On the Complexity of Privacy-Preserving Complex Event Processing

Yeye He
University of Wisconsin-Madison
heyeye@cs.wisc.edu

Siddharth Barman
University of Wisconsin-Madison
sid@cs.wisc.edu

Di Wang
Worcester Polytechnic Institute
diwang@cs.wpi.edu

Jeffrey F. Naughton
University of Wisconsin-Madison
naughton@cs.wisc.edu

ABSTRACT

Complex Event Processing (CEP) Systems are stream processing systems that monitor incoming event streams in search of user-specified event patterns. While CEP systems have been adopted in a variety of applications, the privacy implications of event pattern reporting mechanisms have yet to be studied — a stark contrast to the significant amount of attention that has been devoted to privacy for relational systems. In this paper we present a privacy problem that arises when the system must support desired patterns (those that should be reported if detected) and private patterns (those that should not be revealed). We formalize this problem, which we term privacy-preserving, utility maximizing CEP (PP-CEP), and analyze its complexity under various assumptions. Our results show that this is a rich problem to study and shed some light on the difficulty of developing algorithms that preserve utility without compromising privacy.

Categories and Subject Descriptors: F.2 Analysis of Algorithms and Problem Complexity

General Terms: Algorithms, Theory

Keywords: CEP Stream Processing, Privacy, Complexity

1. INTRODUCTION

Complex Event Processing (CEP) is a stream event processing paradigm that has received increasing attention from the data management research community [4, 5, 7, 17, 21, 22] and also from industry [1, 2, 3]. In the CEP model, the data is a stream of events, which is monitored and queried in search of some user-defined event patterns. When a pattern of interest is detected, it is reported by the CEP system. Such CEP systems have demonstrated utility in a variety of applications including financial trading, credit-card fraud detection, and security monitoring. However, to our knowledge the problem of privacy in such systems has not yet been addressed.

In a nutshell, the problem we consider is this: consider two kinds of patterns, those that should be detected and reported (which we term "public" patterns), and those that should be eliminated and not reported (which we term "private" patterns). The decision of what is public and what is private is made by a user or administrator and is input to the system. Then the task is to "destroy" any private patterns that occur by deleting some events from the stream; however, we wish to do so in such a way that maximizes the number of public patterns that remain to be detected. The challenge of the problem arises from the number of options as to what should be retained or deleted.

We begin with a motivating example inspired by a health care monitoring system, HyReminder [20]. HyReminder, currently being deployed in the University of Massachusetts Memorial Hospital, is a hospital infection control system powered by CEP technologies that aims to track, monitor and remind health-care workers with respect to hygiene compliance. In this hospital setting, each health care worker wears an RFID tag that can be read by sensors installed throughout the hospital. As the worker walks around the hospital, the RFID tags they wear are detected by the sensors, generating "event" data, which is transfered to a monitoring central CEP engine. As an example of such a pattern, a doctor who exits a patient room (represented by an "exit" event), cleanses his hands (indicated by a "sanitize" event), and finally enters an ICU (an "enter" event), generates an event pattern `SEQ(exit, sanitize, enter)`, showing an instance of hygiene regulations compliance. Conversely, an instance of hygiene violation in which the doctor exits the patient room, does *not* cleanse his hands before entering an ICU, would correspond to an event sequence `SEQ(exit, !sanitize, enter)` where "!sanitize" indicates the absence of the "sanitize" event. Both the hygiene compliance and violation instances should be captured and reported by the CEP system.

While the benefit of such CEP systems is apparent, the implications of this CEP querying model for privacy are somewhat subtle. Intuitively, while it is clear that a CEP system may reveal useful event patterns, it may also reveal patterns that the individuals being monitored would prefer to keep hidden. Using the hospital example again, a doctor who visits a patient then immediately enters a psychiatrist's office might serve as an indication that this patient is experiencing psychiatric problems. While a system like HyReminder will not directly monitor and report such private event patterns, the occurrences of hygiene violations it does report may be used by an adversary to infer such private patterns.

In this paper we study an abstract problem inspired by privacy-preserving, utility maximizing CEP. We classify patterns of interest into two categories: the *query patterns*, which should be detected

PODS'11, June 13–15, 2011, Athens, Greece.

and reported; and the *private patterns*, which should be suppressed. The goal of privacy-preserving CEP is to maximize the reporting of query patterns without disclosing any private patterns. To be more concrete, in this paper we explore filtering or dropping events to eliminate the occurrence of private patterns; the "trick" is to do so intelligently, so as to avoid dropping events as much as possible while eliminating the occurrence of query patterns.

This abstract problem has many interesting variants. We begin by studying two of the most straightforward variants: the *windowed* variant, in which, when deciding which events to drop, we consider the events that arrived in a specified window preceding the current event; and the *oblivious* variant, where prior events are not taken into consideration. Both variants are interesting to study: while the *windowed* variant can potentially better preserve utility, it is also intuitively more computationally expensive.

To our knowledge, no previous studies — practical or theoretical — have investigated PP-CEP. We observe that the problem has a simple but rich structure, and point out an interesting array of complexity results. Specifically, the general problem of PP-CEP is NP-hard and hard to approximate (namely, it is $|\Sigma|^{1-\epsilon}$-inapproximable for utility gain by constructing a reduction from independent set, where $|\Sigma|$ is the size of the events, and $\ln(|\mathcal{P}|)$-inapproximable for utility loss by constructing a reduction from set cover, where $|\mathcal{P}|$ is the number of private patterns). However, we explore various combinations of the input parameters and show that there are a variety of scenarios in which the complexity can be reduced. For instance, the oblivious PP-CEP problem can be formulated as an integer program that is polynomially solvable when the sum of the number of private patterns and query patterns is some fixed constant, or when the sum of the number of the event types and query patterns is some fixed constant. In addition, exploiting the submodularity of the utility loss function we prove that the oblivious PP-CEP problem is l_p-approximable where l_p is the maximum length of the private patterns. We also show that in the special case where the window size is some fixed constant the windowed PP-CEP is also l_p-approximable.

These complexity results shed some light on the difficulty of developing PP-CEP solutions in practice, but more importantly also leave open many more possible avenues for future research. We hope that our first step towards understanding the PP-CEP problem will serve as a springboard for future work tackling the theoretic and practical issues that arise from the problem of PP-CEP.

2. A SEQUENCE-BASED CEP MODEL

2.1 The data/query model

We adopt the following event sequence-based model to represent continuously arriving events. Let the domain of possible event types be the alphabet $\Sigma = \{e_i\}$, where e_i represents a type of event. We model a stream of events as follows.

DEFINITION 1. *A **time-stamped event sequence** is a sequence $S = (c_1, c_2, ...c_n)$, where each event c_j is associated with a particular event type $e_i \in \Sigma$, and has a unique time stamp t_j. The sequence S is temporally ordered, that is $t_j < t_k$ for all $j < k$.*

While logically we consider only one temporally ordered event sequence, in practice events can come from multiple data sources. A common complication that can arise is that events from multiple sources may arrive out of order. In this work we make the simplifying assumption that events in the sequence are in strictly increasing time order, and there are no out-of-order events in the sequence. This assumption helps us to better focus our discussions on the privacy aspect of CEP systems, and is consistent with the state-of-the-art CEP literature [4, 5, 17, 22]. In practice, existing work that use buffer-based techniques for out-of-order event streams processing [7, 16] can be similarly applied to handle out-of-order events in our problem.

We further define *subsequence* which is an ordered subpart of a given sequence.

DEFINITION 2. *A **subsequence** S' of the event sequence $S = (c_1, c_2, ...c_n)$ is a sequence $(c_{i_1}, c_{i_2}, ...c_{i_m})$ such that $1 \leq i_1 \leq i_2 ... \leq i_m \leq n$, where $m \leq n$.*

Note that subsequence preserves the temporal order of the original sequence. Also observe that unlike substrings, a subsequence does not have to be a consecutive subcomponent of the sequence. As a result, there are a total of 2^n possible subsequences.

