aware partitioning

Problem 1 is NP-hard. If all sequences have the same length (greater than one), our problem reduces to microdata anonymity [25]. In k-anonymity, any partitioning algorithm (e.g. [36]) can be used to find partitions. In our experiments, we observed that these partitions may violate (g, ℓ)-diversity. Inspired by COP-COBWEB [33], we propose Baseline (Alg. 1) to partition $\mathcal{D}$ into anonymization groups (AGs), such that each group fulfills the desired privacy model (i.e. k-anonymity or (g, ℓ)-diversity).

COP-COBWEB is a hierarchical clustering algorithm, which supports instance-level constraints. An instance-level constraint between objects o_1 and o_2 does not allow two clusters C_1 and C_2 that contain these two objects to be merged. In our setting, if any cluster of $\mathcal{P}$ has at least k sequences, then a partitioning $\mathcal{P}$ fulfills k-anonymity. Thus, we do not need to define any constraint between clusters in k-anonymity, except for the size of clusters. For (g, ℓ)-diversity, we define an instance-level constraints as follows:

Definition 2 (constraints in (g, ℓ)-diversity) *Let clusters C_1 and C_2 be non-overlapping clusters, and $|C_1| < \ell$ and $|C_2| < \ell$. If $\big(e \in E - E_r \bigwedge (e, t_1) \in S_1 \in C_1 \bigwedge (e, t_2) \in S_2 \in C_2 \bigwedge |t_1 - t_2| \leq g\big)$, then we say that C_1 and C_2 violate an instance-level constraint, which is denoted as $(C_1, C_2) \in Con_{\neq}$ following [33].*

Intuitively, when $(C_1, C_2) \in Con_{\neq}$ and C_1 , C_2 are merged into $C_{1,2}$, $\mathbf{P}\big(e, [t_1 - t_2] \, | C_{1,2}\big) \geq \frac{2}{|C_{1,2}|} > \frac{1}{\ell}$; i.e. a violation of (g, ℓ)-diversity unless $C_{1,2}$ is merged later with another cluster. In Baseline we apply a greedy heuristic. In the first step, each sequence is assigned to a single cluster inside clusters group C. While C is not empty, we pick a cluster $C_i \in C$ at random. If C_i violates an instance level constraint with all clusters in C, we move C_i to the result set $\mathcal{P}$ to be processed later. Else, we find cluster $C_j \in C$ s.t. $CP(C_i \cup C_j)$ is minimum and C_j does not violate an instance-level constraint with C_i. We term this step $\mathbf{NN}(C_i)$. If $C_i \cup C_j$ has size at least k (ℓ) in k-anonymity ((g, ℓ)-diversity), we move the merged cluster to

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

1: Set τ to k if $\mathcal{M}$ is k-anonymity and to ℓ if $\mathcal{M}$ is (g, ℓ)-diversity
`/* Step 1: find the list of clusters` $\rightarrow \mathcal{P}$ `*/`
2: Set $C_i = \{S_i\}_{i=1}^{|\mathcal{D}|}$, set $C = \{C_i\}_{i=1}^{|\mathcal{D}|}$, and set $\mathcal{P} = \{\}$
3: **while** C is not empty **do**
4: Let C_i be a random cluster in C and $C_j = \text{NN}(C_i, C, \mathcal{M}, \texttt{TRUE})$
5: **if** C_j is empty **then** remove C_i from C and add it to $\mathcal{P}$
6: **else**
7: Remove C_j from C and merge it with C_i to make a new C_i
8: **if** $|C_i| \geq \tau$ **then** remove C_i from C and add it to $\mathcal{P}$
`/* Step 2) merging under-filled clusters */`
9: **for** each cluster $C_i \in \mathcal{P}$ with less than τ sequences **do**
10: **repeat**
11: $C_j = \text{NN}\big(C_i, \mathcal{P}, \mathcal{M}, \texttt{FALSE}\big)$
12: Remove C_j from $\mathcal{P}$ and merge it with C_i to a make new cluster
13: **until** C_i has at least τ sequences and satisfies privacy model $\mathcal{M}$
14: return $\mathcal{P}$
15: **function** NN(cluster C_i, cluster set C, privacy model $\mathcal{M}$, boolean *verify*)
16: Set $minDist = \infty$ and $C_k = \{\}$
17: **for** each cluster C_j in C **do**
18: **if** *verify* **and** $(C_i, C_j) \in Con_{\neq}$ **then** skip C_j ▷ According to $\mathcal{M}$
19: **else if** $minDist > CP(C_i \cup C_j)$ **then**
20: Set $C_k = C_j$ and $minDist = CP(C_i \cup C_j)$
21: return C_k

$\mathcal{P}$. Else, we put the merged cluster back into C. In the second step, each under-filled cluster $C_i \in \mathcal{P}$ is iteratively merged with its nearest clusters in $\mathcal{P}$, by calling $\mathbf{NN}(C_i)$, until all privacy constraints are satisfied. The monotonicity of k-anonymity ([30]) and (g, ℓ)-diversity (Lemma 2) ensure that BASELINE always finds a partitioning that meets the desired privacy model. In the extreme case, if only grouping all sequences into a single group fulfills the desired privacy model, the algorithm finds this grouping.

4 Optimizations for SSR

The search for the closest cluster to C_i in terms of CP, i.e. $\mathbf{NN}(C_i)$, is the bottleneck of BASELINE. An exhaustive search to find $\mathbf{NN}(C_i)$ requires $O(N)$ computation of information loss and loading disk resident sequences, where N is the number of active clusters in C or $\mathcal{P}$. 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. On the first direction, in Sec. 4.1, 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 Sec. 4.2. In the second direction, in Sec. 4.3 we propose a heuristic which uses the distribution of timestamps in sequences as a clue to (indirectly) reduce the number of **NN** calls. As we consider the SSR setting, only the timestamps of events monitored by the receiver(s) contribute to information loss. Thus, we extract summaries only from these timestamps. We later (in Sec. 5) use the techniques that we develop in this section as a module for the SCR setting.

4.1 Cluster summaries and lower bounds

Intuitively, if the lower bound of $CP(C_i \cup C_k)$ is greater than $CP(C_i \cup C_j)$, then cluster C_i is definitely closer to C_j than to C_k and evaluating $CP(C_i \cup C_k)$ is redundant. In this section, we briefly (see Appendix A

for detail) introduce two summaries, cluster fingerprint and cluster extent, and three lower bounds for information loss, D_{fp}, D_{Extent}, and D_{LB}. How the summaries and lower bounds are used to speed up **NN** is the subject of Sec. 4.2.

- **I. Cluster fingerprint -** A cluster fingerprint maps the timestamps of each cluster into a bitmap with b *buckets*. Each bucket represents a time interval which determines the resolution of the bucket, in terms of the number and the distribution of timestamps a bucket can capture. Fingerprints have three key properties. (1) they are compact, thus they can be stored in main memory. (2) when two clusters merge, the fingerprint of the new cluster can be computed by fast bit-wise operators. (3) $D_{fp}(C_i, C_j)$, the distance between the fingerprints of two clusters C_i and C_j, provides a lower bound of $CP(C_i \cup C_j)$ in $O(b)$ time.

- **II. Cluster extent -** A cluster extent represents a *set* of clusters as a minimum bounding rectangle (MBR). A cluster extent has three properties which make it useful for *batch pruning*. (1) $D_{Extent}(C_i, E_C)$, the distance of cluster C_i and extent E_C, can be used to find a lower bound of $CP(C_i \cup C_j)$ for cluster C_i and *any* cluster represented by extent E_C. (2) D_{Extent} is computed more efficiently, compared to CP. (3) cluster extents can be organized by an index structure to guide pruning process (Sec. 4.2).

- **III. Cluster-level lower bound -** We propose a simple greedy heuristic to efficiently find a lower bound for $CP(C_i \cup C_j)$. Assume that only two sequences R and S are in $C_i \cup C_j$. Each timestamp t in R must fall inside at least one interval (due to the SC property) when $CP(C_i \cup C_j)$ is computed. The distortion of timestamp t in R is at least $|t - nearest(t, S)|$, where $nearest(t, S)$ is the closest timestamp of sequence S to t. By induction, we can aggregate this loss to find D_{LB} as another lower bound of CP.

4.2 Indexing clusters to boost NN search

We propose an index structure for clusters and an incremental search algorithm to speed up **NN** search. We organize clusters using the R-Tree [16] structure as follows: the internal nodes of the index are cluster extents. The cluster extent for the root node represents 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 cluster fingerprints in the main memory. We observed a negligible space overhead for fingerprints in our experiments. The overhead can be tuned to available memory.

