# Optimal event sequence sanitization

Grigorios Loukides* Robert Gwadera†

**Abstract**

Frequent event mining is a fundamental task to extract insight from an event sequence (long sequence of events that are associated with time points). However, it may expose sensitive events that leak confidential business knowledge or lead to intrusive inferences about groups of individuals. In this work, we aim to prevent this threat, by deleting occurrences of sensitive events, while preserving the utility of the event sequence. To quantify utility, we propose a model that captures changes, caused by deletion, to the probability distribution of events across the sequence. Based on the model, we define the problem of sanitizing an event sequence as an optimization problem. Solving the problem is important to preserve the output of many mining tasks, including frequent pattern mining and sequence segmentation. However, this is also challenging, due to the exponential number of ways to apply deletion to the sequence. To optimally solve the problem when there is one sensitive event, we develop an efficient algorithm based on dynamic programming. The algorithm also forms the basis of a simple, iterative method that optimally sanitizes an event sequence, when there are multiple sensitive events. Experiments on real and synthetic datasets show the effectiveness and efficiency of our method.

## 1 Introduction

Applications in various domains, including marketing, web analysis, and medicine, feature *event sequences.* Event sequences are usually associated with a large number of time points and have a segmental structure (i.e., they are comprised of segments of differing probability distributions). Such sequences are often produced by monitoring systems, or from aggregating detailed data to a desired time granularity (e.g., days or weeks). A fundamental step to extract insight from an event sequence is mining *frequent* events. These are events whose relative frequency, measured in the entire sequence, is at least equal to a specified threshold [9, 16]. For example, frequent events represent profitable products, popular search queries, or prevailing symptoms in patient populations. In addition, mining frequent events is required to perform other frequent pattern mining tasks (e.g., partial order mining [4]). However, frequent event mining may also result in the exposure of *sensitive* events. These events represent confidential knowledge, such as business secrets, sensitive religious beliefs of groups of individuals, or health-related information of patient subpopulations that may cause unwarranted concerns (e.g., flu-associated hospitalizations). Thus, data owners must ensure that sensitive events are not discovered, when frequent event mining is applied using a specified threshold [24, 23, 19, 3, 11, 14]. As a motivation, consider the following example.

Example 1. *To improve supply and demand planning, a supermarket collaborates with a marketing company. The latter requested last year's weekly sales of products to perform common business intelligence tasks, such as identifying popular products, at various weekly time periods starting from the beginning of the year, and mining temporal trends of product sales. Thus, the supermarket creates an event sequence, by aggregating detailed sales data, and shares it with the marketing company. The sequence is comprised of products* (events) *that are associated with weeks* (time points)*. In each week, products are sold one or more times. However, certain products, which account for a "high" percentage of the total sales, at several time periods starting from the beginning of the year, provide competitive advantage to the supermarket* (sensitive events)*. Thus, the supermarket wants to prevent the mining of these products using a specified threshold, from each prefix of the sequence, so that confidential knowledge will not leak to competitors or used by the marketing company to their advantage. In addition, the utility of the sequence must be preserved, to allow business intelligence tasks to be performed accurately.*

In this work, we aim to prevent the exposure of sensitive events from an event sequence. This can be achieved by deleting occurrences of sensitive events, so that their relative frequency, measured in each prefix of the sequence, is below a specified threshold. Deletion is also employed by existing sanitization methods, which are applied to a collection of transactions (sets of events) [23, 24, 19], sequences (multisets of events that are not associated with time points) [3, 11], or trajectories (lists of spatiotemporal points) [3]. However, arbitrary deletion can significantly reduce the utility of the sequence. This is because deleting occurrences of a sensitive event

*Cardiff University, email: g.loukides@cs.cf.ac.uk
†EPFL, email: robert.gwadera@epfl.ch

$(\{\mathtt{a}.1, \mathtt{b}.4, \mathtt{c}.5, \mathtt{d}.0\}, t_1)$ $(\{\mathtt{a}.2, \mathtt{b}.4, \mathtt{c}.7, \mathtt{d}.7\}, t_2)$ $(\{\mathtt{a}.8, \mathtt{b}.7, \mathtt{c}.1, \mathtt{d}.4\}, t_3)$ $(\{\mathtt{a}.12, \mathtt{b}.13, \mathtt{c}.5, \mathtt{d}.0\}, t_4)$ $(\{\mathtt{a}.7, \mathtt{b}.3, \mathtt{c}.10, \mathtt{d}.0\}, t_5)$

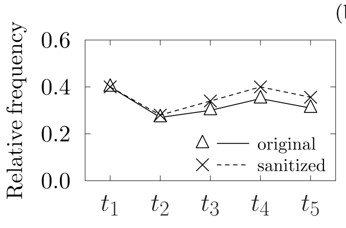
(a)

$(\{\mathtt{a}.0, \mathtt{b}.4, \mathtt{c}.5, \mathtt{d}.0\}, t_1)$ $(\{\mathtt{a}.0, \mathtt{b}.4, \mathtt{c}.7, \mathtt{d}.7\}, t_2)$ $(\{\mathtt{a}.5, \mathtt{b}.7, \mathtt{c}.1, \mathtt{d}.4\}, t_3)$ $(\{\mathtt{a}.7, \mathtt{b}.13, \mathtt{c}.5, \mathtt{d}.0\}, t_4)$ $(\{\mathtt{a}.5, \mathtt{b}.3, \mathtt{c}.10, \mathtt{d}.0\}, t_5)$

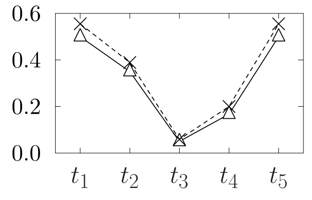
(b)

| Period (weeks) | Rel. freq. of a in Fig. 1(b) |
|---|---|
| $t_1$ | 0 |
| $t_1$ to $t_2$ | 0 |
| $t_1$ to $t_3$ | 0.11 |
| $t_1$ to $t_4$ | 0.17 |
| $t_1$ to $t_5$ | 0.19 |

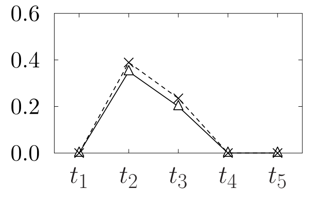
(c) sensitive event a



(d) event b



(e) event c



(f) event d

Figure 1: (a) Event sequence ($.x$ is the number of occurrences of an event). (b) Sanitized event sequence. (c) Relative frequency of a, *measured from week* $t_1$. (d)-(f) Relative frequencies of b, c, and d, *measured at each week*.

that are associated with a time point $t$ affects the relative frequency of *all* events, in *any* part of the sequence that contains $t$. Thus, the output of tasks which require measuring relative frequency in such parts may change. Examples of such tasks are frequent pattern mining, sequence segmentation [13], and temporal trend mining.

Therefore, sanitizing an event sequence while preserving utility is important. It is also challenging, because: (I) the number of ways to delete occurrences of sensitive events is exponential in the number of time points in the sequence, and (II) sanitized event sequences are often used in tasks that are difficult to determine a priori. For instance, one may apply segmentation to a sequence and then frequent event mining to certain prefixes, to discover actionable knowledge [9, 16].

*Our work makes the following contributions:*

First, we propose a model for quantifying the utility of a sanitized event sequence. Our model captures the impact of deletion on the relative frequency of all events, across the entire sequence. This is achieved by measuring the change to the probability of each event, at each time point. Thus, the model penalizes drastic changes in large parts of the sequence, which affect its segmental structure (i.e., significantly change the probability distribution of events along the sequence) and the output of the aforementioned mining tasks. For instance, it avoids the deletion of all occurrences of a sensitive event, from parts in which it is typically frequent (e.g., car antifreeze in winter) and the excessive increase of the relative frequency of typically rare events.

