On Load Shedding in Complex Event Processing

Yeye He
Microsoft Research
Redmond, WA, 98052
yeyehe@microsoft.com

Siddharth Barman
California Institute of Technology
Pasadena, CA, 91106
barman@caltech.edu

Jeffrey F. Naughton
University of Wisconsin-Madison
Madison, WI, 53705
naughton@cs.wisc.edu

ABSTRACT

Complex Event Processing (CEP) is a stream processing model that focuses on detecting event patterns in continuous event streams. While the CEP model has gained popularity in the research communities and commercial technologies, the problem of gracefully degrading performance under heavy load in the presence of resource constraints, or load shedding, has been largely overlooked. CEP is similar to "classical" stream data management, but addresses a substantially different class of queries. This unfortunately renders the load shedding algorithms developed for stream data processing inapplicable. In this paper we study CEP load shedding under various resource constraints. We formalize broad classes of CEP load-shedding scenarios as different optimization problems. We demonstrate an array of complexity results that reveal the hardness of these problems and construct shedding algorithms with performance guarantees. Our results shed some light on the difficulty of developing load-shedding algorithms that maximize utility.

1. INTRODUCTION

The complex event processing or CEP model has received significant attention from the research community [6, 28, 36, 37, 50, 52], and has been adopted by a number of commercial systems including Microsoft StreamInsight [1], Sybase Aleri [3], and StreamBase [2]. A wide range of applications, including inventory management [4], behavior monitoring [47], financial trading [5], and fraud detection [8, 51], are now powered by CEP technologies.

A reader who is familiar with the extensive stream data processing literature may wonder if there is anything new here, or if CEP is just another name for stream data management. While both kinds of system evaluate queries over data streams, the important difference is the class of queries upon which each system focuses. In the traditional stream processing literature, the focus is almost exclusively on aggregate queries or binary equi-joins. By contrast, in CEP, the focus is on detecting certain patterns, which can be viewed as multi-relational non-equi-joins on the time dimension, possibly with temporal ordering constraints. The class of queries addressed by CEP systems requires different evaluation algorithms and different load-shedding algorithms than the class previously considered in the context of stream data management.

As an example of a CEP-powered system, consider the health care monitoring system, HyReminder [47], currently deployed at the University of Massachusetts Memorial Hospital. The HyReminder system tracks and monitors the hygiene compliance of healthcare workers. In this hospital setting, each doctor wears an RFID badge that can be read by sensors installed throughout the hospital. As doctors walk around the hospital, the RFID badges they wear trigger "event" data, which is transmitted to a central CEP engine. The CEP engine in turn looks for patterns to check for hygiene compliance. As one example, according to the US Center for Disease Control (CDC) [13], a doctor who enters or exits a patient room (which is captured by sensors installed in the doorway and encoded as an `Enter-Patient-Room` event or `Exit-Patient-Room` event) should cleanse her hands (encoded by a `Sanitize` event) within a short period of time. This hygiene regulation can be tracked and enforced using the following CEP queries.

```
Q1:  SEQ(Sanitize, Enter-Patient-Room) within 1 min
Q2:  SEQ(Exit-Patient-Room, Sanitize) within 1 min
```

In the HyReminder system, these two CEP queries monitor the event sequence to track sanitization behavior and ensure hygiene compliance. As another example consider CIMS [4], which is a system also powered by CEP and deployed in the same University of Massachusetts hospital. CIMS is used for inventory management and asset tracking purposes. It captures RFID events triggered by tags attached to medical equipment and expensive medicines, and uses CEP to track supply usage and reduce inventory cost [15].

While the emergence of CEP model has spawned a wide variety of applications, so far research efforts have focused almost exclusively on improving CEP query join efficiency [6, 28, 36, 37, 50, 52]. *Load shedding*, an important issue that has been extensively studied in traditional stream processing [10, 20, 25, 26, 43, 44, 48, 49, 53], has been largely overlooked in the new context of CEP.

Like other stream systems, CEP systems often face bursty input data. Since over-provisioning the system to the point where it can handle any such burst may be uneconomical or impossible, during peak loads a CEP system may need to "shed" portions of the load. The key technical challenge herein is to selectively shed work so as to eliminate the less important query results, thereby preserve the more useful query results as defined by some *utility function*.

More specifically, the problem we consider is the following. Consider a CEP system that has a number of pattern queries, each of which consists of a number of events and is associated with a *utility function*. During peak loads, memory and/or CPU resources may not be sufficient. A utility-maximizing load shedding scheme should then determine which events should be preserved in memory and which query results should be processed by the CPU, so that not only are resource constraints respected, but also the overall utility of the query results generated is maximized.