We extend the the multi-step nearest neighbor search [29] as FastNN (Alg. 2). Given cluster C_i as query, our algorithm works as follows. A list of visited *objects* is kept in a heap along with a *distance* for each object. An object can be a cluster extent, a cluster, or a fingerprint. First, we push to the heap the cluster extent of the root node and $D_{Extent}(C_i, root)$ as distance. While the heap is not empty, we deheap an object p and process it based on the type of p. 1.a) If p is a cluster extent, we push to the heap every cluster extent q that is covered by p, along with $D_{Extent}(C_i, q)$. 1.b) If p is a leaf node, we add the fingerprint of each cluster q covered by p to the heap along with $D_{fp}(p, q)$. (2) If p is a fingerprint, we load the sequences of cluster p into cluster C_p, and push C_p along with $D_{LB}(C_i \cup C_p)$ to the heap. 3.a) If p is deheaped as a cluster for the first time, we push it back to the heap with $CP(C_i \cup p)$. 3.b) Cluster p is returned as the closest cluster to C_i if this is the second time that p is deheaped as an object of type cluster.

4.3 Hybrid partitioning to reduce NN calls

Indexing improves Baseline by improving the performance of **NN**. In this section, we take an orthogonal approach and propose a *data driven heuristic* to *reduce* the number of **NN** calls. The method we propose here can still benefit from FastNN. The main idea (Alg. 3) is to partition data into two regions, namely A and B. We apply a fast partitioning method (e.g. [15]) on region A and Baseline on region B, and integrate the results.

Algorithm 2 FastNN(cluster C_i, cluster index R)

Initialize an empty heap $\mathcal{H}$
Push $\big(\mathcal{H}, root, D_{Extent}(C_i, root), \texttt{Extent}\big)$ ▷ *root*: the root node of R
while $\mathcal{H}$ is not empty **do**
    Pop an object from $\mathcal{H}$ into p
    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 $\big(\mathcal{H}, q, D_{Extent}(C_i, q), \texttt{Extent}\big)$
    else if type of object p is `FP` **then**
        Load time sequences for cluster p into C_p
        if $(C_i, C_p) \notin Con_{\neq}$ **then** Push $\big(\mathcal{H}, C_p, D_{LB}(C_i, C_p), \texttt{LB}\big)$
    else if cluster p is de-heaped for the 1^{st} time **then**
        Push $\big(\mathcal{H}, p,\ CP(C_i \cup p),\ \texttt{CP}\big)$
    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

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

$(\mathcal{D}_A, \mathcal{D}_B)$ = AssignRegions($\mathcal{D}$) ▷ (Sec. 4.3)
$\mathcal{P}_A$ = FastCluster($\mathcal{D}_A$) ▷ Using E.g.[15]
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($\mathcal{D}_B$, $\mathcal{M}$) ▷ Alg. 1
return $\mathcal{P}_A \cup \mathcal{P}_B$ ▷ Merge $\mathcal{P}_A$ and $\mathcal{P}_B$

4.3.1 Intuition

Fig. 3 shows a mapping of time sequences into points in 2D space (recall that our focus is the SSR setting). The coordinates correspond to the first and the last timestamp 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 [26] 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 Fig. 3. We refer to the area *close* to the diagonal line in the 2D space as *region A* and the top-left corner as *region B*. We describe shortly how to assign sequences to each region. Through experiments on real and synthetic data **we observed** that for sequences in *region A* (S_1, S_2, and S_3) the difference between $hIndex$ is correlated with the information loss of the corresponding cluster. 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 and the diversity of timestamps increases. No consistent correlation is observed between the closeness of $hIndex$ of two sequences and the information loss in region B. For instance, $|hIndex(S_4)\text{-}hIndex(S_5)| < |hIndex(S_4)\text{-}hIndex(S_7)|$, but S_4 is grouped with S_7. **We observed** that a majority of sequences in region A are clustered with sequences in the same region. Thus, an ordering of sequences in region A based on $hIndex$, instead of information loss, should bring closer all sequences that may fall in the same cluster. We cluster sequences in region A using a fast heuristic (described later) and run Baseline only for sequences in region B.

ID	*timestamps*	*hIndex*	*region*
S_1	57,67	39	*A*
S_2	53,57	34	*A*
S_3	41,50	32	*A*
S_5	18,40,74	22	*B*
S_6	7,32,64	20	*B*
S_4	18,65	23	*B*
S_7	17,55	18	*B*

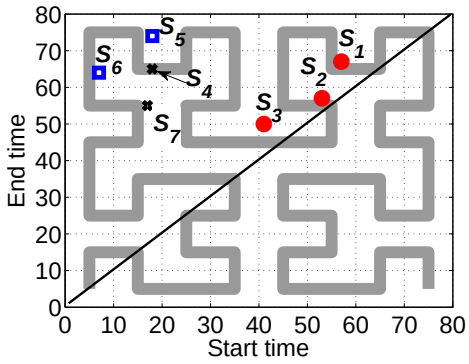


Figure 3: Mapping sequences to 2D (start, end) space

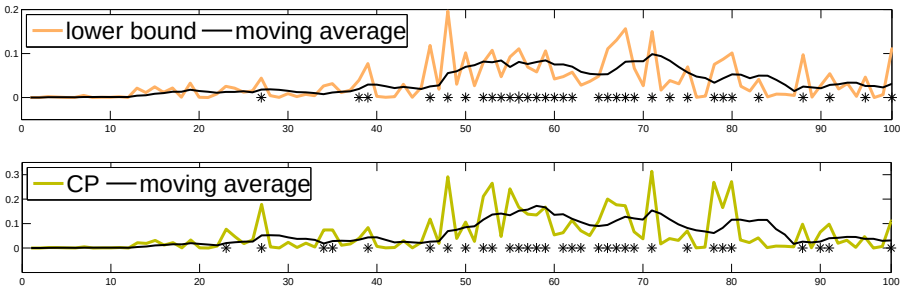


Figure 4: Pairwise D_{LB} and CP for 100 sequences, the sequences assigned to region B are marked by *.

4.3.2 Assigning sequences to regions

We sort sequences by $hIndex$. We assign sequence S_i to region A if $CP(\{S_{i-1}, S_i\}) < \sum_{i=1}^{N} CP(\{S_{i-1}, S_i\})/N$ or $|S_i| = 1$, where $N = |\mathcal{D}|$. Else, S_i is assigned to region B. For S_1 we compute $CP(\{S_1, S_2\})$. This heuristic needs $O(N)$ evaluations of information loss, which is computationally expensive when either N or the average length of sequences is large. Fig. 4 shows CP and lower bound 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 Fig. 4 seems to be correlated with the region assignment based on CP. Thus, we use D_{LB} to speed up the region assignment step of HYBRID.

4.3.3 Clustering 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 [15] to find an optimal grouping in $O(\tau \cdot N_A)$ time where N_A is the size of region A. In k-anonymity, all clusters found can be finalized unless when $N_A < k$. In this case we process all sequences in region A with the sequences in region B. Since during dynamic programming we only check the QID part of sequences, (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 $O(g \cdot n_c)$ time, where n_c is the number of event pairs of cluster C. If a cluster satisfies (g, ℓ)-diversity, it is finalized. Otherwise, the sequences of this cluster will be handled together with

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

1: Set τ to k if $\mathcal{M}$ is k-anonymity and to ℓ if $\mathcal{M}$ is (g, ℓ)-diversity
2: Set $\mathbb{C}$ to a cluster containing all sequences of $\mathcal{D}$
3: $\mathcal{P}$ = PARTITION($\mathbb{C}$, τ) ▷ start the recursive drill down
`/* resolve privacy violations of clusters in P*/`
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 $\mathcal{P}$
`/* splitting along the event dimension */`
10: **function** PARTITION(cluster C, τ)
11: **if** C has less than 2τ 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
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
17: **else** Add S_i to C_2
18: return REFINE(C_1, τ) $\cup$ REFINE(C_2, τ)
`/* Splitting along the time dimension*/`
19: **function** REFINE(cluster C, τ)
20: Set $\kappa = \lfloor |C|/2 \rfloor$
21: Split C to κ-anonymous clusters C_1, C_2 ▷ BASELINE and FASTNN
22: return PARTITION(C_1, τ) $\cup$ PARTITION(C_2, τ)

all sequences in region B by BASELINE.

5 Optimizations for SCR

We propose a greedy top-down heuristic to derive AGs, based on the construction step of kd-trees [12]. We recursively reduce the information loss; each time along either the event dimension or the time dimension. To speed up our approach, we take advantage of (1) our optimizations for SSR setting (Sec. 4), and (2) our fast-to-compute taxonomy-based distance, described in Sec. 5.2.

5.1 Top-down partition and refinement

We give an overview of our approach (Alg. 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 Sec. 5.2) and ignoring the timestamps. 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 timestamps and ignoring events. Then, it calls PARTITION for each split cluster. (3) 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. (4) We use the same heuristic as the second step of BASELINE to resolve the privacy violations of clusters, if any, by merge to finalize the AG.

Intuitively, REFINE reduces information loss along the time dimension only. Thus, we use the optimizations of SSR (Sec. 4) to speed up the refinement step. 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 than to S_2 is assigned into C_1. Other sequences are assigned to C_2. As a distance metric for sequences, we can use CP with $w_t = 0$ (note: our focus is event similarity). However, CP is expensive to compute. Thus, we propose a fast to compute alternative to CP which regards event taxonomy.