Second, we formally define the optimization problem of *Event Sequence Sanitization* (ESS). The problem requires preventing the mining of sensitive events using a specified threshold, from each prefix of the sequence, while optimizing utility according to our model. This ensures that sensitive events are not exposed when frequent event mining is applied to prefixes, as is typical in practice [9, 16].

Third, we develop ODESA, an optimal algorithm for solving the ESS problem, when there is one sensitive event. We observe that simply deleting the minimum number of occurrences required to sanitize a prefix, from each prefix, may not lead to an optimal solution. This is because deleting occurrences from a prefix in this way may incur large changes to the relative frequency of events in the corresponding overlapping prefixes. Thus, we need to find the exact number of occurrences that must be deleted from each prefix, to optimize the utility of the event sequence. However, directly computing the utility of each way of sanitizing the sequence is prohibitive. This follows from the fact that there are $O(\binom{n+m-1}{m-1})$ ways to delete $n$ occurrences from a sequence containing $m$ time points (each way is a *weak composition* of $n$ into $m$ parts [6]), and $n$ and $m$ are very large. Thus, ODESA applies a dynamic programming equation that finds the best way of deleting $i$ occurrences from a prefix ending at a time point $t$, by deleting $k$ occurrences associated with $t$ and $i-k$ occurrences from the prefix ending at $t-1$, for all possible $k$ values. The equation is applied recursively, to increasingly longer prefixes. This allows ODESA to find an optimal solution in linear time and space (in the number of events and the number of time points of the event sequence, respectively).

Example 2. *Fig. 1(a) shows a part of the event sequence of Example 1, which records the sales of* a *to* d*, over weeks* $t_1$ *to* $t_5$*. Each product is sold at a quantity, denoted after the ".", during a week (products not sold are for illustration only). The supermarket wants to prevent the marketing company from discovering that* a *accounts for at least* 20% *of the total sales, at each time period starting from* $t_1$*. ODESA sanitized the event sequence, as shown in Fig. 1(b). The mining of* a*, from any prefix, has been prevented (see Fig. 1(c)). Furthermore, utility has been preserved, as the relative frequency of nonsensitive events did not change* *significantly at any time point (see Figs. 1(d), 1(e), and 1(f)). Moreover, the output of frequent event mining did not change and temporal trends were preserved.*

Fourth, we develop ESSA, an optimal algorithm for solving the ESS problem, when there are multiple sensitive events. The algorithm works by applying ODESA iteratively and scales linearly with the number of sensitive events in practice. Experiments on real and synthetic datasets demonstrate that ESSA can produce sanitized event sequences that permit accurate mining, and that it has low time and space requirements.

The rest of the paper is organized as follows. Section 2 discusses preliminary concepts. Section 3 presents our utility model and problem formulation. Section 4 presents our sanitization algorithms and Section 5 the experimental evaluation. Section 6 discusses related work and Section 7 concludes the paper.

## 2 Preliminaries

Let $\mathcal{A}$ be a finite set of events and $\mathcal{T}$ a set of time points. Each event $e \in \mathcal{A}$ is associated with a time point $t_i \in \mathcal{T}$. A *Simultaneous Event Multiset* (SEM) is defined as a multiset of events, which occur at the same time point. The set of events in an SEM $M$ is denoted with $\mathcal{E}_M$, and the length (i.e., number of events) of $M$ is denoted with $|M|$. The notation $e.x$ denotes that an event $e$ occurs $x$ times in an SEM (events with zero occurrences are for illustration, and they are not released).

An *event sequence* $D = [(M_1, t_1), \ldots, (M_{|\mathcal{T}|}, t_{|\mathcal{T}|})]$ is an ordered sequence of SEMs, each of which is associated with a different time point. The length of $D$, denoted with $|D|$, is equal to the sum of the lengths of all SEMs in $D$. Given an event sequence $D$ and integers $i$, $j$, such that $i \leq j$, an *event subsequence* $D[i, j] = [(M_i, t_i), \ldots, (M_j, t_j)]$ of $D$ is an event sequence that contains all SEMs, occurring between time points $t_i$ and $t_j$. The length of $D[i, j]$, denoted with $|D[i, j]|$, is equal to the sum of the lengths of all SEMs in $D[i, j]$. The event subsequence $D[1, j]$ is referred to as the *$j$-prefix* of $D$ and denoted with $D_j$. We may omit $j$ from a $j$-prefix, when it is clear from the context.

The *frequency* (number of occurrences) of an event $e$ in an SEM $M$ is denoted with $fr(e, M)$ and its *relative frequency* with $P(e, M)$. The frequency and relative frequency of $e$ in an event sequence $D$ are denoted with $fr(e, D)$ and $P(e, D)$, respectively.

A *sanitized* SEM $M'$ is obtained by deleting *zero or more* event occurrences from an SEM $M$. A *sanitized event sequence* $D' = [(M'_1, t_1), \ldots, (M'_{|\mathcal{T}|}, t_{|\mathcal{T}|})]$ is obtained by deleting *zero or more* event occurrences from each SEM in an event sequence $D = [(M_1, t_1), \ldots, (M_{|\mathcal{T}|}, t_{|\mathcal{T}|})]$. We use deletion, as most sanitization methods [23, 3, 11, 24, 19] do, because it preserves data semantics, unlike noise addition and generalization. That is, the sanitized event sequence is a part of the event sequence, which can be mined without obtaining fake or generalized (aggregate) events. This is important in many applications [15, 25].

Given a threshold $\delta$, an event $e$ is *frequent* in an event sequence $D$, if $P(e, D) \geq \delta$.

## 3 Utility model and problem formulation

This section presents a model for quantifying the impact of deletion on the utility of an event sequence. The model captures changes to the relative frequency of events at each time point, since deletion at a time point $t$ affects the relative frequency of all events, in any part of the sequence that contains $t$. The changes at a time point are captured by measuring the difference between the probability of events, before and after deletion. Subsequently, these differences are summed up across the entire sequence, to capture its utility. In the following, we explain how the model computes the probability of events and the overall utility.

**Assigning probabilities to events.** A utility model should: (I) be applicable to any event sequence, and (II) heavily penalize event sequences from which many event occurrences have been deleted. However, a utility model that computes the probability of an event as its relative frequency, in an SEM $M$ or a sanitized SEM $M'$, does not satisfy these properties. This model is the *maximum likelihood estimate* (MLE) that is not suitable for event sequences, because it assigns zero probabilities to the events that are not contained in $M$ (resp., $M'$) and overestimates the probabilities of the events in $M$ (resp., $M'$) [8]. In particular, the relative frequency of $e$ is *undefined*, when $M'$ is empty, and equal to 1, when $M'$ is produced by deleting *any* number of occurrences from an SEM $M$ that contains a single event.

To overcome these problems, we apply *additive smoothing* with the commonly-used constant 0.5 [8]. Specifically, we assign a probability $\hat{P}(e, M)$ to an event $e$ in an SEM $M$, which is computed as $\frac{fr(e,M)+0.5}{\sum_{\hat{e}\in\mathcal{A}}(fr(\hat{e},M)+0.5)}$, where $\hat{e}$ is an event *in the set of events* $\mathcal{A}$, including $e$. The probability $\hat{P}(e, M')$ of $e$ in a sanitized SEM $M'$ is computed similarly. We will refer to a probability computed using additive smoothing as AS-probability. Note that a non-zero AS-probability is assigned to every event, not just to an event in $M$ or $M'$. Thus, $\hat{P}(e, M')$ can be defined when $M'$ is empty and decreases as more occurrences of $e$ are deleted, when $M'$ contains only $e$. This is because the function $\frac{a-x}{b-x}$, where $a$ and $b$ are the numerator and denominator of $\hat{P}(e, M')$ and $x$ is the number of deleted occurrences of

$e$, is decreasing for any $x \in [1, a)$.