We next define queries over event sequences. Unlike relational data processing, where the queries are typically ad-hoc and dynamically composed at query time, queries in the CEP model are standing queries, which are submitted ahead of time and are static. Let the set of queries be $\mathcal{Q} = \{Q_i\}$, where each query Q_i is a sequence query defined as follows.

DEFINITION 3. *A **sequence query pattern** Q_i is of the form $Q_i = (q_1^i, q_2^i, ...q_n^i)$, with $q_k^i \in \Sigma$ being event types. In addition, for each query Q_i there is a specified window of size $T(Q_i) \in \mathbb{R}^+$ over which Q_i will be evaluated.*

A sequence query looks for the conjunction of the occurrences of events of certain types in specified order within a given time window. This sequence query is essentially the "`SEQ`" query construct in the CEP query language used by most existing systems [10, 17, 22]. While negation of an event can also be included in "`SEQ`" queries to indicate the absence of event occurrences, we in this work only consider positive event occurrences and leave negation of events as future work.

DEFINITION 4. *A sequence query pattern $Q_i = (q_1^i, q_2^i, ...q_n^i)$ with window size $T(Q_i)$ produces an instance of **query pattern match** over an event sequence S, if there exists a subsequence $S' = (c_{j_1}, c_{j_2}, ... c_{j_n})$ of S, such that c_{j_l} is of type q_l^i for all $l \in [1, n]$, and $t_{j_n} - t_{j_1} \leq T(Q_i)$. We say the sequence of events S' is an instance of match of Q_i. We denote $M(Q_i, S)$ as the set of all instances of matches produced for query Q_i over S.*

We use the following example to illustrate event sequences and query matches in our CEP model.

EXAMPLE 1. *Suppose there are five event types denoted by the five characters $\Sigma = \{A, B, C, D, E\}$. Each character represents a certain type of event (for example, A could stand for doctors entering the ICU, B denotes doctors washing hands at the sanitization desks, and so forth).*

Let the stream of arriving events be $S = (A_1, B_2, A_3, C_4, D_5, D_6, C_7, A_8, D_9, E_{10})$ where each character is an instance of a particular type of event arriving at time stamp t_i given subscript i. Assume in this example that events arrive at fixed intervals t for simplicity. In other words, A_1 arrives at $t_1 = t$, B_2 arrives at $t_2 = 2t$, so on and so forth.

Suppose there are two query patterns: $Q_1 = (A, B)$ with the associated window size $T(Q_1) = t$, and $Q_2 = (C, D)$ with $T(Q_2) = 2t$. To give some interpretation for the queries, if A stands for doctors entering the ICU, B for doctors washing hands, then Q_1 is looking for such patterns where doctors wash hands immediately

after entering the ICU, and the time interval between the two events is within the window size $T(Q_1) = t$.

Over the event sequence S, there is only one subsequence (A_1, B_2) that matches Q_1. Note that the subsequence (B_2, A_3) does not match Q_1, because by definition a subsequence that matches for Q_1 has to maintain that an event of type A arrives before an event of type B.

There are three matches of Q_2, namely (C_4, D_5), (C_4, D_6), and (C_7, D_9). Observe that the subsequence (C_4, D_9) does not match Q_2 because the time interval between C_4 and D_9 is $9t - 4t = 5t$, which is larger than $T(Q_2) = 2t$.

DEFINITION 5. *A **CEP** system is a function that takes as input an event sequence S, a set of query patterns $\mathcal{Q} = \{Q_i\}$, and outputs a set $\mathcal{M} = \bigcup_{Q_i \in \mathcal{Q}} M(Q_i, S)$, where $M(Q_i, S)$ is the set of all query matches of Q_i over S.*

In this work we focus on the CEP that only accept sequence query patterns. While in practice CEP systems may also include support for query language extensions like negation [10] and Kleene closure [10], among other features, we in this work limit the query language to allow only conjunctions of positive event occurrences and study the complexity of different variants based on this core language subset. Investigating the privacy implication of CEP on this restricted language subset already presents us with a rich problem to study. Expanding the query language to a more expressive one and exploring its impact on complexity is an interesting direction for future work.

2.2 Private patterns

We now turn to introduce the notion of private patterns. Intuitively, private patterns are just like query patterns. The fundamental difference, however, is that as many query patterns as possible should be detected, whereas there is a strict requirement that no private patterns should be reported.

There are multiple ways in which an adversary could compromise privacy in search of private patterns. The most intrusive way is to break into the CEP system and monitor the arriving event streams. Preventing this type of adversary involves addressing security aspects of the CEP system and is out of the scope of our work. Accordlingly, we assume the CEP system is secure and focus on adversaries who observe the reported matches for the query patterns, and only infer the occurrences of the private patterns from the *externally observable event sequence*, which is a "union" of all the events reported in the query results. We first define externally observable event sequences.

DEFINITION 6. *Let S be a time-stamped event sequence. Let $M = \{M_i\}$ be the set of matches in S produced by a CEP system with respect to the query set $\mathcal{Q}$, that is, $\forall i, M_i$ is a match of some query in $\mathcal{Q}$. The sequence of events obtained by unioning the events in M is called the **externally observable event sequence**. We denote the externally observable event sequence of the original sequence S with respect to the query set $\mathcal{Q}$ as $\overline{S} = O(S, \mathcal{Q})$.*

The semantics for matches of private patterns $\mathcal{P}$ over a given event sequence S are similar to matches of query patterns $\mathcal{Q}$, but they are defined over the external observable event sequence $\overline{S} = O(S, \mathcal{Q})$, instead of the original event sequence S. Specifically, let the set of private patterns be $\mathcal{P} = \{P_i\}, 1 \leq i \leq |\mathcal{P}|$. Each private pattern is defined as follows.

DEFINITION 7. *A **sequence private pattern** P_i is of the form $(p_1^i, p_2^i, ...p_n^i)$, with $p_k^i \in \Sigma$ being event types in the alphabet. Let $T(P_i) \in \mathbb{R}^+$ be the window size for each query P_i. There is an instance of **private pattern match** of P_i over a given event sequence S, if there exists a sequence $\overline{S}' = (c_{j_1}, c_{j_2}, ...c_{j_n})$, that is the subsequence of the externally observable event sequence $\overline{S} = O(S, \mathcal{Q})$, such that $c_{j_l} = p_l^i$ for all $l \in [1, n]$, and $t_{j_n} - t_{j_1} \leq T(P_i)$. We say the sequence $\overline{S}'$ is an instance of match of P_i.*

Clearly any match of private pattern over the externally observable event sequence constitutes a breach of privacy and should be disallowed. We use the following example to illustrate matches of private patterns.

EXAMPLE 2. *Given the event sequence S and the query patterns Q_1 and Q_2 in Example 1, suppose there are also two private patterns $P_1 = (B, C)$ with $T(P_1) = 2t$, $P_2 = (D, E)$ with $T(P_2) = t$.*

Recall that in Example 1, four instances of the query matches are reported, namely (A_1, B_2), (C_4, D_5), (C_4, D_6) and (C_7, D_9). Using the time-stamps associated with each event, an adversary observing the set of query results will be able to reconstruct the externally observable event sequence in temporal order $\overline{S} = (A_1, B_2, C_4, D_5, D_6, C_7, D_9)$, which is a subsequence of the original event sequence $S = (A_1, B_2, A_3, C_4, D_5, D_6, C_7, A_8, D_9, E_{10})$.