5.2 *eventDist*: A fast taxonomy-based distance

We slightly modify each sequence by *removing* the order of events in each sequence. This produces a distance measure for sequences that captures the diversity of events w.r.t. a taxonomy, and the frequency distribution of events. From sequence S, we derive multiset $M_S = \{(e_i, f_i)\}_{i=1}^{|M_S|}$ where $f_i = |\{t|(e_i, t) \in S\}|$ is the frequency of event e_i in sequence S. For instance, the multiset of sequence S_1 in Fig. 1 (left) is {(Google,2),(Bing,1),(Amazon,2)}.

Several set similarity measures have been proposed in the literature, e.g. the Jaccard coefficient and the cosine measure. However, these measures ignore the *semantics* of an application domain modeled by a taxonomy. The similarity of two multisets M_1 = {(Facebook,1), (Myspace,1),(Google,1),(eBay,1),(Seenit,1)} and M_2 = {(Bing,1), (Facebook,2),(Amazon,1)} must capture not only the number of common events (i.e. Facebook) but also the number of events under the same categories (e.g. Google and Bing) which are both under *Search Engines*.

We propose *eventDist* as the *minimum cost* of using the categories in the taxonomy to make the events of the two multisets equal. For multiset M and category c_i in the event taxonomy, let $n(M, c_i)$ be the sum of the frequency of events in M that are covered by 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.

Definition 3 (Cost of Match) *The cost of matching two multisets A and B, given the set of categories $\mathbb{C} = \{c_i\}_{i=1}^{|\mathbb{C}|}$ is:*

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

Example 2 *Let c_1 = Social networks, c_2 = Search engines, c_3 = Shopping, and c_4 = Amazon for the taxonomy of Fig. 2(c). For M_1, $n(M_1, c_1) = 2$, $n(M_1, c_3) = 2$, and $n(M_1, c_4) = \infty$. For M_2, $n(M_2, c_1) = 2$, $n(M_2, c_2) = 1$, and $n(M_2, c_4) = 1$. Thus, $cost(M_1, M_2, \{c_i\}_{i=1}^{3}) = \frac{25}{72}$, $cost(M_1, M_2, \{c_i\}_{i=1}^{4}) = \infty$.*

Definition 4 (Computing Event Distance) *For two sequences S_1 and S_2,*

$$eventDist(S_1, S_2) = \min_{\forall \mathbb{C} \in 2^H} \big\{cost\big(M(S_1), M(S_2), \mathbb{C}\big)\big\},$$

where $M(S)$ is the multiset of sequence S and H is the taxonomy.

Finding *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 best $\mathbb{C}$. However, some checkings are *redundant* and can be pruned effectively by branch and bound using: (1) an ordered list of events in multisets, and (2) the structure imposed by taxonomy H. To efficiently compute *eventDist*, we first assign an *id* to each event in E by performing a depth-first traversal on H (the *numbers* under URLs in Fig. 5). We then represent each multiset using *an ordered list* of event ids. For instance, M_1 = {`(Facebook,1), (Myspace,1), (Google,1), (eBay,1), (Seenit,1)`} and M_2 = {`(Bing,1), (Facebook,2), (Amazon,1)`}, respectively, becomes {(1,1), (2,1), (3,1), (5,1), (6,1)} and {(1,2), (4,1), (7,1)}. We use Alg. 5 to prune the search space for optimal set of nodes using the taxonomy H and the order of event as a yardstick. We use a running example to convey the intuition behind Alg. 5.

Example 3 *Fig. 5 reproduces the same taxonomy of Fig. 2(c), with node labels replaced by $\pi_{i,j}$ for brevity, and the frequency of events are not shown for brevity. To find the distance of multisets M_1 and M_2, we*

Algorithm 5 EVENTDIST(Multiset A, Multiset B, Node π)

1: **if** both A **and** B are empty **then** return 0 ▷ Equal multisets
2: **if** either A **or** B is empty **then** return ∞ ▷ Either one (not both)
3: **if** π is a leaf node **then** return COST(A, B, π) ▷ Case 1 (Def. 3)
4: Set sp to the maximum event Id in LEFTSUBTREE(π) ▷ sp : split point
5: Split A using sp into two non-overlapping multisets A_1 and A_2
6: Split B using sp into two non-overlapping multisets B_1 and B_2
7: **if** both A_1 and B_1 are empty **then** ▷ Case 2.a
8: return EVENTDIST$(A_2, B_2,$ RIGHTSUBTREE$(\pi))$
9: **if** both A_2 and B_2 are empty **then** ▷ Case 2.b
10: return EVENTDIST$(A_1, B_1,$ LEFTSUBTREE$(\pi))$
11: **if** *any* one of A_1, A_2, B_1, or B_2 is empty **then** ▷ Case 3
12: return COST(A, B, π) ▷ Using Def. 3
13: $c_1 =$ EVENTDIST$(A_1, B_1,$ LEFTSUBTREELEFT$(\pi))$
14: $c_2 =$ EVENTDIST$(A_2, B_2,$ RIGHTSUBTREE$(\pi))$
15: return $(c_1 + c_2)$ ▷ Case 4

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 *sub-tree of $\pi_{4,1}$ can be applied to both $M_{1,1}$ and $M_{2,1}$. Likewise, only the nodes in the right sub-tree of $\pi_{4,1}$ can be applied to both $M_{1,2}$ and $M_{2,2} = \{7\}$. Thus, we need to recursively call eventDist (Case 4 of Alg. 5); once with $M_{1,1}$, $M_{2,1}$, and $\pi_{3,1}$ and once with $M_{1,2}$, $M_{2,2}$, and $\pi_{3,2}$. In the first recursive call, shown in Fig. 5, there are 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 $cost(M_{2,1}, M_{2,2}, \pi_{3,2})$, which consequently returns $\frac{25}{72}$ as $eventDist(M_1, M_2)$. 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 Fig. 5). Thus, we return $cost(M_1, M_2, \pi^*)$ as the eventDist of sequences corresponding to multisets M_1 and M_2.*

Alg. 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 $B_i|_{i=1}^{d(n_i)}$. **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* non-empty, A and B must be split recursively (similar to Case 4 of Alg. 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 i^{th} 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 the leaf node (Case 1 of Alg. 5). Alg. 5 computes the distance in a time linear to the number of events of the multisets. In the AOL dataset (Sec. 6), the taxonomy of URLs has only four levels and fits in main memory.

Lemma 3 (Computing Event Distance) *For two sequences A and B, an event taxonomy of height h and maximum fan-out d, $eventDist(A, B)$ can be computed in $O(|A| + |B| + nhd)$ time, where n is the maximum number of distinct events in A and B.*

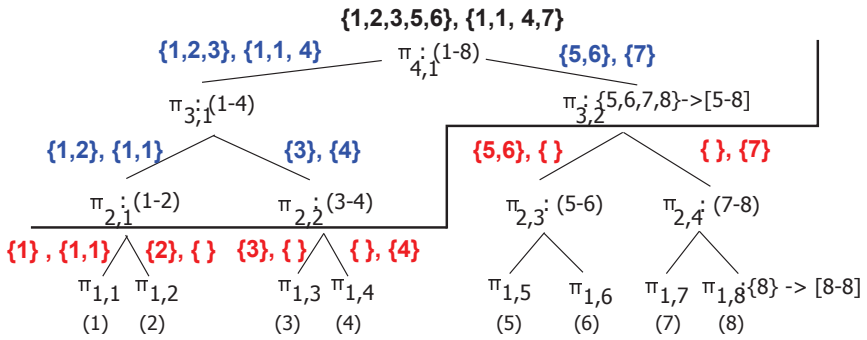


Figure 5: Pruning taxonomy nodes for *eventDist*

Table 2: Methods used in experimental study

Method	Description
TA	Trajectory Anonymization algorithm [27]
SA	Symmetric Anonymization algorithm [37]
HN	BASELINE(Alg. 1)
HI	HN with the optimizations of Sec. 4.1, 4.2
HY	HN with the optimizations of Sec. 4.3
HL	HY if all sequences are assigned to region A
ET	Our greedy top-down heuristic of Sec. 5

Proof 3 *Alg. 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* $O(|A| + |B|)$ *in average. Let* n *be the maximum number of distinct events in* A *or* B*. In the extreme case, Alg.5 assigns a leaf category (at height* h*) to each event, making the cost of traversing to* $O(hdn)$*. This is because at each step, for a balanced taxonomy of height* h *and a maximum fan-out* d*, a loop is required to split* A *and* B *into* $d(\pi_i) \leq d$ *splits. Thus, the total time complexity of Alg. 5 is* $O(|A| + |B| + hdn)$*.*

6 Experimental Evaluation

We present the result of an empirical evaluation of our algorithms on real and synthetic data. We compare efficiency, information loss, reduction in computations and disk access, as well as the quality of the generated anonymizations. Table 2 summarizes the methods used in our empirical study.