By assigning an AS-probability $\hat{P}(e, M)$ to each event $e$ in $\mathcal{A}$, we associate a probability mass function (*pmf*), denoted with $\hat{P}_M$, with $M$. Similarly, we obtain a pmf $\hat{P}_{M'}$, associated with a sanitized SEM $M'$.

**Comparing probability distributions.** We now measure utility based on the impact of deletion on the probability distribution of events. Specifically, the impact of deletion on an SEM $M$ that is sanitized to $M'$ can be quantified, based on the distance between the *pmf*s $\hat{P}_M$ and $\hat{P}_{M'}$. This distance is denoted with $E(M)$ and computed as $\sum_{e \in \mathcal{A}} (\hat{P}_M(e) - \hat{P}_{M'}(e))^2$. We will also refer to $E(M)$ as the *sanitization error* for $M$.

Based on the sanitization error for $M$, the impact of deletion on an event sequence $D$ that is sanitized to $D'$ is quantified as $E(D) = \sum_{i \in [1, |\mathcal{T}|]} E(M_i)$. $E(D)$ will also be referred to as the *sanitization error* for $D$.

We use Squared Euclidean distance to compute $E(M)$ and $E(D)$, as it is effective for measuring the similarity of event sequences [17], although there are alternatives, such as $KL$-divergence. Clearly, the sanitization error for an event sequence favors the deletion of: (I) a small number of occurrences of sensitive events, and (II) occurrences from a small number of SEMs (i.e., few time points). Consequently, our model can be used to avoid the deletion of all occurrences of a sensitive event and changes to the segmental structure of the sequence, which affect the output of mining tasks. Also, the sanitization error can be computed efficiently, even when $\mathcal{A}$ is large, as the AS-probability for each event that is not contained in the SEM (resp., in the sanitized SEM) is the same.

**Problem formulation.** The *Event Sequence Sanitization* (ESS) problem is defined as follows.

Problem. (Event Sequence Sanitization (ESS)) *Given an event sequence $D$, a threshold $\delta$ and a set of* sensitive *events $\mathcal{S}$ (selected by data owners), construct a sanitized event sequence $D'$ from $D$, such that:* (I) *$P(e, D'_j) < \delta$, for each event $e \in \mathcal{S}$ and each $j$-prefix of $D'$, where $j \in [1, |\mathcal{T}|]$, and* (II) *the sanitization error $E(D)$ is minimum.*

The problem requires constructing a sanitized event sequence $D'$, so that: (I) no sensitive event is frequent, in any $j$-prefix of $D'$, for a specified threshold $\delta$, and (II) $D'$ has optimal utility, according to our model.

We aim to prevent the mining of sensitive events, from any prefix of $D'$ (the longest prefix, $D'_{|\mathcal{T}|}$, is $D'$ itself), because mining is often applied to prefixes [9, 16]. We opt for protecting all prefixes, as it is difficult to know which of them will be mined, in a certain application. Enforcing requirement (I) also prevents the exposure of sensitive events, through mining for sets of events, using a threshold $\delta$. This is because sets that contain sensitive events cannot be mined using a threshold $\delta$, due to the anti-monotonicity property of the relative frequency. Note that we treat sensitive events as not exposed, when their relative frequency is lower than $\delta$. This approach aims to minimize the impact of deletion on data utility and is adopted by most sanitization methods [23, 3, 11, 14, 24, 19]. It makes the reasonable assumption that sensitive knowledge is not actionable, when it is mined at a lower threshold than $\delta$. A different approach that provides stronger protection, but lower data utility [26], can also be adopted by our method (see *Supplementary document* [1]).

ESS is *weakly* NP-hard [20], when $\mathcal{S}$ contains one sensitive event, as shown in Theorem 3.1 (the proof is in the *Supplementary document* [1]). Clearly, the problem remains weakly NP-hard, when $|\mathcal{S}| > 1$.

Theorem 3.1. *The* ESS *problem is weakly NP-hard, when $\mathcal{S}$ contains one sensitive event.*

## 4 Event sanitization method

This section details our method for solving the ESS problem. We first present a dynamic programming equation, used to delete a given number of occurrences from a prefix, in a way that minimizes the sanitization error of the prefix. Next, we present the ODESA algorithm, which uses the equation to optimally solve ESS for a single sensitive event. ODESA forms the basis of an algorithm that solves ESS optimally. This algorithm is called ESSA and is presented subsequently.

### 4.1 Optimal deletion from a prefix using dynamic programming.

Consider a prefix $D_{j+1}$ that must be sanitized by deleting $i$ occurrences of a sensitive event $s$. The prefix can be sanitized optimally, using the following dynamic programming equation:

$$C[i][j] = \min_{\forall k \in [0, fr(s, M_{j+1})],\ k \leq i} (C[i-k][j-1] + E(M^k_{j+1})) \tag{4.1}$$

where $j \geq 0$ and

- $C[i][j]$ is the optimal error of sanitizing $D_{j+1}$
- $C[i-k][j-1]$ is the error incurred by deleting $i-k$ occurrences of $s$ from the prefix $D_j$, and
- $E(M^k_{j+1})$ is the error incurred by deleting $k$ occurrences of $s$ from the last SEM of $D_{j+1}$.

Eq. 4.1 selects the optimal way of deleting $i$ occurrences from $D_{j+1}$, by computing the error of all ways of deleting $i-k$ occurrences from $D_j$ and $k$ occurrences from the last SEM of $D_{j+1}$. This avoids bias towards specific parts of the event sequence (e.g., the first few prefixes where $s$ may be "too" frequent).

**4.2 ODESA: Optimal Dynamic-programming Event SAnitization.** The ODESA algorithm works in three phases: (I) it identifies the minimum number of occurrences of a sensitive event that must be deleted, from each prefix of the event sequence $D$, (II) it computes the optimal sanitization error of $D$, by finding the exact number of occurrences that must be deleted from each prefix, and (III) it performs the sanitization of $D$ with the optimal error. The pseudocode of ODESA describes these phases.

**Phase I (Steps 1-6).** The algorithm identifies all prefixes of $D$ in which the sensitive event $s$ is frequent (Step 1). Then, for each identified prefix, it computes the minimum number of occurrences, whose deletion would make $s$ infrequent in the prefix, and it assigns the maximum of these numbers to $b$ (Steps 3-6). The number corresponding to a prefix $D_j$ is denoted with $b_j$, and it is computed by solving $P(s, D_j) = \frac{fr(s,D_j)-b_j}{|D_j|-b_j} < \delta$, when $fr(s, D_j) < |D_j|$ (otherwise, $b_j = fr(s, D_j)$).

**Phase II (Steps 7-21).** The algorithm computes the optimal sanitization error of the event sequence $D$. Observe that the error will be optimal, if *at least* $b_j$ occurrences are deleted from each prefix $D_j$, in a way that minimizes the sanitization error of the sequence. As $D$ is essentially the prefix $D_{|\mathcal{T}|}$, the optimal error can be computed using Eq. 4.1.