	Memory-bound	CPU-bound	Dual-bound
Integral	IMLS	ICLS	IDLS
Fractional	FMLS	FCLS	FDLS

Table 1: Problem variants for CEP load shedding

We note that, in addition to utility-maximization in a single CEP application, the load-shedding framework may also be relevant to cloud operators that need to optimize across multiple CEP applications. Specifically, stream processing applications are gradually shifting to the cloud [34]. In cloud scenarios, the cloud operator in general cannot afford to provision for the aggregate peak load across all tenants, which would defeat the purpose of consolidation. Consequently, when the load exceeds capacity, cloud operators are forced to shed load. They have a financial incentive to judiciously shed work from queries that are associated with a low penalty cost as specified in Service Level Agreements (SLAs), so that their profits can be maximized (similar problems have been called "profit maximization in a cloud" and have been considered in the Database-as-a-Service literature [18, 19]). Note that this problem is naturally a utility-maximizing CEP load shedding problem, where *utility* essentially becomes the financial rewards and penalties specified in SLAs.

While load shedding has been extensively studied in the context of general stream processing [10, 20, 25, 26, 43, 44, 48, 49, 53], the focus there is aggregate queries or two-relation equi-join queries, which are important for traditional stream joins. The emerging CEP model, however, demands multi-relational joins that predominantly use non-equi-join predicates on timestamp. As we will discuss in more detail in Section 4, the CEP load shedding problem is significantly different and considerably harder than the problems previously studied in the context of general stream load shedding.

We show in this work that variants of the utility maximizing load shedding problems can be abstracted as different optimization problems. For example, depending on which resource is constrained, we can have three problem variants, namely *CPU-bound load shedding*, *memory-bound load shedding*, and *dual-bound load shedding* (with both CPU- and memory-bound). In addition we can have *integral load shedding*, where event instances of each type are either all preserved in memory or all shedded; and *fractional load shedding*, where a sampling operator exists such that a fraction of event instances of each type can be sampled according to a predetermined sampling ratio. Table 1 summarizes the six variants of CEP load shedding studied in this paper: IMLS (integral memory-bound load shedding), FMLS (fractional memory-bound load shedding), ICLS (integral CPU-bound load shedding), FCLS (fractional CPU-bound load shedding), IDLS (integral dual-bound load shedding), and FDLS (fractional dual-bound load shedding).

We analyze the hardness of these six variants, and study efficient algorithms with performance guarantees. We demonstrate an array of complexity results. In particular, we show that CPU-bound load shedding is the easiest to solve: FCLS is solvable in polynomial time, while ICLS admits a FPTAS approximation. For memory-bound problems, we show that IMLS is in general NP-hard and hard to approximate. We then identify two special cases in which IMLS can be efficiently approximated or even solved exactly, and describe a general rounding algorithm that achieves a bi-criteria approximation. As for the fractional FMLS, we show it is hard to solve in general, but has approximable special cases. Finally, for dual-bound load shedding IDLS and FDLS, we show that they generalize memory-bound problems, and the hardness results from memory-bound problems naturally hold. On the positive side, we describe a tri-criteria approximation algorithm and an approximable special case for the IDLS problem.

The rest of the paper is organized as follows. We first describe necessary background of CEP in Section 2, and introduce the load shedding problem in Section 3. We describe related work in Section 4. In Section 5, Section 6 and Section 7 we discuss the memory-bound, CPU-bound and dual-bound load-shedding problems, respectively. We conclude in Section 8.

2. BACKGROUND: COMPLEX EVENT PROCESSING

The CEP model has been proposed and developed by a number of seminal papers (see [6] and [52] as examples). To make this paper self-contained, we briefly describe the CEP model and its query language in this section.

2.1 The data model

Let the domain of possible event types be the alphabet of fixed size $\Sigma = \{E_i\}$, where E_i represents a type of event. The event stream is modeled as an event sequence as follows.