6.1 Datasets

Table 3 summarizes the datasets used in our experimental study:

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 [18]. 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 ={Yahoo} and E_r ={Flickr}, respectively, in datasets Yahoo and

Table 3: Summary of datasets used for evaluations

property	Flickr	Yahoo	Mixed	Oldenburg	Worldcup
No. of Sequences	6,919	33.4K	243.4K	11,571	19.4K
Size of region A	6,074	25.4K	159.1K	8,060	15.6K
Max. length	54	867	8.6K	186	685
Avg. length	1.2	3.8	4.9	7.0	14.3
(e, t) pairs	1.5M	5.6M	14.8M	0.5M	6.2M
$\|E\|$	37.6K	451.8K	1.4M	635	19K

Flickr. Second, AOL shares data with a number of parties that might exchange (SCR). We extract dataset **Mixed** from AOL, and set E_r = {Google, Yahoo, MySpace, eBay, Amazon, Wikipedia}. Fig. 6 presents a closer look at the distribution of time points, the number of clicks, and the number of URLs for this dataset.

Oldenburg - Contains synthetic data generated using Brinkhoff's traffic data generator [9] 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 timestamp 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. Fig. 7 presents a closer look at the distribution of time points, the number of locations, and the number of trajectories for this dataset.

Worldcup98 - Contains the access log to World Cup 1998 web site[5] with 2,140,622 click streams. We considered sharing click streams with a party that collects timestamps 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. Fig. 8 presents a closer look at the distribution of time points, the number of clicks, and the number of URLs for this dataset.

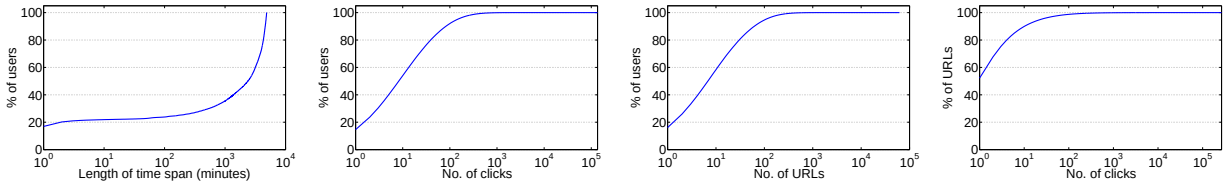
6.2 Settings of experiments

We implemented algorithms in C. Experiments were conducted on a machine with a dual core 3Ghz CPU, 2GB RAM, running Linux Fedora 12. The index for clusters was implemented using R-Tree[6] 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. All fingerprints were stored in main memory. In SSR experiments, we set $w_e = 0$ and in SCR experiments, we set $w_e = w_t = 1$.

6.3 Building a taxonomy for URLs

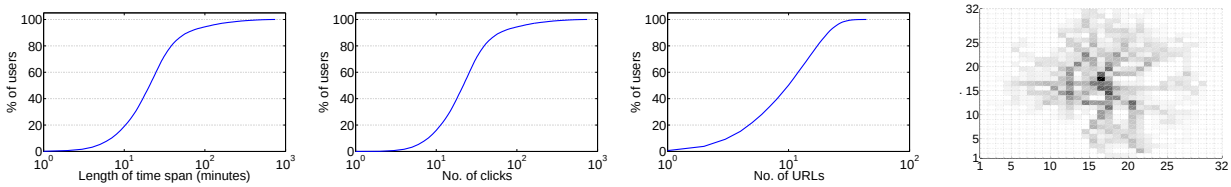
We extracted the category of URLs in AOL dataset by sending queries to Alexa[7] in October 2009. One of three cases happens for each URL: (1) Alexa has the category of the URL, (2) Alexa returns a list of *similar* URLs, and (3) Alexa has no entry. In case 1, e.g. for Facebook, Alexa returned `Computers/Internet/On the Web/Online Communities/Social Networking` as the longest category. We only consider the *first four categories* from the longest category returned by Alexa for the URL. We organize URLs, based on four categories, into a taxonomy. In case 2, a URL is assigned to the category which is shared among its similar URLs. In case 3, the URL is assigned to the Miscellaneous category, which includes

[5] http://ita.ee.lbl.gov/html/contrib/WorldCup.html
[6] http://www2.research.att.com/~marioh/spatialindex/
[7] http://www.alexa.com



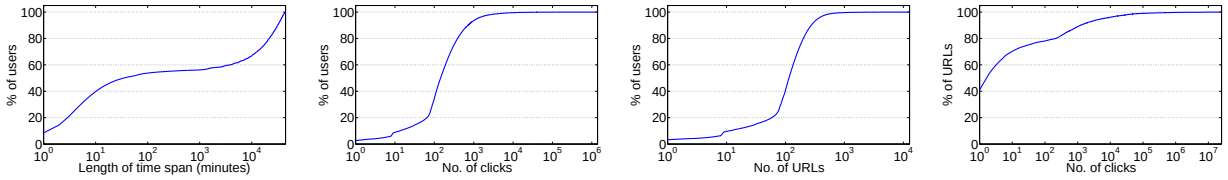
(a) (b) (c) (d)

Figure 6: AOL dataset; frequency distribution of time span of click-streams vs. number of users (a), length of click-streams vs. number of users (b), number of distinct URLs vs. number of users (c), and number of clicks vs. number of URLs (d); all x-axis are in log-scale.



(a) (b) (c) (d)

Figure 7: Oldenburg dataset; frequency distribution of time span vs. number of trajectories (a), locations vs. number of trajectories (b), number of distinct locations vs. number of trajectories (c), and density of locations in grid by the number of trajectory visits, **darkness** proportional with number of visits (d), all axis in log-scale except (d).



(a) (b) (c) (d)

Figure 8: Woldcup98 dataset; frequency distribution of time span of click-streams vs. number of users (a), length of click-streams vs. number of users (b), number of distinct URLs vs. number of users (c), and number of clicks vs. number of URLs (d); all x-axis are in log-scale.

Table 4: Suppression, f_p and f_n, changing k on Oldenburg

	TA [27]			HN (Alg.1)	SA [37]
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

typos and un-popular URLs. By this method, we could build a taxonomy of height four with $45,383$ nodes, maximum fan-out of 128 (avg. 3), and 884K categorized URLs (case 1 or 2) from 1.6M URLs.

6.4 Results

6.4.1 Comparison with related work

Except TA [27] and SA [37], our approach is not comparable to previous works (e.g. [2, 31]) as *they ignore time points both in forming groups and anonymization*. We stress that the anonymized data produced by HN conforms with the symmetric requirement of SA; each AG of HN is a complete bipartite graph, thus the induced attack graph [37] is symmetric. To make a comparison independent of the underlying anonymization methods, we measured false positive (f_p) and false negative (f_n) ratios on k-anonymized data (i.e. same privacy level) produced by HN, TA, and SA. The ratios reflect the accuracy of *count* queries (e.g. used in complex spatio-temporal pattern queries [17]). We picked Oldenburg because TA performs geometric transformations on spatial coordinates. We divided the time, x, and y coordinates into intervals of equal length. A combination of temporal and spatial dimensions provided the query workload Q. For dataset $\mathcal{D}$ and anonymized data $\mathcal{D}^*$, let $nHits(q, \mathcal{D})$ and $nHits(q, \mathcal{D}^*)$, respectively, be the number of unique trajectories in $\mathcal{D}$ and $\mathcal{D}^*$ that pass the spatial range of query $q \in Q$, during q's time range. The ratios f_p and f_n are defined as:

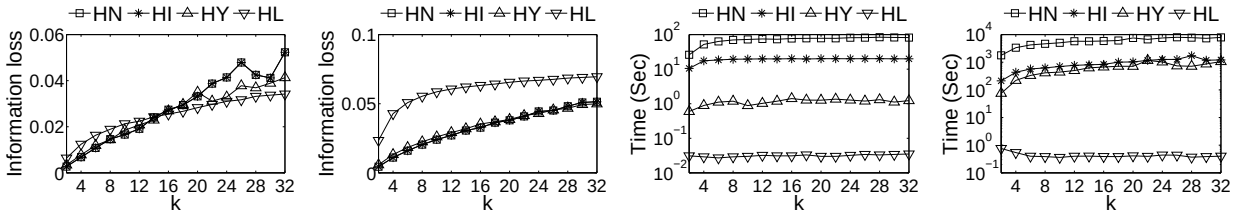
$$f_p(\mathcal{D}^*) = \frac{\left|\{q \,|\, q \in Q,\, nHits(q, \mathcal{D}^*) > nHits(q, \mathcal{D})\}\right|}{|Q|},$$

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

Both f_p and f_n are in range $[0, 1]$ and a smaller value is favorable for both ratios. We observed that $f_n = 0$ for both HN and SA. In Table 4, 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. The AG can get biased towards a single sequence. However, in HN each AG is found iteratively: an AG is initialized to a single sequence and in an iterative process, it is merged with the sequence closest to the current AG. (2) The symmetric merging of AGs in SA generates larger AGs in an attempt to maintain the attack graph symmetric. This results into more data distortion in form of larger f_p. HN is the best approach when false negatives are not acceptable.