The process begins by creating the array $C$, which will store the results of computing Eq. 4.1, and the array $I$, which will store information about the number of the corresponding deleted occurrences (Steps 7-8). Then, the cells of $C$ that correspond to deleting no occurrences of the sensitive event from a prefix are set to zero (Steps 9-11). Next, ODESA computes the error of deleting $i$ occurrences from the prefix $D_1$, for each $i \in [b_1, fr(s, D_1)]$, and it stores the error values in $C$ and the number of deleted occurrences that correspond to these values in $I$ (Steps 12-15). Following that, ODESA uses Eq. 4.1 to compute the optimal error of deleting $i$ occurrences from $D_{j+1}$, for each $i \in [1, b]$ and each $j \in [1, |\mathcal{T}| - 1]$, and it updates $C$ and $I$ (Steps 16-21). Thus, all ways of deleting at least $b_j$ occurrences from a prefix $D_j$ are considered.

**Phase III (Steps 22-34).** In this phase, ODESA sanitizes $D$ with the optimal error and returns its sanitized counterpart. Before explaining the process, let us examine the information of the tuple in $I[b][|\mathcal{T}| - 1]$. The first element of this tuple contains the number of occurrences of the sensitive event that must be deleted from the last SEM of $D$, as part of the optimal solution, while the second element serves as an index to the cell of $C$ that stores the optimal error for the prefix $D_{|\mathcal{T}|-1}$. Thus, the information in $I$ can be used to construct

```
Algorithm: ODESA
Input: Event sequence $D$, threshold $\delta$, sensitive event $s$
Output: Sanitized event sequence $D'$
   // Phase I: identification of minimum number of
      occurrences that must be deleted
 1 $\mathcal{D} \leftarrow$ set of $j$-prefixes of $D$ such that, for each $D_j$ in the set,
       $P(s, D_j) \geq \delta$
 2 $\mathcal{B} \leftarrow \varnothing$
 3 foreach $D_j \in \mathcal{D}$ do
 4     $b_j \leftarrow$ the minimum integer that is larger than
           $\lceil \frac{fr(s,D_j)-\delta \cdot |D_j|}{1-\delta} \rceil$ and at most $fr(s, D_j)$
 5     $\mathcal{B} \leftarrow \mathcal{B} \cup \{b_j\}$
 6 $b \leftarrow$ the maximum element of $\mathcal{B}$
   // Phase II: optimal sanitization error computation
 7 $C \leftarrow$ a 2D-array with $b + 1$ rows, $|\mathcal{T}|$ columns, and each cell
       initialized to $\infty$
 8 $I \leftarrow$ a 2D-array with $b + 1$ rows, $|\mathcal{T}|$ columns, and each cell
       initialized to $\langle -1, -1 \rangle$
 9 foreach $j = 0$ to $|\mathcal{T}| - 1$ do
10     if $P(s, D_{j+1}) < \delta$ then
11         $C[0][j] \leftarrow 0$
12 foreach $i = 1$ to $b$ do
13     if $i \in [b_1, fr(s, D_1)]$ then
14         $C[i][0] \leftarrow E(M_1^i)$       /* the error of deleting $i$
                                                 occurrences from $M_1$      */
15         $I[i][0] \leftarrow \langle i, -1 \rangle$
16 foreach $j \leftarrow 1$ to $|\mathcal{T}| - 1$ do
17     foreach $i = 1$ to $b$ do
18         $C[i][j] \leftarrow \min_{\forall k \in [0, fr(s, M_{j+1})],\ k \leq i} (C[i-k][j-1]$
                                             $+ E(M_{j+1}^k))$
19         if $C[i][j] \neq \infty$ then
20             $k' \leftarrow \operatorname{argmin}_{\forall k \in [0, fr(s, M_{j+1})],\ k \leq i} (C[i-k][j-1]$
                                             $+ E(M_{j+1}^k))$
21             $I[i][j] \leftarrow \langle k', i - k' \rangle$
   // Phase III: sanitized event sequence construction
22 $D' \leftarrow D$
23 $h \leftarrow |\mathcal{T}| - 1$
24 $\sigma \leftarrow$ the cell $I[b][h]$ of array $I$
25 if $\sigma[0] \neq -1$ then // $\sigma[0]$ denotes the first element of $\sigma$
26     Delete $\sigma[0]$ occurrences of $s$ from the last SEM of the
           prefix $D'_{h+1}$
27 else
28     return $D'$
29 if $\sigma[1] \neq -1$ then // $\sigma[1]$ denotes the second element of $\sigma$
30     $h \leftarrow h - 1$
31     $\sigma \leftarrow$ the cell $I[\sigma[1]][h]$ of array $I$
32     goto Step 25
33 else
34     return $D'$
```

the sanitized event sequence by a backtracking process, which is performed in Steps 23-34.

Specifically, in Steps 23-28, ODESA considers the tuple in $I[b][|\mathcal{T}| - 1]$ and deletes the number of occurrences specified by its first element, from the last SEM of $D'_{|\mathcal{T}|}$, or it returns the sanitized event sequence, if no deletion is required. Thus, the last SEM of $D$ is sanitized, as dictated by the optimal solution. Then, in Steps 29-34, ODESA considers the second element of the tuple in $I[b][|\mathcal{T}| - 1]$ and backtracks to the cell that is specified by this element, or it returns the sanitized event sequence, if no further event deletion is required.

**Complexity.** ODESA needs $O\left(|D| + |\mathcal{T}| \cdot b^2 \cdot |\mathcal{A}|\right)$ time and $O\left(|\mathcal{T}| \cdot (b + |\mathcal{A}|)\right)$ space (see *Supplementary document* [1]). Thus, both its time and space complexity are *pseudopolynomial* [20] in $b$.

| $i \setminus j$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | $\infty$ | ... | ... |
| 1 | $9.2 \cdot 10^{-3}$ | $2.4 \cdot 10^{-3}$ | $\infty$ | ... | ... |
| 2 | $\infty$ | $1.1 \cdot 10^{-2}$ | $3.6 \cdot 10^{-3}$ | ... | ... |
| ... | ... | | | | |
| 13 | ... | | | | $6 \cdot 10^{-2}$ |

(a)

| $i \setminus j$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | $\langle -1, -1 \rangle$ | $\langle -1, -1 \rangle$ | $\langle -1, -1 \rangle$ | ... | ... |
| 1 | $\langle 1, -1 \rangle$ | $\langle 1, 0 \rangle$ | $\langle -1, -1 \rangle$ | ... | ... |
| 2 | $\langle -1, -1 \rangle$ | $\langle 2, 0 \rangle$ | $\langle 1, 1 \rangle$ | ... | ... |
| ... | ... | | | | |
| 13 | ... | | | | $\langle 2, 11 \rangle$ |

(b)

| | | | | |
|---|---|---|---|---|
| $I[1][0] = \langle 1, -1 \rangle$ | $I[3][1] = \langle 2, 1 \rangle$ | $I[6][2] = \langle 3, 3 \rangle$ | $I[11][3] = \langle 5, 6 \rangle$ | $I[13][4] = \langle 2, 11 \rangle$ |

(c)

Figure 2: (a) Array $C$ (the cells contain error values). (b) Array $I$ (the cells contain information about the number of deleted events). (c) Cells of the array $I$ that are considered from right to left, during backtracking.

**Example of applying ODESA.** We apply ODESA to the event sequence $D$ in Fig. 1(a). The sensitive event is a and the threshold $\delta$ is set to 0.2.