In this example, one instance of matches for P_1 can be produced over $\overline{S}$, namely (B_2, C_4), which results in a breach of privacy. However, there is no match for $P_2 = (D, E)$ with $T(P_2) = t$ over $\overline{S}$, even though in the real event stream S that passes through the CEP engine there is an instance of match (D_9, E_{10}) for P_2. Because we assume that the adversary is only able to observe the reported query results to construct $\overline{S}$, and has no direct knowledge of S, or more specifically the existence of E_{10}, no matches for P_2 can be produced by the adversary.

Since we restrict queries to be simple conjunction of event occurrences, certain events that participate in private patterns have to be suppressed in order to ensure that no private patterns are disclosed. While alternative strategies are possible with an extended query language (inserting fake events, for instance, may be used to prevent private patterns when the private pattern includes negation), in this work we only consider the strategy in which events are suppressed.

DEFINITION 8. *For fixed query and private sets, $\mathcal{Q}$ and $\mathcal{P}$ respectively, a **PP-CEP** system is a function which, for any event sequence S, produces a subsequence S' of S such that no matches of private pattern in $\mathcal{P}$ can be produced over the externally observable sequence $\overline{S} = O(S', \mathcal{Q})$*

Apparently there are various kinds of PP-CEP in which different events can be suppressed to ensure privacy. For instance, in Example 1, given that there is a match of private patterns $P_1 = (B, C)$, the intuitive way to suppress such match is to drop either of the events B or C. In order to measure the relative merits of different strategies and to ultimately determine a preferable strategy, we define a utility function.

2.3 The utility function

We quantify utility based on the number of instances of query matches reported. In addition, because each query pattern may have different real-world importance, we differentiate between queries using a weight function $w(Q_i)$, which measures the importance of reporting an instance of matching Q_i. We define the query utility as follows.

DEFINITION 9. *Let $\mathcal{C}(Q_i, S)$ be the number of distinct matches for query Q_i in S. The **utility** generated for query pattern Q_i is*

$$U(Q_i, S) = w(Q_i) \cdot \mathcal{C}(Q_i, S) \quad (1)$$

The sum of the utility generated over $\mathcal{Q} = \{Q_i\}$ is

$$U(\mathcal{Q}, S) = \sum_{Q_i \in \mathcal{Q}} U(Q_i, S). \quad (2)$$

As we have discussed, there are multiple ways in which events can be suppressed to prevent private pattern matches. The utility metric can be used to choose the strategy that preserves the most utility. We will illustrate the concept of utility in Example 3.

EXAMPLE 3. *We continue to use the event stream and query/private patterns of the Example 2. To measure utility, assume for simplicity that the weight associated with query Q_1 and Q_2 is one. Given that there is one instance of match for Q_1 and three matches for Q_2, the total utility gain over $S = (A_1, B_2, A_3, C_4, D_5, D_6, C_7, A_8, D_9, E_{10})$ without privacy consideration is thus $1 + 1 \times 3 = 4$.*

As we have mentioned, in order to suppress the one match produced for $P_1 = (B, C)$ over $\overline{S} = (A_1, B_2, C_4, D_5, D_6, C_7, D_9)$ (namely, (B_2, C_4)), there are two possible event dropping strategies: we could either drop B_2, or drop event C_4. If B_2 is dropped, we will derive $\overline{S}_1 = (A_1, C_4, D_5, D_6, C_7, D_9)$. With that suppression there are a total of three matches for Q_2 left (the match (A_1, B_2) for Q_1 is dropped due to the suppression of B_2), yielding a utility of 3.

Alternatively, the event C_4 can be dropped. This gives us $\overline{S}_2 = (A_1, B_2, D_5, D_6, C_7, D_9)$. With this suppression strategy there is one match of Q_1 ((A_1, B_2)) and one match of Q_2 ((C_7, D_9)) retained, thus the utility for this strategy is $1 + 1 = 2$.

Apparently, while both strategie suppress matches for private patterns, dropping B_2 is the better strategy as measured by the amount of utility it preserves.

Ultimately, the choice of dropping event B or C in the previous example depends on the characteristics of the arriving events in general, or alternatively the distribution of the event types. Intuitively, if C arrives much more frequently than B, and is more likely to produce query match for $Q_2 = (C, D)$, then it may be a good idea to drop B instead of C, as the utility preserved for Q_2 can outweigh the loss of Q_1.

In order to quantify the "expected" query matches to estimate expected utility, we need statistical information about the arriving events. In this work we make the simplifying assumption that the arrival of each type of event is an independent Poisson process, which is typical statistical assumption and is used to model event arrival in performance modeling literature [14]. Furthermore, we assume that the arrival rate λ_i of each Poisson process is known to us. In practice, such an arrival rate can be estimated by sampling the arriving events.

Further, let $|\Sigma|$ be the number of event types, $|\mathcal{Q}|$ and $|\mathcal{P}|$ number of query patterns and private patterns, respectively. Denote by l_q and l_p the maximum length of query patterns and private patterns, respectively, and finally d_q and d_p the maximum number of query patterns/private patterns in which any type of event $e_i \in \Sigma$ participates. We summarize the symbols used in this paper in Table 1.

The end goal of PP-CEP is to find an event suppression strategy such that utility can be maintained as much as possible without compromising privacy. In the following we will discuss in detail two variants that arise from the general formulation.

Σ	The set of all possible event types
e_i	Event of type i
λ_i	The arrival rate of e_i
$\mathcal{Q}$	The set of query patterns
Q_i	Query pattern i
$T(Q_i)$	The time window associated with Q_i
$w(Q_i)$	The utility weight associated with Q_i
q_k^i	The k-th participating event in query pattern Q_i
$\mathcal{P}$	The set of private patterns
P_i	Private pattern i
$T(P_i)$	The time window associated with P_i
$w(P_i)$	The utility weight associated with P_i
p_k^i	The k-th participating event in private pattern P_i
l_q	The maximum length of any query patterns
l_p	The maximum length of any private patterns
d_q	The maximum number of query patterns that any event participates
d_p	The maximum number of private patterns that any event participates

Table 1: Summary of the symbols used

3. THE OBLIVIOUS PP-CEP

The first variant of privacy-preserving CEP we discuss is the *oblivious* PP-CEP.

DEFINITION 10. *Let $\mathcal{P} = \{P_i\}$ be the set of private patterns, with $P_i = (p_1^i, p_2^i, ...p_n^i)$. An **oblivious PP-CEP** is a PP-CEP that suppresses all events of type $e_i \in D$, $D \subseteq \Sigma$, while preserving events in $K = \Sigma \setminus D$, such that for all private pattern P_i, at least one participating event p_j^i is suppressed, or $p_j^i \in D$.*

An oblivious suppression decision drops at least one participating event type from the private patterns. This is a sufficient condition for privacy — if all instances of some event type that participates the private pattern are dropped, no matches for that private pattern can be produced. Oblivious suppression is a global approach that preserves privacy irrespective of the event sequence that may occur, that is, regardless of what has just arrived and what may arrive in the future. The notion of obliviousness is in contrast to the *windowed* PP-CEP that we will discuss in the next section, which also takes into consideration the events that have just arrived in the local window to devise an event suppression strategy. We first use the following example to demonstrate the concept of oblivious PP-CEP.

EXAMPLE 4. *Continuing with Example 3, we illustrate oblivious PP-CEP using the private pattern $P_1 = (B, C)$. Recall that an oblivious PP-CEP decision suppresses at least one participating event type in each private pattern. In this case it could either drop event type B or C.*

Specifically, given the sequence $\overline{S} = (A_1, B_2, C_4, D_5, D_6, C_7, D_9)$, one could either drop all events of type B, or drop all events of type C. Observe that either oblivious decision is sufficient for privacy — if all instances of B (similarly, C) are dropped, the private pattern $P_1 = (B, C)$ will not be disclosed. However oblivious event suppression is not necessary for privacy. In particular, if we decide to drop events of type C, C_7 does not have to be dropped from $\overline{S} = (A_1, B_2, C_4, D_5, D_6, C_7, D_9)$, because based on the events that have arrived at time-stamp t_7 it is clear that preserving C_7 will not compromise privacy (this is because there is no event of type B that arrives between t_5 to t_7, or $T(P_1) = 2t$ prior to C_7).