6.4.2 Evaluation of SSR setting

In k-anonymity (Fig. 9), we observe that the information loss is very small for all methods (less than 6%). The running time increases monotonically with k as more comparisons and disk accesses are required to form larger AGs. In Flickr, HL is very close to HN in terms of information loss (i.e. NCP) 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



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

Figure 9: k-anonymity changing k

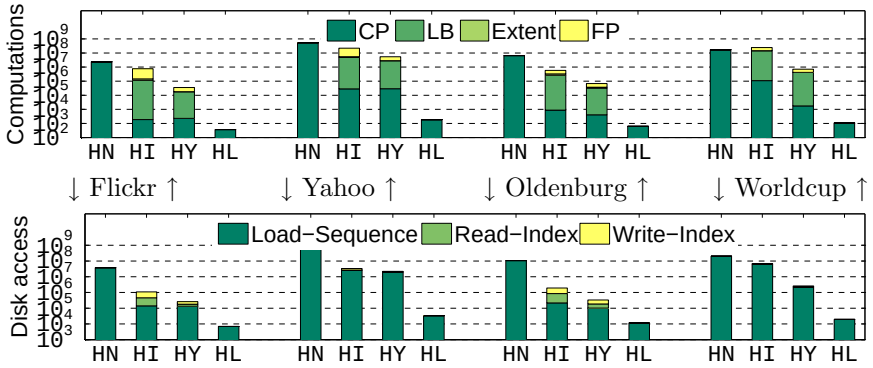
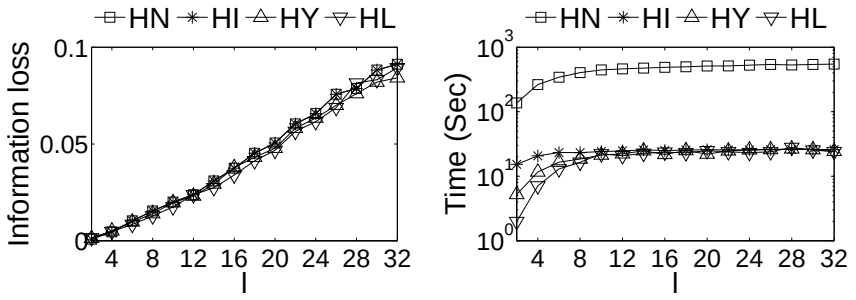


Figure 10: Computations and disk accesses, k-anonymity ($k = 16$)

from partitioning and runs ten times faster than HI without a significant drop in information loss (NCP). For 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 Flickr and Yahoo, HI runs up to an order of magnitude faster than HN as the index and FastNN reduce the number of CP computations by almost four orders of magnitude and the number of disk accesses by more than one order of magnitude (Fig. 10), which explains the observed improvement of running time for HY (Fig. 9). HI and HY have two overheads: creating index (I/O) and evaluating lower bounds (CPU). However, the lower bounds are very efficient to compute and both HI and HY benefit from the index. HL 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.

In (g, ℓ)-diversity, again we observe that the information loss is very small for all datasets. 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 Oldenburg in which we see a *unification* of HI, HL, and HY (Fig. 11). 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 (see Fig. 7d). These sequences are candidates for (g, ℓ)-diversity violations. In HL and HY, these sequences are handled by HI. Fig. 12 confirms this pattern; HL and HY make the same number of CP computations as HI.



(a) NCP, Oldenburg (b) Time, Oldenburg

Figure 11: (g, ℓ)-diversity for Oldenburg, changing ℓ ($g = 1$ hour)

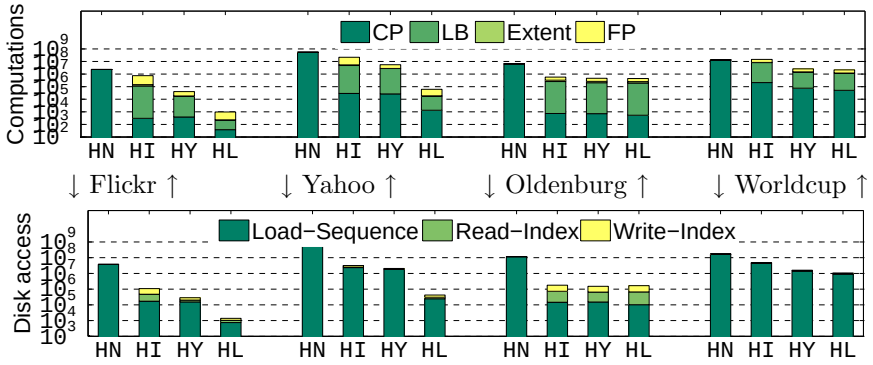


Figure 12: Computations and disk accesses, (g, ℓ)-diversity, $\ell = 16$, $g = 1$ hour

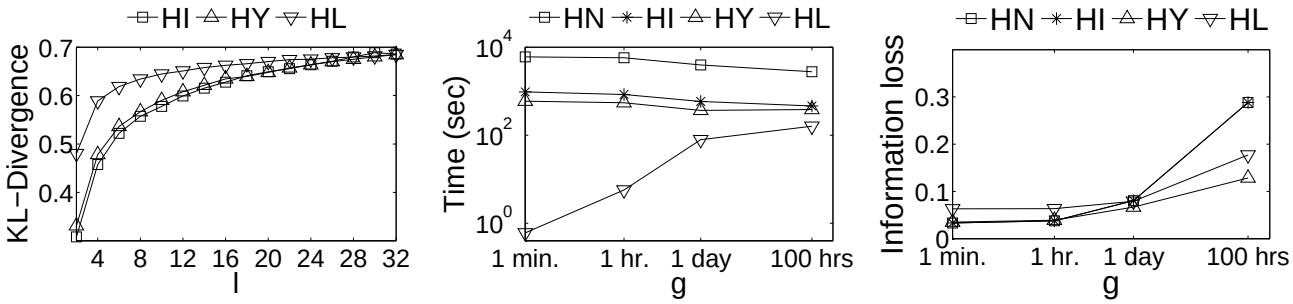
6.4.3 The utility of anonymized data

To measure the usefulness of the anonymized data, we considered temporal prediction queries: *A receiver that collects timestamps of URL A wants to find the distribution of visits for click streams within* $\boldsymbol{w}$ *time units of visits to URL A*, (e.g. Google may want to know the probability that one visits Amazon after 5 minutes of visiting Google, or conversely, the probability that one visits Bing 5 minutes before visiting Google). Let P_o and P_a be the probability distribution of events in the original and the anonymized data, respectively. To measure the closeness of the two distributions, we use the Kullback-Leibler Divergence [22] as it has been acknowledged as a representative metric in the data anonymization literature [20]:

$$D_{KL}(P_a, P_o) = \sum_{event} P_o(event) \log \frac{P_o(event)}{P_a(event)}.$$

D_{KL} is zero for identical distributions and a smaller value denotes a more distribution preserving anonymization. We observed that D_{KL} increases with ℓ (Fig. 13a). 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 in line with the trend observed for information loss (not reported). Again, HL has the largest D_{KL} among other methods (D_{KL} is the same for HN and HI).

Increasing g adds more grouping constraints. This has two impacts in Fig. 13b: (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



(a) D_{KL} (b) Time (c) NCP

Figure 13: (g, ℓ)-diversity in Yahoo changing ℓ while $g = 1$ hour (a) and changing g while $\ell = 16$ (b) and (c)

Table 5: KL-Divergence changing $\boldsymbol{w}$, $g = 1$ hour, $\ell = 16$

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

time of HL and HY converge to HI. In terms of information loss (Fig. 13c), it seems hat greedily checking constraints locally (as in HN and HI) does not always yield the best AG.

The distribution of P_a gets closer to P_o when $\boldsymbol{w}$ takes larger values (Table 5). This indicates that while the anonymized data fulfills the desired privacy model, the anonymized data is more useful for predicting long term trends.

6.4.4 Evaluation of SCR setting

We compared HI with ET on a sample of AOL dataset with 5K sequences, where $E_r = E$[8]. The avg. length of sequences is 72. Because ET intentionally provides more privacy coverage, it incurs a larger information loss, as observed in Table. 6. Besides, it takes longer to anonymize data using ET. Because ET may generalize *every* time point of sequences, we observe a relatively larger information loss vs. HI, compare information loss with Flickr/Yahoo in Fig. 9. 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 the progressive clustering, its running time is higher than HI due to calling *eventDist* for partitioning.