Phase I: The algorithm identifies the prefixes $D_3$, $D_4$, and $D_5$, in which a is frequent. Then, it computes the minimum number of occurrences of a that must be deleted from $D_3$ to make the event infrequent in the prefix. This yields $b_3 = 2$. Similarly, ODESA computes $b_4 = 9$, $b_5 = 13$, and $b = max(b_3, b_4, b_5) = 13$.

Phase II: ODESA constructs and initializes the arrays $C$ and $I$, and then sets $C[0][0] = 0$ and $C[0][1] = 0$, because a is infrequent in $D_1$ and in $D_2$ (Steps 7-11). Next, the algorithm computes the error of deleting 1 occurrence of a from $D_1$ (Steps 12-15). The error is stored in $C[1][0]$, as shown in Fig. 2(a), and $I[1][0]$ is set to $\langle 1, -1 \rangle$, as shown in Fig. 2(b). After that, ODESA computes the error of deleting 1 to 13 occurrences from $D_2$ and updates $C$ and $I$ (Steps 16-21). The error for $D_3$, $D_4$ and $D_5$ is also computed and $C$ and $I$ are updated accordingly, as shown in Fig. 2. Thus, $C[13][4]$ contains the optimal sanitization error of $D_5$ and $I[13][4] = \langle 2, 11 \rangle$.

Phase III: ODESA performs backtracking based on the cells of $I$ which correspond to the optimal solution and are shown in Fig. 2(c). Since $I[13][4] = \langle 2, 11 \rangle$, the algorithm deletes 2 occurrences of a from the last SEM and backtracks to $I[11][3]$ (Steps 25-32). The same process is repeated for all other cells in Fig. 2(c) (from right to left), and then ODESA returns the sequence in Fig. 1(b) (Step 34).

**4.3 ESSA: Event Sequence SAnitization.** The ESSA algorithm optimally sanitizes an event sequence, when there are multiple sensitive events. As can be seen in the pseudocode, the algorithm simply applies ODESA to each sensitive event iteratively (Step 3). That is, the result for a sensitive event is given as input to the next iteration. This is performed until the event sequence satisfies the requirement (I) of the ESS problem, at which point the loop in Step 2 terminates. Last, the sanitized event sequence is returned (Step 5).

```
Algorithm: Event Sequence SAnitization (ESSA)
Input: Event sequence $D$, threshold $\delta$, set of sens. events $S$
Output: Sanitized event sequence $D'$
1 $D' \leftarrow D$
2 while there is $s \in \mathcal{S}$ such that $P(s, D'_j) \geq \delta$, for any $D'_j$ do
3     foreach event $s \in \mathcal{S}$ do
4         $D' \leftarrow$ ODESA$(D', \delta, s)$
5 return $D'$
```

Theorem 4.1 establishes that ESSA optimally solves the ESS problem.

THEOREM 4.1. *ESSA finds an optimal solution to the ESS problem.*

*Proof.* (*Sketch*) The proof is based on three properties: (1) $P(s, \tilde{D}'_j) \leq P(s, D'_j)$, where $\tilde{D}'_j$ is a prefix, produced by applying ODESA to $s$, and $D'_j$ is a sanitized prefix in the output of ESSA. This holds because $|\tilde{D}'_j| \geq |D'_j|$. (2) No more occurrences of $s$ than those required by the condition (I) of the ESS problem are deleted, in an iteration of the loop in Step 2. This follows from property (1) and the fact that the minimum number of occurrences of a single sensitive event is deleted by ODESA in an iteration, whereas condition (I) applies to any sensitive event. (3) $P(s, \tilde{D}'_j)$ can only increase in an iteration of the loop in Step 2, when an occurrence of $s$ is deleted from $\tilde{D}'_j$, and $P(s, \tilde{D}'_j) = P(s, D'_j) \geq \delta$, for each $s \in \mathcal{S}$, when the loop terminates. Thus, $P(s, \tilde{D}'_j)$ becomes equal to $P(s, D'_j)$, by deleting the minimum number of occurrences of $s$, for any $s \in \mathcal{S}$ and any $j \in [1, |\mathcal{T}|]$, and the error $E(D)$ is minimum. Consequently, the ESS problem is solved optimally. □

Note that ESSA will always terminate, because the loop in Step 2 can be executed at most $O(fr(s_1, D) + \ldots + fr(s_{|\mathcal{S}|}, D))$ times, when $\mathcal{S} = \{s_1, \ldots, s_{|\mathcal{S}|}\}$. This corresponds to the case when all occurrences of sensitive events must be deleted and only one occurrence is deleted, in an iteration of the loop.

**Complexity.** ESSA needs $O(c \cdot (|D| + |\mathcal{T}| \cdot b^2 \cdot |\mathcal{A}|))$ time and $O(|\mathcal{T}| \cdot (b + |\mathcal{A}|))$ space, as it executes ODESA $c$ times. While $c = O(|D|)$, in practice $c \approx |\mathcal{S}| \cdot \alpha$, where $\alpha$ is the average number of occurrences of a sensitive event in $D$, as shown in our experiments.

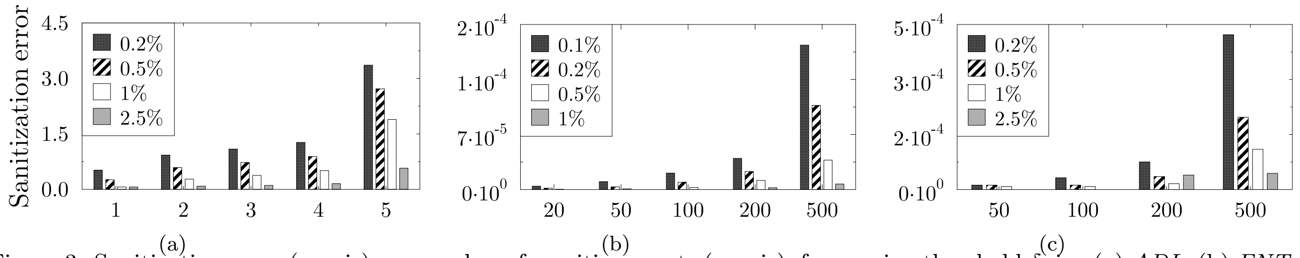


Figure 3: Sanitization error ($y$-axis) vs. number of sensitive events ($x$-axis), for varying threshold $\delta$, in: (a) *ADL*, (b) *ENT*, and (c) *SYN*.

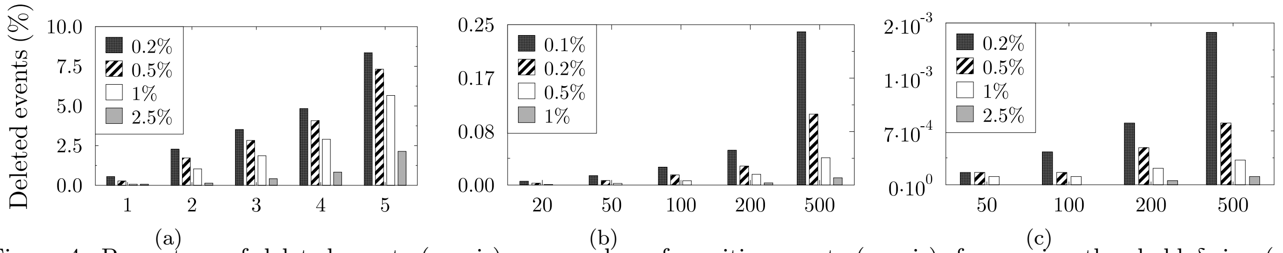


Figure 4: Percentage of deleted events ($y$-axis) vs. number of sensitive events ($x$-axis), for varying threshold $\delta$, in: (a) *ADL*, (b) *ENT*, and (c) *SYN*.