As a matter of fact, not dropping C_7 allows us to produce one more query match (C_7, D_9) and improves utility.

This example illustrates the key idea of oblivious suppression. Oblivious PP-CEP devise an event suppression strategy that is sufficient, but not always necessary, to preserve privacy. Precisely due to the simplicity in its obliviousness, in practice we expect it to have lower added computation overhead to an existing CEP system than a more sophisticated PP-CEP system (for example, the windowed PP-CEP that we will discuss in Section 4). This property may be especially desired for the CEP model, which typically requires real-time query processing and low computational overhead [5, 10, 17, 22]. However, as we will show, even this oblivious PP-CEP problem is very difficult in terms of computational complexity. This problem of utility-maximizing oblivious PP-CEP is formally defined as follows.

DEFINITION 11. *Let $\Sigma = \{e_i\}$ be the domain of event types where each type of event e_i arrives following an independent Poisson process with arrival rate λ_i. Let the set of query patterns and private patterns be $\mathcal{Q} = \{Q_j\}$ and $\mathcal{P} = \{P_k\}$, respectively. The utility of an oblivious PP-CEP that preserves events $K \subseteq \Sigma$ is denoted as $\mathcal{G}(\mathcal{Q}, K, S) = U(\mathcal{Q}, \Pi_K(S))$, where S is the given event sequence, and $\Pi_K(S)$ is the projection of S which only preserves events in K and none other. The problem of* ***utility gain maximizing oblivious PP-CEP*** *given an event sequence S is to find the oblivious PP-CEP with the maximum utility $\mathcal{G}(\mathcal{Q}, K, S)$. Similarly we can define* ***utility loss minimizing oblivious PP-CEP*** *as the one that minimizes the utility loss $\mathcal{L}(\mathcal{Q}, K, S) = U(\mathcal{Q}, S) - U(\mathcal{Q}, \Pi_K(S))$.*

Given an event sequence S we can choose over the space of all possible oblivious suppression decisions to find the optimal one with respect to utility. Observe that in Definition 11 we deliberately introduce two utility metrics to optimize, the utility gain metric $\mathcal{G}$ and the utility loss metric $\mathcal{L}$. While they are essentially equivalent to each other and reach the optimal point at the same time, their approximability results are different as we will discuss in detail. We will see that it is easier to approximate the utility loss function than to approximate the utility gain function.

It is worth noting that the event sequence S we consider is the most likely sequence constructed based on the event arrival rate λ_i, and we optimize the utility only based on the expected count of query matches over this most likely event sequence rather than the analysis of the space of all possible event sequence. This technique of using expected counts of random variables to produce simple estimates is used in [12], although more sophisticated and complex techniques like *stochastic programming* [12] can be used to account for the random variables more accurately.

Our reason for using expected counts over stochastic programming is twofold. First, since the arrival rate is already an estimate produced by sampling, over-complicating the problem formulation by using stochastic programming for extra accuracy seems to be overkill. Furthermore, recall that our goal is to explore the difficulty of the PP-CEP problem rather than to produce an accurate utility estimate. And as we will see, using the simple expected count to estimate utility for each query already renders the problem hard in general.

In the following we will first analyze the complexity of the general oblivious PP-CEP and show inapproximability results in Section 3.1. We will then descend into a number of special cases with assumptions on various parameters, and show that for each case how the complexity of the problem can be reduced.

3.1 A general complexity analysis

We study in this section the complexity of the general version of the oblivious PP-CEP problem. Again, rather than considering the stochastic event sequence, we in this work only investigate the complexity of the PP-CEP problem over the expected event sequence that may arise given the arrival rates. As we will see in the following, even with this simplification we can already obtain a number of negative complexity results for the general version of the problem.

THEOREM 1. *The problem of utility gain maximizing oblivious PP-CEP is NP-hard.*

Proof Sketch. We use a reduction from the independent set problem to prove the hardness result. Recall that a set of vertices $V' \subseteq V$ in a graph $G = (V, E)$ is said to be an independent set if no two vertices in V' are adjacent in the graph G (or $\forall u, v \in V'$, $(u, v) \notin E$). The maximum independent set is the independent set with the largest size.

Given an instance of the independent set problem over a graph $G = (V, E)$, we construct an instance of oblivious PP-CEP as follows. We set the symbol set Σ to be the vertex set V, that is we place a symbol corresponding to each vertex in the graph. In addition we set the event sequence S as a list of all symbols in the alphabet in any order, to correspond to all the vertices in the graph. For each edge (u, v) in E we add a private pattern $P_i = (u, v)$ (and (v, u)) into the set of private patterns $\mathcal{P}$. Finally let all vertices $v \in V$ be a query pattern $Q_j = (v)$ in the set of query patterns $\mathcal{Q}$. Let all query patterns $Q_j \in \mathcal{Q}$ have the unit utility weight, $w(Q_j) = 1$ for all j, so that each instance of query match will produce unit utility 1.

Say a solution with utility k or more for the constructed oblivious PP-CEP involves dropping events $D \subseteq \Sigma$ while keeping events $K = \Sigma \setminus D$. Note that the feasibility of D ensures that no private pattern is disclosed. Then the vertex set V_K that corresponds to the set of events K must be an independent set of size k (recall that there is a one-to-one mapping between the event types in the PP-CEP problem we construct and the vertices in the graph).

Overall if dropping D, equivalently reporting all of K, preserves privacy, then no private pattern is a subsequence of K. If this is not the case then there is a pair of vertices $u, v \in V_K$ such that $(u, v) \in E$, which means that both the events corresponding to u and v are kept in the event set K. Since $P = (u, v)$ is a private pattern, this would have violated the privacy constraints, contradicting the fact that keeping K is privacy preserving. Since the solution achieves a utility of k or more, this implies $|K|$ is at least k. This follows from the fact that each alphabet is a query pattern with unit utility. In the other direction if the original graph has an independent set I of size k the we can drop the symbols of the alphabet corresponding to vertices in $V \setminus I$ to obtain a privacy preserving solution of utility k.

Hence the constructed oblivious PP-CEP instance has a solution achieving utility of k or more if and only if the original graph has an independent set of size k or more. This shows that the oblivious PP-CEP is NP-hard. □

Note that although in our proof, oblivious PP-CEP algorithms consider the whole expected event sequence as input, they are not really off-line algorithms, which only produce solutions when the whole input event sequence is seen. Instead, the input event sequence from which suppression strategies are derived is the expected input that can be known a prior based on event arrival rate. Oblivious PP-CEP algorithms simply operate based on the strategies so produced independent of the real input seen in a particular random experiment, and are thus on-line algorithms. This is con-

sistent with the real-time event processing requirement essential to CEP.

We further show some inapproximability results for both the utility gain maximization problem and the utility loss minimization problem of oblivious PP-CEP.

THEOREM 2. *There is a fixed constant $\epsilon \geq 0$ such that if there is a $|\Sigma|^{1-\epsilon}$ factor approximation algorithm for utility gain maximizing oblivious PP-CEP problem, then $P = NP$.*

Proof Sketch. We note that the reduction in Theorem 1 is approximation preserving. In particular, the reduction from an independent set problem over graph G of n vertices produces an instance with $|\Sigma| = n$. Also, the value of optimal solution of the PP-CEP instance is no less than the size of the maximum independent set of G and a privacy preserving solution with utility of k implies an independent set of size at least k. This proves that PP-CEP is as hard as independent set, as an α factor approximation algorithm for PP-CEP gives an α factor approximation for independent set. Using the inapproximability of independent set [19] we get the desired result. □

In addition to the $|\Sigma|^{1-\epsilon}$-inapproximability, we have an alternative inapproximability result based on the parameters l_q, d_p and d_q.