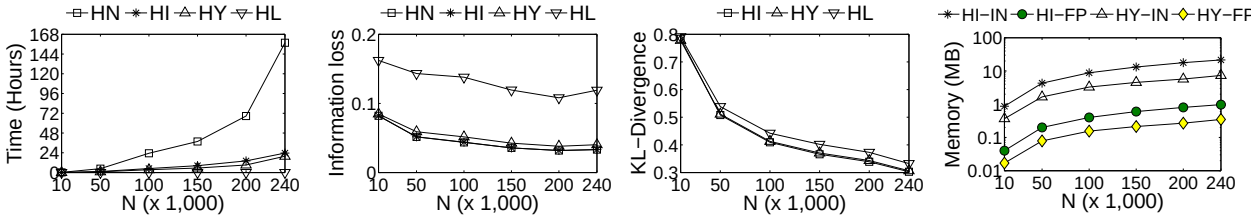
6.4.5 Scalability evaluation

Fig. 14 summarizes scalability evaluations for k-anonymity; we got very similar results for (g, ℓ)-diversity. All our algorithms scale better than HN (Fig. 14a), which has a quadratic scalability. Except for HL, the information loss is relatively small (under 10%) for all methods (Fig. 14b). 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} (Fig. 14c). 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 (Fig. 14b).

[8]the set of all URLs

Table 6: *time* vs. event & time generalization ($w_t = w_e = 1$)

	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	121.1	263.8	398.5	565.9	0.81	0.89	0.92	0.94



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

Figure 14: Scalability of k-anonymity in Mixed ($k = 16$)

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 Fig. 14b.

We next measure the overhead of HI and HY (Fig. 14d); HI and HY require an index for clusters and memory to store fingerprints. In Fig. 14d, 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 cluster 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 indicates that HI and HY are practical for large datasets with a relatively small resource overhead.

6.5 Summary of experimental evaluations

- Under the same privacy model, our algorithm is superior to existing techniques, in terms of an independent measure, when false negatives are not acceptable.
- The information loss is very small in SSR setting. However, due to the increased privacy level of SCR, the information loss can be very large.
- In both k-anonymity and (g, ℓ)-diversity, our proposed improvements (HI, HL, and HY) produce anonymizations with the same quality as the baseline method (HN). HL is very fast but incurs a larger information loss for long sequences (e.g. in Yahoo). When sequence length is small (e.g. Flickr), we did not observe a large drop in quality for HL. HI provides the same information loss as HN but much faster (an order of magnitudes).
- Data distribution may degenerate HL and HY into HI (e.g. in (g, ℓ)-diversity for Oldenburg). The existence of a large number of constraints may render our heuristics, except for HI, ineffective.
- Our methods scale well with input size N, HI is expected to be $O(N \log N)$ in average, and the resource overhead (main memory and disk) of our methods are relatively small.

7 Conclusions and Future Work

We extended traditional privacy models to event sequences and introduced a novel anonymization based on time and event generalization. We proposed index-based algorithms, compact summaries, and effective lower bounds to speed up anonymization. On the way, we also proposed a natural distance function for multisets which uses domain semantics modeled by a taxonomy. Our experiments show the efficiency of our algorithms, the quality of anonymizations, and the scalability of our methods.

In the future, we will focus on the problem of event inference attacks in a setting where an association between *time* and *sensitive* events can be learned from the published data. Evaluating the utility of the anonymized data in data mining domain is another interesting direction for future work.

References

[1] The octopus company. http://www.octopus.com.hk.

[2] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, 2008.

[3] C. C. Aggarwal and P. S. Yu. On anonymization of string data. In *SDM*, 2007.

[4] C. C. Aggarwal and P. S. Yu. On privacy-preservation of text and sparse binary data with sketches. In *SDM*, 2007.

[5] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR*, 2006.

[6] R. Agrawal and R. Srikant. Privacy-preserving data mining. *SIGMOD Records*, 29(2), 2000.

[7] N. Alon, , Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, 1996.

[8] M. Barbaro and T. Zeller. A face is exposed for aol searcher no. 4417749. *The New York Times*, 2006.

[9] T. Brinkhoff. Generating traffic data. *IEEE Data Engineering Bulletin*, 26(2), 2003.

[10] M. Deshpande and G. Karypis. Selective markov models for predicting web page accesses. *ACM TOIT*, 4(2), 2004.

[11] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR*, 2008.

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

[13] B. Fung, M. M. Cao, B. Desai, and H. Xu. Privacy protection for RFID data. In *SAC*, 2009.

[14] G. Ghinita. Private queries and trajectory anonymization: a dual perspective on location privacy. *Transactions on Data Privacy*, 2(1), 2009.

[15] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. Fast data anonymization with low information loss. In *VLDB*, 2007.

[16] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD*, 1984.

[17] M. Hadjieleftheriou, G. Kollios, P. Bakalov, and V. J. Tsotras. Complex spatio-temporal pattern queries. In *VLDB*, 2005.

[18] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *PVLDB*, 2(1), 2009.

[19] D. Kifer. Attacks on privacy and definetti's theorem. In *SIGMOD*, 2009.

[20] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *SIGMOD*, 2006.

[21] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of ACM*, 53(4), 2010.

[22] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1), 1951.

[23] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, 2007.

[24] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[25] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *PODS*, 2004.

[26] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE TKDE*, 13(1), 2001.

[27] M. E. Nergiz, M. Atzori, Y. Saygin, and B. Güç. Towards trajectory anonymization: a generalization-based approach. *Transactions on Data Privacy*, 2(1), 2009.

[28] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *PODS*, 1998.

[29] T. Seidl and H.-P. Kriegel. Optimal multi-step k-nearest neighbor search. In *SIGMOD*, 1998.

[30] L. Sweeney. k-anonymity: a model for protecting privacy. *Intl. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5), 2002.

[31] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *MDM*, 2008.

[32] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *PVLDB*, 1(1), 2008.

[33] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *ICML*, 2000.

[34] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Computational Biology*, 1(4), 1994.

[35] X. Xiao and Y. Tao. Anatomy: simple and effective privacy preservation. In *VLDB*, 2006.

[36] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *KDD*, 2006.

[37] R. Yarovoy, F. Bonchi, L. V. S. Lakshmanan, and W. H. Wang. Anonymizing moving objects: how to hide a mob in a crowd? In *EDBT*, 2009.

A Summaries and lower bounds

Because we consider the SSR setting, in this section by *sequence* we mean an ordered set of timestamps, (e.g. those in Fig. 1-right).

Definition 5 (Cluster fingerprint) *A fingerprint maps the timestamps of each cluster into a bitmap of length b.*

We divide the general time span T into b buckets of length $|b| = (\max(T) - \min(T))/b$ each. Let B_i denote the fingerprint corresponding to cluster C_i and let $B_i[r]$, $1 \leq r \leq b$, be the bit at index r of B_i. We set $B_i[r]$ to one if there exists at least one timestamp t in any sequence of C_i such that $r \cdot |b| > t \geq (r-1) \cdot |b|$, otherwise $B_i[r] = 0$ is set to zero. At least one bit of B_i must be set to one because C_i is non-empty. The minimum generalization of each timestamp in a cluster can be estimated from the relative position of the bits that are set to one in fingerprints. For fingerprint B and location r, $1 \leq r \leq b$,

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

Intuitively, $mg(B, r)$ 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 the number of 0-bits is zero. The minimum information loss of a timestamp which is mapped to location r can be estimated from $mg(B, r)$ and the bucket length.

Lemma 4 *Let B_1 and B_2 be the fingerprints for clusters C_1 and C_2, respectively. Let $\overline{B[r]}$, $1 \leq r \leq b$, be one when $B[r]$ is zero and zero otherwise. $D_{fp}(B_1, B_2)$ defined as*

$$\frac{\sum\limits_{\substack{1 \leq r \leq b \\ B_1[r] \bigwedge \overline{B_2[r]}}} mg(B_2, r) + \sum\limits_{\substack{1 \leq r \leq b \\ \overline{B_1[r]} \bigwedge B_2[r]}} mg(B_1, r)}{b \cdot \sum_{S \in C_1 \cup C_2} |S|} \tag{5}$$

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

Proof 4 *Let timestamp s_1 in cluster C_1 be mapped to bucket r_1 in B_1 and s_2 be the closest timestamp to s_1 in cluster C_2. Let s_2 be mapped to bucket r_2 in B_2. Because there are $mg(B_2, r_1)$ zero bits in B_2 between s_1 and the bucket of its closest timestamp, the minimum information loss when s_1 is generalized to interval $[s_1\text{–}s_2]$ is at least $|b| \cdot mg(B_2, r_1)$. To find the minimum information loss for generalizing s_2, we need to find the closest timestamp to s_2 as it can be different from s_1 but the same argument applies. If both $B_1[r]$ and $B_2[r]$ are set to one, the smallest generalization of s_1 and s_2 happens when $s_1 = s_2$, making $minLoss(s_1, C_1 \cup C_2) = minLoss(s_2, C_1 \cup C_2) = 0$. But when $B_1[r] \neq B_2[r]$, the minimum information loss can be greater than zero. Therefore,*

$$D_{fp}(B_1, B_2) = \frac{\sum\limits_{\substack{1 \leq r \leq b \\ B_1[r] \bigwedge \overline{B_2[r]}}} mg(B_2, r) + \sum\limits_{\substack{1 \leq r \leq b \\ \overline{B_1[r]} \bigwedge B_2[r]}} mg(B_1, r)}{b \cdot \sum\limits_{S \in \{C_1 \cup C_2\}} |S|} \leq$$

$$\frac{\sum\limits_{S \in C_1 \cup C_2} \sum\limits_{t \in S} minLoss(t, C)}{\underbrace{b \cdot |b|}_{|\max_T - min_T|} \cdot \sum\limits_{S \in \{C_1 \cup C_2\}} |S|} = D_{LB}(C_1 \cup C_2).$$

where T is the global time interval. To establish the proof, later in Eq. 9 we define minLoss and prove in Lemma 7 that D_{LB} provides a lower bound of CP.