## 5 Experiments

In this section, we evaluate ESSA in terms of data utility and efficiency. We do not compare against existing sanitization methods, because they cannot be used to solve our problem (see Section 6).

To capture data utility, we measure the sanitization error, the percentage of deleted events, and the percentage of *ghost events*. A nonsensitive event $e$ is *ghost*, if it is infrequent in a $j$-prefix of an event sequence $D$, but it is frequent in the sanitized counterpart of the prefix. Ghost events model knowledge that cannot be mined from the prefix but can be mined from its sanitized counterpart. Thus, fewer ghosts imply that data utility is higher. The notion of ghost events appears in [14] but here we adapt it to event sequences.

We implemented ESSA in C++ and applied it to the *ADL* and Entree (*ENT*) datasets (available at `http://archive.ics.uci.edu/ml/`), as well as to a synthetic dataset (*SYN*), created using the IBM data generator [5]. Table 1 shows the characteristics of the datasets and the thresholds we used. The default parameters were $\delta = 0.1\%$ and $|\mathcal{S}| = 200$, and the sensitive events were selected randomly. All experiments ran on an Intel Xeon at 2.4 GHz with 12GB of RAM.

| **Dataset** | $\|D\|$ | $\|\mathcal{A}\|$ | $\|\mathcal{T}\|$ | **Avg.** $\|M_1\|$ | **Max.** $\|M_i\|$ | **Threshold** $\delta$ (%) |
|---|---|---|---|---|---|---|
| *ADL* | 1,448 | 10 | 443 | 3.26 | 17 | 0.2, 0.5, 1, 2.5 |
| *ENT* | 239,790 | 5,005 | 50,364 | 4.76 | 116 | 0.1, 0.2, 0.5, 1 |
| *SYN* | 1,784,984 | 909 | 97,280 | 18.35 | 52 | 0.2, 0.5, 1, 2.5 |

Table 1: Characteristics of datasets and thresholds $\delta$.

**Data utility.** We first measured data utility, based on our utility model. The sanitization error of all datasets, for varying number of sensitive events (i.e., $|\mathcal{S}|$) and threshold $\delta$ is reported in Fig. 3. The error increases when there are more sensitive events and decreases when $\delta$ is larger. This is because more event occurrences need to be deleted in such cases. Furthermore, the errors were larger for *ADL*, because its length and domain size are small. For instance, half of the events are treated as sensitive and each sensitive event must occur at most 2 times in any prefix of *ADL*, when $|\mathcal{S}| = 5$ and $\delta = 0.2\%$.

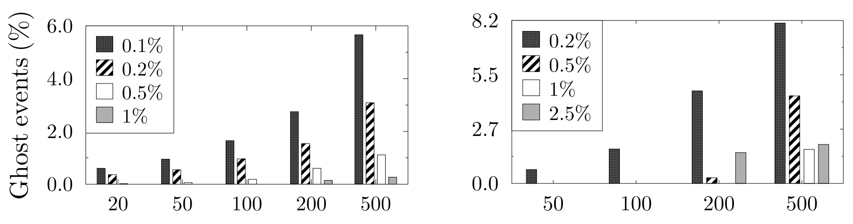
Next, we report, in Fig. 4, the percentage of deleted event occurrences. Note that the percentage was low in all cases (the median percentage was 2%, $9 \cdot 10^{-3}$%, and $2 \cdot 10^{-4}$%, for *ADL*, *ENT*, and *SYN*, respectively, and the maximum percentage was 8.4%) and that it follows the same trend with the error in Fig. 3. This is expected because ESSA optimizes the sanitization error, which increases with the amount of event deletion.

In addition, we demonstrate that ESSA permits accurate frequent event mining. This is because, sanitizing *ADL* as in the previous experiment, did not create ghost events and event occurrences were not deleted from most SEMs (see Fig. 5(a)). Thus, there were minimal changes in the set of frequent events and in their relative frequency. Specifically, the percentage of ghost events for *ENT* and *SYN* was low (the median percentage was 0.56% and 0.49%, respectively) and follows the same trend with the error in Fig. 3. This shows that optimizing the error helps preventing ghosts, which occur when many occurrences are deleted from a prefix.

**Efficiency.** We measured the runtime of ESSA using random event subsequences of increasing length, whose events were contained in all longer subsequences. The results are reported in Fig. 6. It can be seen that ESSA scaled better than quadratically and close to linearly,

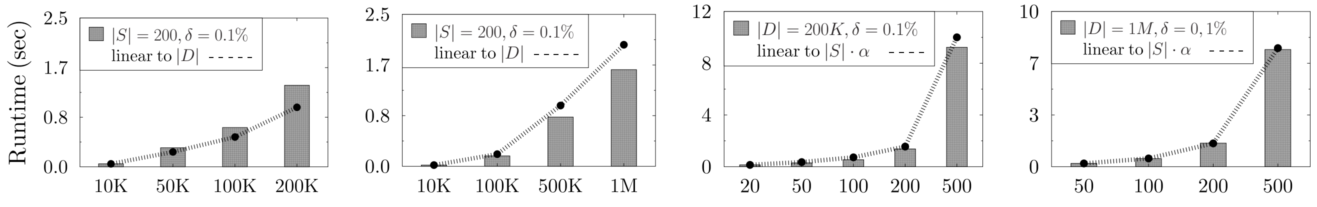
| Event id | Non-affected SEMs (%) |
|---|---|
| 0 | 100% |
| 1 | 100% |
| 2 | 99.7% |
| 3 | 95.72% |
| 4 | 82.43% |

(a)



(b) (c)

Figure 5: (a) SEMs in which no deletion was applied, for each nonsensitive event in *ADL*, when $|\mathcal{S}| = 5$ and $\delta = 0.2\%$. Percentage of ghost events ($y$-axis) vs. number of sensitive events ($x$-axis), for varying $\delta$, in: (b) *ENT*, and (c) *SYN*.



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

Figure 6: Runtime in seconds ($y$-axis) vs. event sequence length ($x$-axis) for: (a) *ENT*, and (b) *SYN*. Runtime vs. number of sensitive events ($x$-axis), for: (c) *ENT*, and (d) *SYN*.

with the length of the event sequence (Figs. 6(a) and 6(b)) and sublinearly with $|\mathcal{S}| \cdot \alpha$ (Figs. 6(c) and 6(d)), as predicted by our analysis in Section 4.3. The impact of the threshold $\delta$ can be seen in Figs. 7(a) and 7(b). More time was needed when $\delta$ was smaller, as more occurrences of sensitive events had to be deleted (all sensitive events had a relative frequency of at least $\delta$ in the event sequence). However, ESSA needed less than 10 seconds, in all tested cases.

We also measured the peak memory consumption of ESSA. The results in Figs. 7(c) and 7(d) show that ESSA scaled well, with respect to the length of the event sequence, and that it required less than 30 MBs of memory. Longer event sequences needed more memory to be sanitized, as they contain more time points.

## 6 Related work

*Data sanitization* (also known as *knowledge hiding*) is an important direction in privacy-preserving data mining that aims to prevent the mining of sensitive knowledge. Existing sanitization approaches are applied to a collection (multiset) of transactions [23, 24, 19], sequences [3, 11, 14], or trajectories [3]. Most approaches prevent the mining of frequent sensitive patterns [23, 3, 11] or association rules [24, 19], by applying deletion. To preserve utility, these approaches attempt to minimize the total number of deleted items (events) [11] and/or changes in the frequency of nonsensitive patterns [3], as well as in the output of frequent pattern [23, 11, 14] or association rule [24, 19] mining. On the contrary, we consider a long sequence of events that are associated with time points and follow differing probability distributions, in different parts of the sequence. Consequently, the utility of the event sequence in mining tasks cannot be preserved based on the number of deleted events, or the frequency of events in the entire sequence, as in existing sanitization methods. To achieve this, our approach minimizes changes in the probability distribution of events, across the sequence. Furthermore, sensitive events may be exposed by applying frequent event mining to prefixes of an event sequence. Therefore, our approach sanitizes each prefix of the sequence, unlike existing methods, which cannot prevent this threat.