THEOREM 3. *There is a fixed constant $\epsilon \geq 0$ such that if there is a $(1-\epsilon)(l_q d_q d_p - 2)$ factor approximation algorithm for utility gain maximizing oblivious PP-CEP problem, then $P = NP$.*

Proof Sketch. This second inapproximability result is based on l_q (the maximum length of the query patterns), d_q and d_p (the maximum number of query/private patterns that any event participates). This result follows from the inapproximability result shown in [19], which states that for independent set, the approximation factor cannot be better than $(d-2)$, where d is the edge degree of the graph. We show the inapproximability of $(1-\epsilon)(l_q d_q d_p - 2)$ by contradiction. If there is an algorithm that has approximation factor of $(1-\epsilon)(l_q d_q d_p - 2)$, using the approximation preserving reduction from independent set the corresponding independent set problem instance would also have an approximation factor of $(1-\epsilon)(l_q d_q d_p - 2)$. Given that $d_q = 1$, $l_q = 1$, and d_p being the degree of the graph in our reduction construction, we would have an approximation factor better than $d-2$. This contradicts [19]. We cannot approximate beyond the factor of $(l_q d_q d_p - 2)$ as a result. □

THEOREM 4. *For any constant $\delta > 0$, if there is a $(1-\delta) \times \ln(|\mathcal{P}|)$ factor approximation algorithm for utility loss minimizing oblivious PP-CEP problem then $NP \subseteq DTIME(n^{O(\log \log n)})$.*

Proof Sketch. We provide an approximation preserving reduction from set cover. The claim then follows from the inapproximability of set cover [8].

Given a universe U of n elements and a collection of m subsets of U, $\mathcal{C} = \{S_i\}_{i=1}^{m}$, the cardinality set cover problem is to determine if $\mathcal{C}$ contains k or fewer subsets such that their union covers all the elements. The reduction to utility loss minimizing oblivious PP-CEP is as follows: we construct an alphabet set Σ of size m, with an alphabet symbol e_i for each set S_i in the collection. A private pattern P_j corresponding to each element $j \in U$ is introduced. Say element j is contained in p distinct sets: $S_{i_1}, S_{i_2}, \ldots, S_{i_p}$ with $i_1 < i_2 < \ldots < i_p$, we set private pattern P_j to be $(e_{i_1}, e_{i_2}, \ldots e_{i_p})$ with unbounded time window. The query patterns are simply the m alphabets of Σ with unit utility and unbounded time window. Finally the sequence S is set to be $(e_1, e_2, \ldots, e_m)$.

First note that if the set cover instance has a solution of size k' or less then we can achieve a solution for the PP-CEP problem with utility loss of k'. In particular, say $\mathcal{K} \subseteq \mathcal{C}$ is a collection of subsets covering U. Then we can drop the symbols of the alphabet corresponding to sets in $\mathcal{K}$ to achieve a privacy preserving solution with utility loss of $|\mathcal{K}|$. This follows from the fact that if we drop $|\mathcal{K}| = k'$ alphabet symbols then the loss in utility is exactly k', as each query pattern is an alphabet symbol with utility of one. Each element j is covered by at least one subset in $\mathcal{K}$, say S_a, which implies that dropping alphabet symbol e_a would preserve the privacy of pattern P_j. This overall implies that the generated solution is privacy preserving.

In the other direction, a privacy preserving solution for the PP-CEP instance with a utility loss of k' implies a set cover of size k'. Say we drop all the alphabet symbols in D. Then the utility loss is exactly $|D|$. The relevant observation is that the sets corresponding to alphabet symbols in D form a cover of U. For any private pattern P_j, there is an alphabet symbol, say e_a, which is in D (else D would not be privacy preserving). This implies that set S_a contains element j, thereby proving that the collection of subsets corresponding to subsets in D forms a cover and the size of this cover is exactly $|D|$.

The above approximation preserving reduction shows that utility loss minimizing oblivious PP-CEP problem is at least as hard as set cover, thereby proving the stated claim. □

While the problem of oblivious PP-CEP is in general NP-hard, there are scenarios where certain parameters are small constants, in which cases the complexity of the problem can be different. In the following sections we carve out two special cases and show complexity results under alternative problem formulations.

3.2 A polynomially solvable special case

In this section we formulate the problem as an integer programming problem as follows. Let $x_i \in \{0,1\}$ be the decision variables to keep/drop events of type $e_i \in \Sigma$ in order to ensure privacy, with $x_i = 1$ being the decision of keeping event e_i, and $x_i = 0$ dropping event e_i. Further let $y_j \in \{0,1\}$ to be the variable to denote whether query Q_j can be reported. The set of queries that can be reported depends on decision variables, in other words y_js are completely dependent on x_is.

We determine the expected utility produced by query patterns using arrival rates of event types. Recall that λ_i is the arrival rate of event type e_i. Given a query Q_j and its associated time window $T(Q_j)$, we can first compute for each event type in Q_j the expected number of event occurrences in the time window $T(Q_j)$ using the event arrival rate. Then using the expected count of each event that participates Q_j we can compute the expected number of query matches for Q_j in the time window $T(Q_j)$. In this section we will just use $F(Q_j, \vec{\lambda})$ to denote the expected query matches for Q_j so computed, with $\vec{\lambda} \in \mathbb{R}_{+}^{|\Sigma|}$ being the vector with arrival rate λ_i as components. For details of the computation of $F(Q_j, \vec{\lambda})$ please see the explanation in Appendix A.

Using $F(Q_j, \vec{\lambda})$, the expected utility produced by query pattern Q_j over its associated time window $T(Q_j)$ can be written as

$$E(Q_j) = w(Q_j) \times F(Q_j, \vec{\lambda}) \tag{3}$$

The expected utility gain per unit time of Q_j, $\overline{E}(Q_j)$, can be

obtained by normalizing $E(Q_j)$

$$\overline{E}(Q_j) = \frac{E(Q_j)}{T(Q_j)} \tag{4}$$

Hence expected total utility gain per unit time, denoted as $G(X)$, where $X = \{x_i\}$ being the vector of event dropping decisions, can be expressed as

$$G(X) = \sum_{Q_j \in \mathcal{Q}} \overline{E}(Q_j) \cdot y_j \tag{5}$$

Similarly the utility loss function $L(X)$ can be defined as

$$L(X) = \sum_{Q_j \in \mathcal{Q}} \overline{E}(Q_j) \cdot (1 - y_j) \tag{6}$$

As we mentioned, here we use the expected value of query matches to quantify the utility for each query $E(Q_i)$. Using this simple technique to estimate utility allows us to explore the intrinsic difficulty in the structure of the problem unencumbered.

The oblivious PP-CEP can then be viewed as the problem of maximizing $G(X)$ (or, equivalently, minimizing $L(X)$) under the constraint that no private patterns are disclosed, which means that for each private pattern, at least one of the events that participate the private pattern should be dropped to ensure that no matches for the private patterns will ever be produced. Formally, for each $P_i \in \mathcal{P}$, the privacy-preserving constraints can be written as:

$$\sum_{p_j^i \in P_i} x_j < |P_i| \tag{7}$$

Furthermore, whether or not query pattern Q_i can be reported (y_i variables) can be expressed using the variables denoting whether events have been dropped (as represented by x_i) using the following linear constraints. For all $Q_i \in \mathcal{Q}$

$$y_i \leq \frac{1}{|Q_i|} \sum_{q_j^i \in Q_i} x_i \tag{8}$$

Intuitively, this is to say that Q_i can be reported ($y_i = 1$) if and only if all participating events are not dropped.