Definition 6 (Cluster Extent) *The cluster extent E_C is a compressed representation for a set of clusters $C = \{C_k\}_{k=1}^{|C|}$. E_C has the following features:*

$s_1(E_C) = \min_{C_k \in C} start(C_k)$	$s_2(E_C) = \max_{C_k \in C} start(C_k)$
$e_1(E_C) = \min_{C_k \in C} end(C_k)$	$e_2(E_C) = \max_{C_k \in C} end(C_k)$
$c_1(E_C) = \min_{C_k \in C}\{\sum_{S \in C_k} \lvert S\rvert\}$	$c_2(E_C) = \max_{C_k \in C}\{\sum_{S \in C_k} \lvert S\rvert\}$
$l_1(E_C) = \min_{C_k \in C} \lvert\{t \mid t \in S \in C_k,\, s_2(E_C) < t < e_1(E_C)\}\rvert$	
$l_2(E_C) = \max_{C_k \in C} \lvert\{t \mid t \in S \in C_k,\, s_2(E_C) < t < e_1(E_C)\}\rvert$	

where $start(C_k) = \min_{S \in C_k} start(S)$, $end(C_k) = \max_{S \in C_k} end(S)$.

Intuitively, a cluster extent represents a set of clusters as a minimum bounding rectangle (MBR) with four features: start time, end time, total number of timestamps, and the number of timestamps between the latest first timestamp and the earliest last timestamp. A cluster extent E_C can derive a lower bound of $CP(C_i \cup C_k)$ for C_i and any $C_k \in C$ depending on the overlap of C_i and E_C; depending on $l_1(E_C)$ and $l_2(E_C)$.

1. No overlap if $l_2(E_C)$ is zero - Finding the minimum generalization of each timestamp in C_i and any cluster covered by E_C is straightforward. If $end(C_i) \leq s_1(E_C)$, since each timestamp in C_i must be included in one interval that covers a timestamp of a sequence in cluster $C_k \in C$, the information loss of each timestamp in C_i is at least $s_1(E_C) - start(C_i)$, if all timestamps of C_k are at $start(C_k)$. But the last timestamp of C_k is not before $e_1(E_C)$. Thus, the smallest interval which covers all time points of C_i and C_k is the range $[start(C_i)$–$e_1(E_C)]$. A similar argument applies if $start(C_i) \geq e_2(E_C)$.

Lemma 5 *For cluster C_i and cluster extent E_C corresponding to $C = \{C_k\}_{k=1}^{|C|}$, if $end(C_i) \leq s_1(E_C)$ then*

$$D_{Extent}(C_i, E_C) = \frac{e_1(E_C) - start(C_i)}{\max_{S \in \mathcal{D}} end(S) - \min_{S \in \mathcal{D}} start(S)}$$

provides a lower bound for $\min_{C_k \in C} CP(C_i \cup C_k)$. Likewise, if $start(C_i) \geq e_2(E_C)$ then $D_{Extent}(C_i, E_C)$ is defined as:

$$D_{Extent}(C_i, E_C) = \frac{end(C) \; - s_2(E_C)}{\max_{S \in \mathcal{D}} end(S) - \min_{S \in \mathcal{D}} start(S)}.$$

Proof 5 *Consider the case $start(C_i) > e_2(E_C)$ first; the case where $end(C_i) < s_1(E_C)$ is very similar. Because the intervals in $\mathcal{I}$ must cover every timestamp of both C_i and $C_k \in C$, only one interval is possible else the intervals would overlap. The length of this single interval is at least $end(C_i) - s_2(E_C)$. We must show that for any cluster $C_k \in C$, $D_{Extent}(C_i, E_C)$ is no more than $CP(C_i \cup C_k)$; this is because for cluster C_k the optimum interval is $[start(C_k)$–$end(C_i)]$ and its information loss is at least $end(C_i) - s_2(E_C)$, by definition of $s_2(E_C)$.*

2. Possibly overlap if $l_1(E_C) > 0$ - For a timestamp t and sequence $S \in C_k \in C$, let $nearest(t, S)$ be a timestamp in which is closest to t. For a timestamp t in a sequence of cluster C_i, if t is in $range_1 = [s_1(E_C), s_2(E_C)]$, then $nearest(t, S)$ could be the same as t. Thus, the minimum information loss for any timestamp in $range_1$ could be zero. The same argument applies to $range_2 = [e_1(E_C), e_2(E_C)]$. The only ranges with possibly non-zero loss are outside the range $[s_1(E_C), e_2(E_C)]$, or inside $range_3 = [s_2(E_C), e_1(E_C)]$, provided that there is no timestamp in $range_3$ for *any* $C_k \in C$.

Lemma 6 *For cluster extent E_C corresponding to $C = \{C_k\}_{k=1}^{|C|}$ let the range $M_{E_C} = \big(s_2(E_C), e_1(E_C)\big)$. For cluster C_i, let $C_i \cap M_{E_C}$ be the set of timestamps of cluster C_i which are inside the range M_{E_C}. If the timespan of cluster C_i overlaps with extent E_C, then $D_{Extent}(C_i, E_C)$ defined as*

$$\frac{D_1 + D_2 + D_3}{\Big(\max_{S\in\mathcal{D}} end(S) - \min_{S\in\mathcal{D}} start(S)\Big)\Big(c_2(E_C) + \sum_{S\in C_i} |S|\Big)}$$

provides a lower bound for $\min_{C_k\in C} CP(C_i \cup C_k)$, where

$$\begin{aligned} D_1 &= |\{t \mid t \in S \in C_i, t < s_1(E_C)\}| \cdot \big(s_1(E_C) - start(C_i)\big), \\ D_2 &= |\{t \mid t \in S \in C_i, t > e_2(E_C)\}| \cdot \big(end(C_i) - e_2(E_C)\big) \end{aligned}$$

are the information loss of the timestamps of cluster C_i before $s_1(E_C)$ and after $end(E_C)$, respectively, and D_3 as

$$D_3 = \begin{cases} 0 & \exists\ C_k \in C,\ C_k \bigcap M_{E_C} \neq \emptyset \\ \min\big(D_3^1, D_3^2\big) & otherwise \end{cases}$$

provides a lower bound to the information loss of the timestamps of cluster C_i in range M_{E_C}, where D_3^1 and D_3^2 are:

$$\begin{aligned} D_3^1 &= \big|\ C_i \cap M_{E_C}\ \big| \cdot (\max_{t\in C_i\cap M_{E_C}} t - s_2(E_C)) \\ D_3^2 &= \big|\ C_i \cap M_{E_C}\ \big| \cdot (e_1(E_C) - \min_{t\in C_i\cap M_{E_C}} t). \end{aligned}$$

Proof 6 *Because $c_2(E_C) = \max_{C_k\in C}\{\sum_{S\in C_k} |S|\}$,*

$$\sum_{S\in C_i\cup C} |S| \leq \Big(\sum_{S\in C_i} |S| + c_2(E_C)\Big). \tag{6}$$

There are three possible generalization for timestamp $t \in C_i$. i) When $t < s_1(E_C)$ every timestamp of C_i in this range is covered by a single interval, otherwise the intervals would overlap. The shortest interval which also covers one timestamp of a cluster in C is $I_1 = [start(C_i), s_1(E_C)]$. Similarly, when $t > e_2(E_C)$, the shortest interval which covers t and one timestamp of any cluster in C is $I_2 = [e_2(E_C), end(C_i)]$. ii) When C_i overlaps with either the start range or the end range of an extent; either $[s_1(E_C), s_2(E_C)]$ or $[e_1(E_C), e_2(E_C)]$. The nearest timestamp to t in any cluster of C could be the same timestamp as t in the extreme case, resulting into an interval of length zero and consequently a zero information loss. iii) When t is in range $M_{E_C} = (s_2(E_C), e_1(E_C))$ and there is no timestamp in any cluster of E_C in this range, an interval which contains all timestamps of C_i in this range and at least *one timestamp from a cluster in C must pick the candidate timestamp from $s_2(E_C)$ or $e_1(E_C)$. The interval will be $I_3^1 = [s_2(E_C), \max_{t\in C\cap M_{E_C}} t]$ in the former case and $I_3^2 = [min_{t\in C\cap M_{E_C}} t, e_1(E_C)]$ in the latter case. By definition*