*Anonymization* is a different direction in privacy-preserving data mining. Most anonymization approaches are applicable to a collection of transactions [7], trajectories [2], or sequences (with [22] or without [18] time points), each of which is associated with a different individual. Another category of approaches anonymizes an individual's time-series [21] or event sequence [12], using differential privacy [10]. Anonymization approaches guard against the disclosure of an individual's identity [2, 18] and/or sensitive information [7, 2, 22, 21, 12]. However, they cannot be applied to prevent the mining of sensitive events, from each prefix of an event sequence. This is because their privacy models do not limit the relative frequency of sensitive events, in the prefixes of the sequence.

*Suppressing sensitive patterns* from an infinite event sequence has been studied in [15, 25]. These works aim to achieve privacy by minimizing the number of occurrences of sensitive patterns, while preserving the occurrences of certain nonsensitive patterns that are specified by data owners. Both types of patterns are sets of events. However, these works are not applicable to our problem, because they do not aim to prevent the mining of sensitive events and do not consider the utility of the event sequence in mining tasks.

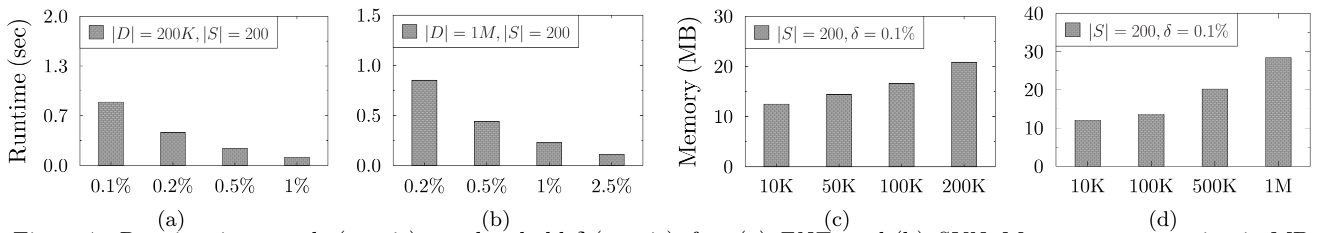


Figure 7: Runtime in seconds ($y$-axis) vs. threshold $\delta$ ($x$-axis), for: (a) *ENT*, and (b) *SYN*. Memory consumption in MBs ($y$-axis) vs. event sequence length ($x$-axis) for: (c) *ENT*, and (d) *SYN*.

## 7 Conclusions

Frequent event mining is a fundamental task in applications that feature event sequences. However, it may lead to the exposure of sensitive events. In this work, we studied how to prevent this threat by deleting events, while preserving the utility of the event sequence. To quantify utility, we proposed a model that captures changes, caused by deletion, to the probability distribution of events across the sequence. Based on the model, we introduced the ESS problem, which seeks to sanitize an event sequence with optimal utility. To solve the problem, we developed an optimal method based on dynamic programming that is effective and efficient, as shown in our experiments.

## 8 Acknowledgments

The work was supported by a RAENG Research Fellowship (Grigorios Loukides) and the EU OpenIOT project (Robert Gwadera).

## References

[1] Supplementary document. `http://users.cs.cf.ac.uk/G.Loukides/sdm15s.pdf`.

[2] O. Abul, , F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385, 2008.

[3] O. Abul, F. Bonchi, and F. Giannotti. Hiding sequential and spatiotemporal patterns. *TKDE*, 22(12):1709–1723, 2010.

[4] A. Achar, S. Laxman, R. Viswanathan, and P.S. Sastry. Discovering injective episodes with general partial orders. *DMKD*, 25(1):67–108, 2012.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, pages 3–14, 1995.

[6] M. Bona. *A walk through combinatorics.* World Scientific, 2006.

[7] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *PVLDB*, 4(11):1087–1098, 2011.

[8] S.F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *ACL*, pages 310–318, 1996.

[9] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. *PVLDB*, 1(2):1530–1541, 2008.

[10] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[11] A. Gkoulalas-Divanis and G. Loukides. Revisiting sequential pattern hiding to enhance utility. In *KDD*, pages 1316–1324, 2011.

[12] M. Götz, S. Nath, and J. Gehrke. Maskit: Privately releasing user context streams for personalized mobile applications. In *SIGMOD*, pages 289–300, 2012.

[13] R. Gwadera, A. Gionis, and H. Mannila. Optimal segmentation using tree models. In *ICDM*, pages 244–253, 2006.

[14] R. Gwadera, A. Gkoulalas-Divanis, and G. Loukides. Permutation-based sequential pattern hiding. In *ICDM*, pages 241–250, 2013.

[15] Y. He, S. Barman, D. Wang, and J. F. Naughton. On the complexity of privacy-preserving complex event processing. In *PODS*, pages 165–174, 2011.

[16] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM TODS*, 28(1):51–55, 2003.

[17] H. Mannila and J. K. Seppänen. Finding similar situations in sequences of events via random projections. In *SDM*, pages 1–16, 2001.

[18] A. Monreale, D. Pedreschi, R. G. Pensa, and F. Pinelli. Anonymity preserving sequential pattern mining. *Artif. Intell. Law*, 22(2):141–173, 2014.

[19] S. R. M. Oliveira and O. R. Zaïane. Protecting sensitive knowledge by data sanitization. In *ICDM*, pages 211–218, 2003.

[20] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[21] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, pages 735–746, 2010.

[22] R. Sherkat, J. Li, and N. Mamoulis. Efficient time-stamped event sequence anonymization. *ACM Trans. Web*, 8(1):4:1–4:53, 2013.

[23] X. Sun and P.S. Yu. A border-based approach for hiding sensitive frequent itemsets. In *ICDM*, pages 426–433, 2005.

[24] V. S. Verykios, A. K. Emagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *TKDE*, 16(4):434–447, 2004.

[25] D. Wang, Y. He, E. Rundensteiner, and J. F. Naughton. Utility-maximizing event stream suppression. In *SIGMOD*, pages 589–600, 2013.

[26] Z. Zhu and W. Du. K-anonymous association rule hiding. In *ASIACCS*, pages 305–309, 2010.

# Optimal event sequence sanitization – Supplementary Document

Grigorios Loukides[*] Robert Gwadera[†]

## 1 Proof of Theorem 3.1 (Sketch)

We first show that the following Constrained Multiple-choice Subset Sum Problem (CMSSP) is weakly NP-hard. Then, we reduce our ESS problem, when $|\mathcal{S}| = 1$, from CMSSP.