The problem of utility maximization $G(X)$ is then a integer linear programming problem subject to the constraints in Equation (7) and Equation (8). With this formulation, using Lenstra's algorithm [15], we have the following theorem:

THEOREM 5. *The oblivious PP-CEP problem is polynomially solvable if the total size of the alphabet and query patterns, $|\Sigma| + |\mathcal{Q}|$, or the total number of query and private patterns, $|\mathcal{P}| + |\mathcal{Q}|$ is some fixed constant.*

The proof of this Theorem follows directly from Lenstra's algorithm [15], which shows that the ILP problem is polynomially solvable if the number of variables, or the number of constraints is no more than a small constant. In our ILP formulation, we have a total of $|\Sigma|$ x variables, and $|\mathcal{Q}|$ y variables, with a total of $|\mathcal{P}|+|\mathcal{Q}|$ constraints. Given Lenstra's algorithm, the optimal utility gain (and similarly utility loss) is solvable if either of these two is no greater than a small constant, thus the Theorem 5.

3.3 An approximable special case

Alternatively, we could formulate the oblivious PP-CEP problem as as follows. Let $\overline{x}_i \in \{0, 1\}$ be the reverse of the decision variable x_i defined before. In other words, it is the decision variables to drop/keep events of type $e_i \in \Sigma$, with $\overline{x}_i = 1$ being the decision of dropping event e_i, and $\overline{x}_i = 0$ keeping event e_i (the reverse of the meaning of x_i). Instead of introducing variables y_i, we express the expected overall utility using $\overline{x}_i$ directly. Specifically, given the expected utility $\overline{E}(Q_i)$ produced for each query pattern Q_i per unit time in Equation 5, the expected total utility gain per unit of time can be expressed as

$$G'(X) = \sum_{Q_i \in \mathcal{Q}} \overline{E}(Q_i) \cdot r(Q_i) \tag{9}$$

where $r(Q_i)$ denotes whether Q_i can be reported based on the decisions of dropping events x_i: query Q_i can be reported only if no participating events are dropped.

$$r(Q_i) = \prod_{q_j^i \in Q_i} (1 - \overline{x}_i) \tag{10}$$

Similarly the utility loss function $L'(X)$ can be defined using $\overline{x}_i$ as

$$L'(X) = \sum_{Q_i \in \mathcal{Q}} \overline{E}(Q_i) \cdot (1 - r(Q_i)) \tag{11}$$

Like the previous ILP formulation, the privacy constraint in this formulation is that for each private pattern, at least one of the events that participate the private pattern should be dropped. For each $P_i \in \mathcal{P}$, we need to have:

$$\sum_{p_j^i \in P_i} \overline{x}_j \geq 1 \tag{12}$$

Note that instead of the packing constraints as specified in Equation (7) and Equation (8) for the ILP formulation, here with the use of variables $\overline{x}_i$ we instead get a set of covering constraints.

Now we have a new optimization problem with (non-linear) objective function in Equation (9) and Equation (11), and the linear constraints in Equation (12). While we cannot leverage the results established for ILPs, we observe that the new objective function have the nice property of being *sub-modular*.

Submodularity [9] is an important function property that has been studied in the theoretical computer science for its use in optimization algorithms. It can be formally defined as follows.

DEFINITION 12. *[9] Let V be a set of cardinality n. A real-valued functions f over subsets of X, $f : 2^V \rightarrow \mathbb{R}$ is submodular if it satisfies*

$$f(X + e) - f(X) \geq f(Y + e) - f(Y), \textit{for all } X \subset Y \subseteq V$$

The intuitive interpretation of submodularity is that as the input set grows, the incremental gain in f will decrease for the same addition in input (the e). This property is sometimes also known as *diminishing returns*.

LEMMA 1. *The utility loss function $L'(X)$ in Equation (11) is a submodular function.*

We can intuitively see why $L'(X)$ is submodular. The utility for a given query pattern Q_i will be lost the first time a constituent event gets dropped. Any additional events being dropped for the same query Q_i will not yield any increase in the utility loss $L'(X)$. In other words, the decisions of dropping one additional event given a larger set of events have been dropped will produce less increment in utility loss than that of case when the additional event is dropped given a smaller set of events have been dropped, a classical definition of diminishing returns.

Based on the submodular property, we can show a good approximation ratio for $L'(X)$ in the problem of oblivious PP-CEP processing in Theorem 6.

THEOREM 6. *Let l_p be the maximum length of any private pattern that is no greater than some small constant. The utility loss function $L'(X)$ in oblivious PP-CEP processing is l_p-approximable.*

The proof of Theorem 6 is built on the results in [11]. Specifically, observe that the integer program we have is l_p row sparse, with covering constraints (in Equation (12)), and a submodular objective function (Lemma 1). It has been shown in [11] that such an integer program is l_p approximable, thus our result in Theorem 6.

This approximability result for $L'(X)$, however, does not extend in a similar manner to the utility gain function $G'(X)$. Specifically, while the approximation ratio for $L'(X)$, l_p, is essentially the column sparsity of the coefficient matrix of the constraints, we cannot obtain a similar row sparsity approximation ratio for the utility gain function $G'(X)$. The reason is because for utility gain it becomes a *supermodular* function maximization with packing constraints, which is known to be hard to approximate. We in the following construct an algorithm and show that utility gain is $l_q d_q d_p$ approximable. This is almost tight as it is also $(l_q d_q d_p - 2)(1-\epsilon)$-inapproximable as described in Theorem 3.

THEOREM 7. *The maximization of the utility gain function $G'(X)$ in the oblivious PP-CEP is $l_q d_q d_p$-approximable.*

Proof Sketch. We show the $l_q d_q d_p$-approximation factor by constructing an algorithm that has no less than $\frac{1}{l_q d_q d_p}$ utility gain of the optimal utility gain. The algorithm works as follows. First we pre-process the query set by removing queries that can never be satisfied due to the presence of the private patterns (for example, those queries that are subsequences of some private pattern). We then sort all remaining feasible queries Q_i in $\mathcal{Q}$ by utility weight $W(Q_i)$. We pick in $\mathcal{Q}$ the query Q_{s_1} that has the largest weight. Let the set of events that participate Q_{s_1} be E_{s_1}. For each event $e_i \in E_{s_1}$, denote the set of private patterns that contain e_i as $\mathcal{P}_{e_i}$. Randomly select an event e_i^j in each pattern $P^j \in \mathcal{P}_{e_i}$ and suppress it. This would affect all queries that contain event e_i^j, denoted as $Q_{e_i^j}$, because these queries would never be reported due to the suppression of e_i^j. Removes all such affected queries $Q_{e_i^j}$ from $\mathcal{Q}$. In the remaining queries that are not affected, again find the Q_{s_2} with the highest utility weight and proceed as above. The claim is that this algorithm is $l_q d_q d_p$-approximate.

We get $l_q d_q d_p$ approximation ratio because each query Q_{s_k} we pick has at most l_q events. For each such event e_i there are no more than d_p private patterns that contain it. Since we suppress one event for each such private pattern, the total number of suppressed events cannot be more than $l_q d_p$. In addition, each suppressed event participates no more than d_q query patterns, that will affect at most $d_p d_q l_q$ queries. That means in each iteration, keeping the query with the highest weight entails removing at most $d_p d_q l_q$ queries, all of which with less utility weight than the one we keep. As a result, we can keep at least $\frac{1}{d_p d_q l_q}$ utility out of maximum possible utility (the utility sum of all queries), which has to be no less than the optimal oblivious PP-CEP utility, thus the approximation ratio in Theorem 7. □

3.4 A robustness analysis

The approximation analysis conducted in Section 3.2 and Section 3.3 are based on the assumption that events arrive exactly as expected as specified by the arrival rate λ_i of the Poisson process. In reality the actual arrival process can deviate from the expectations. In this section we give a robustness analysis to show the utility bound in such cases.