$$CP(C_i \cup C_k, \mathcal{I}) = \frac{\sum_{S\in C_i\cup C_k} \sum_{I\in\mathcal{I}} IDD(S, I)}{(\max_T - \min_T) \cdot \sum_{S\in C_i\cup C_k} |S|} \tag{7}$$

where T is the global time interval. Replacing $I(t)$ in Eq.7 with either of I_1, I_2, or I_3^1 which are derived by assuming a greedy nearest neighbor approach to form intervals, and assuming that I_3^1 has a smaller

information loss than I_3^2 and that C has no timestamp in range M_{E_C} yields:

$$\begin{aligned} \sum_{S \in C_i \cup C_k} \sum_{I \in \mathcal{I}} IDD(S, I) \quad &\geq \\ |C_i \cap I_1| \cdot \Big(s_1(E_C) - start(C_i) \Big) \quad &+ \quad // \, D_1 \\ |C_i \cap I_2| \cdot \Big(end(C_i) - e_e(E_C) \Big) \quad &+ \quad // \, D_2 \\ |C_i \cap I_3^1| \cdot \Big(\max_{t \in C_i \cap M_{E_C}} t - s_2(E_C) \Big) \quad & \quad // \, D_3^1. \end{aligned} \tag{8}$$

replacing the nominator of Eq.7 with R.H.S of Eq.8, which is a smaller value, and replacing $\sum_{S \in C_i \cup C_j} |S|$ of Eq.7 with R.H.S of Eq.6, which is a larger value, establishes the proof if there is no cluster $C_k \in C$ that overlaps with M_{E_C}, i.e. $l_2(E_C)$ is zero. Otherwise, there are timestamps of C_k in M_{E_C} and the minimum information loss happens when all time points in $C_i \cup M_{E_C}$) are generalized using an interval of zero length.

Cluster-level lower bound - For two clusters C_i and C_j, let $C_{i,j} = C_i \cup C_j$ for short. If $C_{i,j}$ has only two sequences S and R, the minimum generalization of timestamp t in sequence S is at least $|t - nearest(t, R)|$, where $nearest(t, R)$ is the closest timestamp of sequence R to t. The idea can be generalized when $|C_{i,j}| > 2$ to compute a lower bound for the minimum information loss of timestamp t as follows:

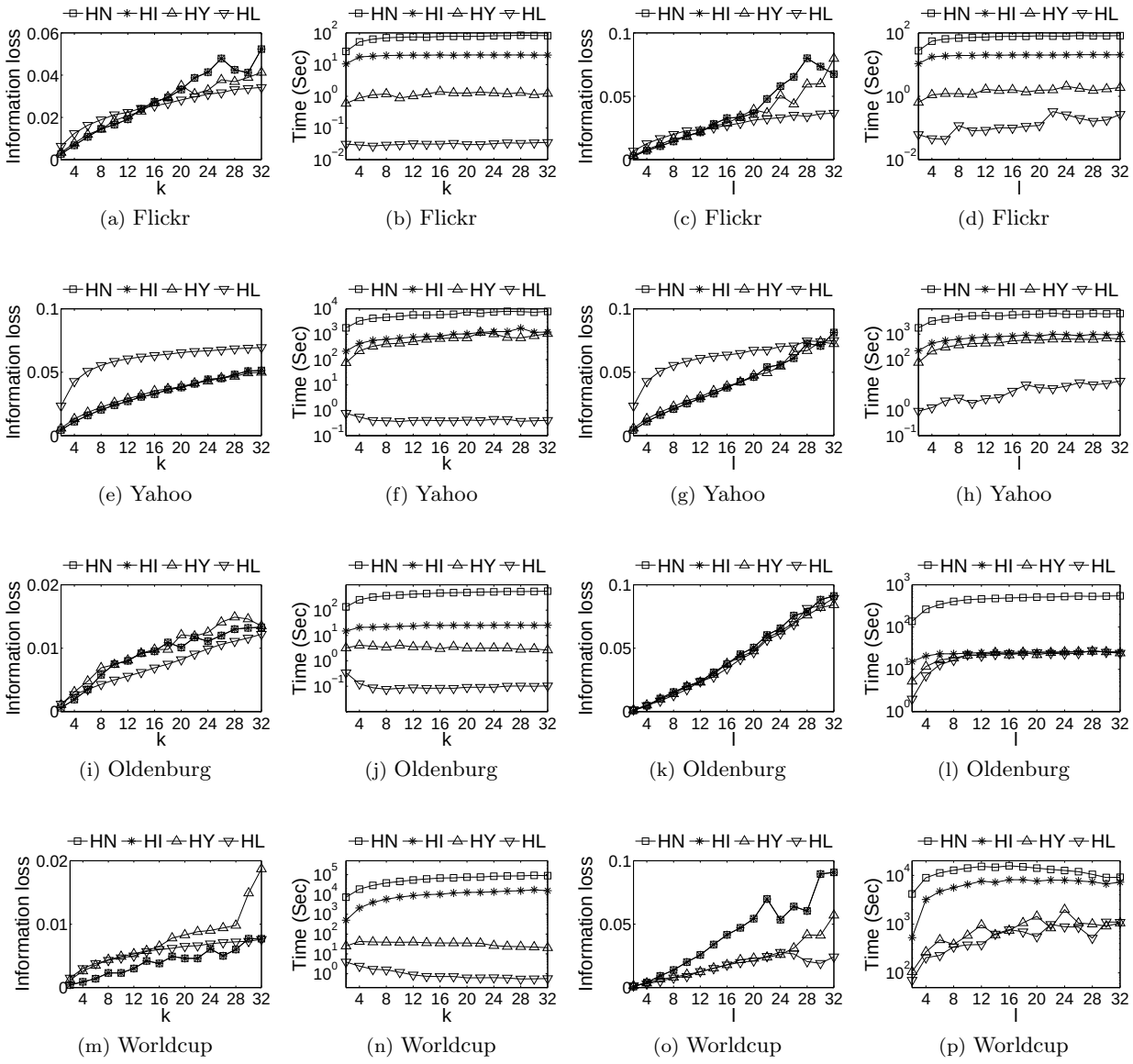
$$minLoss(t, S, C_{i,j}) = \max_{R \, \in \, C_{i,j}/S} \big| \, t - nearest(t, R) \, \big|. \tag{9}$$

Lemma 7 *For clusters C_i and C_j, $D_{LB}(C_i, C_j)$ defined as:*

$$\frac{\sum_{S \in C_{i,j}} \sum_{t \in S} minLoss(t, \, S, \, C_{i,j})}{\Big(\max_T - \min_T \Big) \cdot \sum_{S \in C_{i,j}} |S|} \tag{10}$$

provides a lower bound for $CP(C_i \cup C_j)$ in $O(n_s \cdot n_t)$ where n_s (n_t) is the number of sequences(timestamps) in cluster $C_{i,j}$, where T is the global time interval.

Proof 7 *For any timestamp t, let $I_t = (c_t, [s_t - e_t])$ be the interval in $\mathcal{I}$ that covers timestamp t. Clearly, $(e_t - s_t) \geq minLoss(s, G)$, and there is no other interval $I_{t'} = (c_{t'}, [s_{t'} - e_{t'}])$ such that $(e_t - s_t) < minLoss(s, G)$. Thus, replacing $minLoss(s, G)$ for $(e_t - s_t)$ in the definition of $IL_t(S, I)$ provides the minimum value of IL_t, and IDD, IL, and consequently CP.*



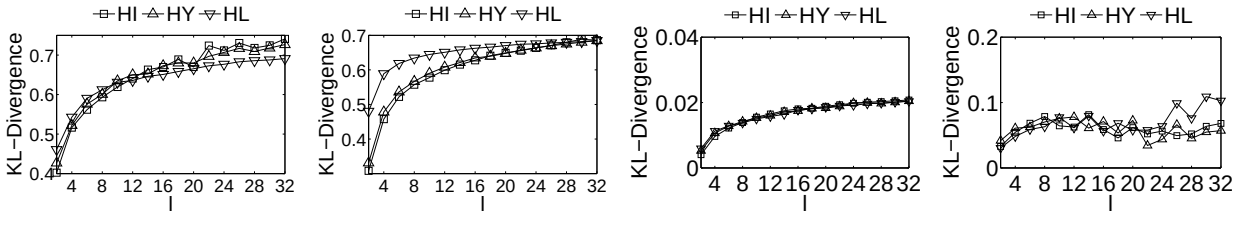
(a) Flickr (b) Flickr (c) Flickr (d) Flickr

(e) Yahoo (f) Yahoo (g) Yahoo (h) Yahoo

(i) Oldenburg (j) Oldenburg (k) Oldenburg (l) Oldenburg

(m) Worldcup (n) Worldcup (o) Worldcup (p) Worldcup

Figure 15: k-anonymity and (g, ℓ)-diversity changing k and ℓ. **Fixed**: $g = 1$ hour in (g, ℓ)-diversity



(a) Flickr (b) Yahoo (c) Oldenburg (d) Worldcup

Figure 16: KL-Divergence for (g, ℓ)-diversity changing ℓ. **Fixed**: $g = w = 1$ hour