CMSSP is defined as:

$$min \sum_{i=1}^{m} \sum_{\forall j \in N_i} w_{ij} \cdot x_{ij} \tag{1.1}$$

subject to:

- (1.2) $\sum_{i=1}^{q} \sum_{\forall j \in N_i} w_{ij} \cdot x_{ij} \geq c_q, \quad q = 1, \ldots m,$
- (1.3) $\sum_{j \in N_i} x_{ij} = 1, \quad i = 1, \ldots m,$ and
- (1.4) $x_{ij} = \{0, 1\}, \quad i = 1, \ldots, m, \quad j \in N_i.$

In CMSSP, we are given a set of elements subdivided into $m$, mutually exclusive classes $N_1, ..., N_m$, and a knapsack. Each class $N_i$ has $|N_i|$ elements. Each element $j \in N_i$ has an integer weight $w_{ij} \geq 0$. The goal is to minimize the total weight (Eq. 1.1) by filling the knapsack with elements whose weights satisfy the constraints of Eq. 1.2, where $c_q \geq 0$, $q \in [1, m]$, and each of these elements belongs to a different class (Eq. 1.3). The variable $x_{ij}$ takes a value 1, if the element $j$ is chosen from class $N_i$ and 0 otherwise. CMSSP can be restricted to the weakly NP-hard Minimization Multiple-choice Subset Sum Problem [4] by allowing only instances where $c_q = 0$, for each $q \in [1, m-1]$.

We can map an instance of CMSSP to an instance of the ESS problem, when $|\mathcal{S}| = 1$, in polynomial time, as follows. We consider an event $e \in \mathcal{S}$ with $w_{i1} + w_{i2} + \ldots + w_{i|N_i|}$ occurrences in $M_i$, for each $i \in [1, m]$. Thus, selecting $x_{ij}$ in CMSSP, which incurs a weight $w_{ij}$, corresponds to deleting $w_{ij}$ occurrences of $e$ from $M_i$. Also, each constant $c_q$, $q \in [1, m]$, corresponds to the number of occurrences of $e$ that need to be deleted from the prefix $D_q$. So, a $\delta$ that is larger than any $\frac{fr(e,D_q)-c_q}{|D_q|-c_q}$, where $q \in [1, m]$, must be chosen.

In the following, we will prove that a sanitized event sequence $D'$ is a solution to the ESS problem, if and only if it corresponds to a solution of CMSSP. Assume that $D'$ is an optimal solution to ESS, which is produced by deleting at least $w_{1j_1}$ occurrences from the prefix $D_1$, at least $w_{2j_2}$ occurrences from $D_2$, ..., at least $w_{mj_m}$ occurrences from $D_m$. As the sanitization error $E(D)$ is minimum, the total weight of the corresponding elements in CMSSP (i.e., $w_{1j_1} \cdot x_{1j_1} + \ldots + w_{mj_m} \cdot x_{mj_m}$) will be minimum as well, hence Eq. 1.1 holds. Also, the total weight of the corresponding elements in $N_i$ will be at least $w_{ij_i}$. Thus, Eq. 1.2 holds, with $c_q = w_{1j_1} + \ldots + w_{qj_q}$, for each $q \in [1, m]$. In addition, Eqs. 1.3 and 1.4 hold, since exactly one element is selected from each class and all elements are 0 or 1. Thus, $D'$ corresponds to a solution of CMSSP. Furthermore, a solution to CMSSP corresponds to a solution of the ESS problem, when $\mathcal{S}$ contains a single event. This is because: (I) the sanitization error $E(D)$ is minimum, since Eq. 1.1 holds, and (II) we delete at least $w_{1j_1}$ occurrences from $D_1$, at least $w_{1j_1} + w_{2j_2}$ from $D_2$, ..., and at least $w_{1j_1} + \ldots + w_{mj_m}$ occurrences from $D_m$, since Eq. 1.2 holds. Thus, $P(e, D'_q) = \frac{fr(e,D_q)-\sum_{i=1}^{q} w_{qj_q}}{|D_q|-\sum_{i=1}^{q} w_{qj_q}}$, for each $q \in [1, |\mathcal{T}|]$. So, the solution to CMSSP corresponds to a solution to the ESS problem, when $|\mathcal{S}| = 1$ and $\delta$ is larger than $max_{q \in [1,|T|]}(P(e, D'_q))$. Thus, the ESS problem is weakly NP-hard, when $|\mathcal{S}| = 1$. □

## 2 Complexity analysis for ODESA

ODESA needs $O\left(|D| + |\mathcal{T}| \cdot b^2 \cdot |\mathcal{A}|\right)$ time, in the worst case. Steps 1-6 need $O(|D|)$ time, because they involve computing $fr(s, D_j)$, for each $D_j$, $j \in [1, |\mathcal{T}|]$. Steps 16-21 need $O(|\mathcal{T}| \cdot b^2 \cdot |\mathcal{A}|)$ time, because Step 18 is executed $O(|\mathcal{T}| \cdot b)$ times. In each execution of Step 18, the $min$ is calculated $O(b)$ times and $E(M^k_{j+1})$ is computed in $O(|\mathcal{A}|)$ time, using a hashtable with key $e$ and value a list $[fr(e, M_1), \ldots, fr(e, M_{|\mathcal{T}|})]$, for each $e \in \mathcal{A}$. The worst-case space complexity of ODESA is $O\left(|\mathcal{T}| \cdot (b + |\mathcal{A}|)\right)$, because the hashtable needs $O(|\mathcal{T}| \cdot |\mathcal{A}|)$ space, and each of the arrays $I$ and $C$ needs $O(|\mathcal{T}| \cdot b)$ space. Thus, both the time and space complexity of ODESA are *pseudopolynomial* [6] in $b$.

## 3 An alternative to the minimum harm approach

Our work avoids the exposure of sensitive events by upper-bounding their relative frequency, in each $j$-

[*]Cardiff University, email: g.loukides@cs.cf.ac.uk
[†]EPFL, email: robert.gwadera@epfl.ch

prefix of a sanitized event sequence $D'$ (see Section 3). This approach is followed by most existing sanitization methods [7, 1, 2, 3, 8, 5] and is referred to as the *minimum harm* approach.

It is also worth noting that our method can be modified to follow an alternative approach, which offers stronger protection but lower data utility [9]. In our setting, this approach requires ensuring that there is a set $\mathcal{N}$ of at least $k-1$ nonsensitive events so that, for each event $e' \in \mathcal{N}$ and each prefix $D'_j$ of the sanitized event sequence, it holds that $P(e', D'_j) = P(e, D'_j)$. This approach aims to prevent an attacker from discovering that $e$ is sensitive, when its relative frequency in a prefix is lower than $\delta$ but higher than that of all other events. Specifically, the approach limits the probability that the attacker will identify $e$ as sensitive to $\frac{1}{k}$, where $k$ is a parameter specified by data owners. To follow this approach, we can easily modify the ODESA algorithm, by limiting the minimum number of deleted occurrences from each SEM.

## References

[1] O. Abul, F. Bonchi, and F. Giannotti. Hiding sequential and spatiotemporal patterns. *TKDE*, 22(12):1709–1723, 2010.

[2] A. Gkoulalas-Divanis and G. Loukides. Revisiting sequential pattern hiding to enhance utility. In *KDD*, pages 1316–1324, 2011.

[3] R. Gwadera, A. Gkoulalas-Divanis, and G. Loukides. Permutation-based sequential pattern hiding. In *ICDM*, pages 241–250, 2013.

[4] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems.* Springer, 2004.

[5] S. R. M. Oliveira and O. R. Zaïane. Protecting sensitive knowledge by data sanitization. In *ICDM*, pages 211–218, 2003.

[6] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[7] X. Sun and P.S. Yu. A border-based approach for hiding sensitive frequent itemsets. In *ICDM*, pages 426–433, 2005.

[8] V. S. Verykios, A. K. Emagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *TKDE*, 16(4):434–447, 2004.

[9] Z. Zhu and W. Du. K-anonymous association rule hiding. In *ASIACCS*, pages 305–309, 2010.