Recall that the overall utility gain is defined in Equation (2) as $\sum_{Q_i \in \mathcal{Q}} w(Q_i) \cdot \mathcal{C}(Q_i, S)$, where $\mathcal{C}(Q_i, S)$ denotes the number of matches for Q_i over the event sequence S. In our analysis we use the arrival rates λ_i for each type of event to compute the expected count of matches $\mathcal{C}_E(Q_i, S)$. Suppose the actual count of matches $\mathcal{C}_A(Q_i, S)$ deviates from the expected value by a ratio of l, or alternatively

$$\frac{1}{l} \leq \frac{\mathcal{C}_E(Q_i, S)}{\mathcal{C}_A(Q_i, S)} \leq l \tag{13}$$

Since we estimate the arrival rate λ_i by sampling of the arriving events which may not perfectly predict the arriving events in the future, we quantify the robustness of the approximation ratio given the inaccurate statistics in the following proposition.

PROPOSITION 1. *If algorithm $\mathcal{A}$ is a k-approximate algorithm for utility gain maximization (or utility loss minimization) PP-CEP, for any real event sequence whose true count of query matches deviates from the expected count by no more than a ratio of l, the approximation ratio of $\mathcal{A}$ must be better than $l^2 k$.*

This robust approximation ratio applies to previous results in Theorem 6 and Theorem 7. We show the approximation ratio $l^2 k$ for utility maximization problem and the ratio for utility loss minimization are similar. Let U_A be the actual utility using algorithm $\mathcal{A}$, and U_E be the expected utility using $\mathcal{A}$. Given the deviation ratio of at most l, we know that $U_A \cdot l \geq U_E$. We know that $\mathcal{A}$ has approximation ratio k, so the expected utility $U_E \cdot k \geq U_E^{OPT}$. Furthermore the actual optimal utility cannot be off from the expected optimal by a ratio of l or $U_E^{OPT} \cdot l \geq U_A^{OPT}$. Summarizing the three inequalities we obtain $U_A \cdot l^2 k \geq U_A^{OPT}$, meaning the utility of using $\mathcal{A}$ on the actual event sequence is at most off by $l^2 k$ from the optimal utility on the actual event sequence.

4. THE WINDOWED PP-CEP

In this section we explore a different variant of the problem, which we term the *windowed* version of PP-CEP. While the oblivious PP-CEP suppresses events in a global manner independent of events just arrived, the windowed version takes into account the events that have just arrived in the window Δ of size $|\Delta|$ to devise suppression decisions. Here Δ is a window over the event sequence, $|\Delta|$ is the number of events present in the window, and $T(\Delta)$ is its length in time.

DEFINITION 13. *A **windowed PP-CEP** is a PP-CEP, that suppresses events based on a suppression function $R : (\Sigma \cup \Phi)^{|\Delta|} \times \Sigma \rightarrow \{0, 1\}$, where the first input, $(\Sigma \cup \Phi)^{|\Delta|}$ represents the events that have arrived in the window Δ immediately prior to the current arriving event, and the second input, Σ, stands for the current arriving event e. A suppression decision has to be made for e, with 1 representing suppressing the current event e and 0 keeping it. Here Φ represents the absence of an event when there is less than $|\Delta|$ events in the window.*

In order to ensure privacy of the CEP system, the size of the window $|\Delta|$ has to be sufficiently large to accommodates all possible events that may arrive in $\max_{P_i \in \mathcal{P}}(T(P_i))$, for otherwise the suppression decisions devised may not be aware of the events that previously arrived and are outside of the window Δ. Such events coupled with the presence of certain events in the current window, can produce matches of private patterns and compromise privacy.

We use the following example to illustrate the windowed PP-CEP.

EXAMPLE 5. *We continue with Example 4, and consider the private pattern $P_1 = (B, C)$. Let the window size $|\Delta| = 2$.*

*Informally, use the shorthand notation $*$ to denote zero or more occurrences of any event. Among the space of all possible windowed PP-CEP, we consider two strategies, $R_B(*, B) = 1$ or intuitively the one that suppresses B; and $R_C(*B*, C) = 1$ or the one that suppresses C.*

Given R_B, and the original event sequence $S = (A_1, B_2, A_3, C_4, D_5, D_6, C_7, A_8, D_9, E_{10})$, at time t_2, B_2 will be suppressed to produce $S_B = (A_1, A_3, C_4, D_5, D_6, C_7, A_8, D_9, E_{10})$, over which the query set will be executed. Note that this particular strategy , $R_B(*, B) = 1$, suppresses B irrespective of the window Δ, and is equivalent to the oblivious PP-CEP that suppresses B obliviously. In this example it produces the same suppression result as the one in Example 4.

Now we consider the R_C suppression function. At t_4 the event C_4 will be suppressed because at that point, window Δ has the sequence (B_2, A_3), which leads to the suppression of C_4 as specified by R_C. In contrast, at t_7, given the window $\Delta = (D_5, D_6)$, C_7 will be preserved as R_C outputs 0. This is possible because at t_7, given the window Δ, it is clear that revealing C_7 will not compromise privacy due to the absence of B in Δ. This produces the suppressed event sequence $S_C = (A_1, B_2, A_3, D_5, D_6, C_7, A_8, D_9, E_{10})$. Note that this is different from the oblivious PP-CEP in Example 4 that suppresses all events of type C, which includes C_7 that does not have to be suppressed.

While it is apparent that it is necessary that any arriving event that can produce an match for some private pattern needs to be suppressed to ensure privacy, a utility maximizing suppression function R does not necessarily suppress the arriving event only when the arriving event can cause some private patterns to be matched (or equivalently, only suppressing the last event of each private pattern). As an intuitive example if there is one private pattern $P = (A, B, C)$, in which the event B rarely happens and has little contribution to the expected utility, while C arrives much more frequently and contributes to many query patterns, it may be a better choice to suppress B whenever it is seen and with A in the window Δ, instead of waiting until the last moment to suppress C. In general the whole space of all windowed PP-CEP suppression functions have to be considered.

The hardness results in Section 3.1 obtained for the oblivious PP-CEP, including the NP-hardness and the inapproximability results (Theorem 2, Theorem 3 and Theorem 4) still hold in the windowed PP-CEP. To see this, the set cover problem and the independence set problem can be reduced to the windowed PP-CEP in the same fashion as previously shown if the time windows of private patterns are set to infinite.

We consider the special case where the window size $|\Delta|$ is no more than some fixed constant (or the window Δ of size $|\Delta|$ can always accommodate all events that arrive within the time window $\max_{P_i \in \mathcal{P}}(T(P_i))$). This can in some sense be viewed as the special case where among the input parameters listed in Table 1, both $T(P_i)$ and λ_i are small. Essentially, this allows us to consider all possible subsequences that may appear in the window. In this special scenario we show an approximability result for utility loss minimization as follows.

THEOREM 8. *Let l_p, the maximum length of any private pattern, and $|\Delta|$, the window size, be fixed constant. Then there exists a polynomial time approximation algorithm that achieves a l_p-approximation factor for utility loss function in windowed PP-CEP processing.*

Proof Sketch. We sketch the proof of this theorem, which is similar to that of Theorem 6. First we enumerate all possible event combinations in the window of size $|\Delta|$. This is possible given that $|\Delta|$ is some fixed constant. Let p_i be the probability variable of each possible window Γ_i, and x_i^j be suppression variables given Γ_i and the arriving event of type e_j (which is essentially the suppression function $x_i^j = R(\Gamma_i, e_j)$). We can build a system of equations that represents the transition of the probabilities between possible windows Γ_i, using the suppression variables x_i^j and event arrival rate λ_k. That is, we set up a system of equations with x_i^j and p_i as unknowns for all windows Γ_i and event types e_j. Solving this system of equations allows us to represent the probabilities of seeing each window Γ_i using x_i^j and event arrival rate λ_k. Given the probabilities of each window as a function of x_i^j and λ_k, and the suppression variables x_i^j, we can write the expected utility loss $L(X)$ as a function of λ_k and x_i^j.

Similar to the argument made in Lemma 1, the utility loss function $L(X)$ in the windowed PP-CEP is also submodular due to the "diminishing gain" in utility loss when additional events of some window are suppressed.

The privacy constraints can also be expressed using a covering constrains of l_p row-sparsity. Specifically, for each query $P_i = (p_1^i, p_2^i, \ldots, p_n^i)$, let the event types of p_k^i be $e_{m_k}^i \in \Sigma$, privacy will be breached if the sequence, with m_n events of types $e_{m_k}^i$, for all such k in $[1, n]$, has none of the m_n events dropped. This can be expressed as a constraint, which is the sum of $R(\Gamma_{w_j}, e_j)$ can be no less than 1, where Γ_{w_j} is any possible window immediately prior to event of type e_j, or $\sum_{e_j \in P_i} R(\Gamma_{w_j}, e_j) \geq 1$, which is a covering constraint. Also observe that $R(\Gamma_{w_j}, e_j)$ are just the x variables used in the utility function. While there are many possible windows ($(|\Sigma| + 1)^{\Delta}$) prior to each event e_j, and consequently many such constraints, in each constraint the total number of variables are at most n, the length of the private patterns. Given that n is no more than l_p, the row sparsity of the constrain matrix cannot be greater than l_p.

It is thus a submodular minimization problem with covering constraints. Using the framework of Iwata et al. [11] we can obtain a factor l_p approximation for minimizing utility loss. □

5. CONCLUSIONS AND FUTURE WORK

There are a number of interesting ways in which this work can be further extended. First, in this work we restrict the CEP query language to be simple conjunctions of occurrences of events. Extending the complexity analysis to more expressive query languages is of practical and theoretical interest. Furthermore, in addition to the oblivious and the windowed PP-CEP, alternative strategies, like delaying the decision of dropping events for a delay window instead of making real-time decision, may offer opportunities to further enhance utility. Understanding the complexity of such a quasi-real-time approach and the utility/responsiveness trade-offs would be interesting. In addition, in this work the event arrival process is assumed to be an independent Poisson process. Modeling the dependence of the arriving events using models like Markov chains is also an interesting direction for future research.

6. REFERENCES

[1] Coral8: www.coral8.com.

[2] Streambase: www.streambase.com.

[3] Streaminsight: http://www.microsoft.com/sqlserver/2008/en/us/r2-complex-event.aspx.

[4] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In *SIGMOD*, 2008.

[5] M. H. Ali and C. G. et al. Microsoft cep server and online behavioral targeting. In *VLDB*, 2009.

[6] N. Bansal, N. Korula, V. Nagarajan, and A. Srinivasan. On k-column sparse packing programs. *CoRR*, 0908.2256, 2009.

[7] R. S. Barga, J. Goldstein, M. Ali, and M. Hong. Consistent streaming through time: A vision for event stream processing. 2007.

[8] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45 (4), 1998.

[9] S. Fujishige. *Submodular functions and optimization*. 2005.

[10] D. Gyllstrom, E. Wu, H.-J. Chae, Y. Diao, P. Stahlberg, and G. Anderson. SASE: Complex event processing over streams. In *CIDR*, 2007.

[11] S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In *FOCS*, 2009.

[12] P. Kall and S. W. Wallace. *Stochastic Programming*. Wiley, 1994.

[13] C. Koufogiannakis and N. E. Young. Greedy Δ-approximation algorithm for covering with arbitrary constraints and submodular cost. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, pages 634–652, Berlin, Heidelberg, 2009. Springer-Verlag.

[14] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance*. Prentice Hall, 1984.

[15] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operation Research*, 1983.

[16] M. Liu, M. Li, D. Golovnya, E. A. Rundensteiner, and K. Claypool. Sequence pattern query processing over out-of-order event streams. In *ICDE*, 2009.

[17] Y. Mei and S. Madden. Zstream: A cost-based query processor for adaptively detecting composite events. In *SIGMOD*, 2009.

[18] D. Pritchard and D. Chakrabarty. Approximability of sparse integer programs. *CoRR*, 0904.0859, 2009.

[19] L. Trevisan. Inapproximability of combinatorial optimization problems. Technical report, University of California Berkeley, 2004.

[20] D. Wang, E. Rundensteiner, and R. Ellison. Active complex event processing for realtime health care. In *VLDB*, 2010.

[21] W. White, M. Riedewald, J. Gehrke, and A. Demers. What is "next" in event processing? In *PODS*, 2007.

[22] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *SIGMOD*, 2006.

APPENDIX

A. EXPECTED QUERY UTILITY

In this section we describe how the expected query matches for Q_j over the time window $T(Q_j)$, denoted as $F(Q_j, \vec{\lambda})$, can be estimated using arrival rates of event types, where λ_i is the arrival rate of event type e_i.

We first compute for each event type in Q_j the expected number of event occurrences in the time window $T(Q_j)$. Specifically, for each event type e_i, the expected number of e_i occurrences in $T(Q_j)$, denoted as l_i, can be computed as $l_i = \lambda_i T(Q_j)$. Further denote $\sigma(Q_j)$ as the set of the types of the events that are part of Q_j, $|Q_j|$ as the total number of events in Q_j, and let $n(e_i)$ be the number of occurrences of event type e_i in Q_j (for example, a query $Q = (A, A, B)$ would have $\sigma(Q) = \{A, B\}$, $n(A) = 2$, $n(B) = 1$, and $|Q| = 3$). Denote $L = \sum_{e_i \in \sigma(Q_j)} l_i$ be the total number of occurrences of events relevant to Q_j in time window $T(Q_j)$.

We can then estimate the expected number of query matches for Q_j in $T(Q_j)$ as

$$F(Q_j, \lambda) = \binom{L}{|Q_j|} \frac{\prod_{e_i \in \sigma(Q_j)} \prod_{k=0}^{n(e_i)} (l_i - k)}{\prod_{r=0}^{|Q_j|} (L - r)} \tag{14}$$

The reasoning of $F(Q_j, \lambda)$ works as follows. Given a total of L event occurrences in $T(Q_j)$, we only pick a total $|Q_j|$ events to form a query match. Let us pick the first $|Q_j|$ events to form random permutations and compute the probability that the first $|Q_j|$ events produces a match. Let the first position of Q_j be an event of type e_{p_1}. The probability of actually seeing the event that type is $\frac{l_{e_{p1}}}{L}$. For the second event in Q_j, if it is also of type e_{p_1}, the probability of seeing that event that type is $\frac{l_{e_{p_1}}-1}{L-1}$, otherwise it is $\frac{l_{e_{p1}}}{L-1}$, so on and so forth. In the end this gives us the term $\frac{\prod_{e_i \in \sigma(Q_j)} \prod_{k=0}^{n(e_i)} (l_i - k)}{\prod_{r=0}^{|Q_j|} (L-r)}$. Given that there are a total of $\binom{L}{|Q_j|}$ such possible positions out of L event occurrences and each of which is symmetric, the expected count of query matches can be expressed as the product of the two, thus the Equation (14).