DEFINITION 1. *An **event sequence** is a sequence $S = (e_1, e_2, ..., e_N)$, where each **event instance** e_j belongs to an **event type** $E_i \in \Sigma$, and has a unique time stamp t_j. The sequence S is temporally ordered, that is $t_j < t_k$, $\forall j < k$.*

EXAMPLE 1. *Suppose there are five event types denoted by uppercase characters $\Sigma = \{A, B, C, D, E\}$. Each character represents a certain type of event. For example, for the hospital hygiene monitoring application HyReminder, event type A represents all event instances where doctors enter ICU, B denotes the type of events where doctors washing hands at sanitization desks, etc.*

A possible event sequence is $S = (A_1, B_2, C_3, D_4, E_5, A_6, B_7, C_8, D_9, E_{10})$, where each character is an instance of the corresponding event type occurring at time stamp given by the subscript. So A_1 denotes that at time-stamp 1, a doctor enters ICU. B_2 shows that at time 2, the doctor sanitizes his hands. At time-stamp 6, there is another enter-ICU event A_6, so on and so forth.

Following the standard practice of the CEP literature [6, 7, 37, 52], we assume events are temporally ordered by their timestamps. Out-of-order events can be handled using techniques from [16, 36].

DEFINITION 2. *Given an event sequence $S = (e_1, e_2, ..., e_N)$, a sequence $S' = (e_{i_1}, e_{i_2}, ..., e_{i_m})$ is a **subsequence** of S, if $1 \leq i_1 < i_2 ... < i_m \leq N$.*

Note that the temporal ordering in the original sequence is preserved in subsequences, and a subsequence does not have to be a contiguous subpart of a sequence.

2.2 The query model

Unlike relational databases, where queries are typically ad-hoc and constructed by users at query time, CEP systems are more like other stream processing systems, where queries are submitted ahead of time and run for an extended period of time (thus are also known as *long-standing queries*). The fact that CEP queries are known a priori is a key property that allows queries to be analyzed and optimized for problems like utility maximizing load shedding.

Denote by $\mathcal{Q} = \{Q_i\}$ the set of CEP queries, where each query Q_i is a sequence query defined as follows.

DEFINITION 3. *A CEP **sequence query** Q is of the form $Q = SEQ(q_1, q_2, ...q_n)$, where $q_k \in \Sigma$ are event types. Each query Q is associated with a time based sliding window of size $T(Q) \in \mathbb{R}^+$, over which Q will be evaluated.*

We then define the skip-till-any-match query match semantics.

DEFINITION 4. *In skip-till-any-match, a subsequence $S' = (e_{i_1}, e_{i_2}, \ldots, e_{i_n})$ of S is considered a **query match** of $Q = (q_1, q_2, \ldots, q_n)$ over time window $T(Q)$ if:*

(1) Pattern matches: Event e_{i_l} in S' is of type q_l for all $l \in [1, n]$,

(2) Within window: $t_{i_n} - t_{i_1} \leq T(Q)$.

We illustrate query matches using Example 2.

EXAMPLE 2. *We continue with Example 1, where the event sequence* $S = (A_1, B_2, C_3, D_4, E_5, A_6, B_7, C_8, D_9, E_{10})$. *Suppose there are a total of three queries:* $Q_1 = SEQ(A, C)$, $Q_2 = SEQ(C, E)$, $Q_3 = SEQ(A, B, C, D)$, *all having the same window size* $T(Q_1) = T(Q_2) = T(Q_3) = 5$.

Both sequences (A_1, C_3) *and* (A_6, C_8) *are matches for* Q_1, *because they match patterns specified in* Q_1, *and are within the time window 5. However,* (A_1, C_8) *is not a match even though it matches the pattern in* Q_1, *because the time difference between* C_8 *and* A_1 *exceeds the window limit 5.*

Similarly, (C_3, E_5) *and* (C_8, E_{10}) *are matches of* Q_2; (A_1, B_2, C_3, D_4) *and* (A_6, B_7, C_8, D_9) *are matches of* Q_3.

A query with the skip-till-any-match semantics essentially looks for the conjunction of occurrences of event types in a specified order within a time window. Observe that in skip-till-any-match a subsequence does not have to be contiguous in the original sequence to be considered as a match (thus the word *skip* in its name). Such queries are widely studied [2, 6, 28, 36, 37, 50, 52] and used in CEP systems.

We note that there are three additional join semantics defined in [6], namely, skip-till-next-match, partition-contiguity and contiguity. The load shedding problem formulated in this work is agnostic of the join semantics used. In the interest of space, more details of other join semantics and their relationship with load shedding can be found in the full version of the paper [31].

We also observe that although there are CEP language extensions like negation [28] and Kleene closure [23], in this work we only focus on the core language constructs that use conjunctive positive event occurrences. We leave such query extensions for load shedding as future work.

3. CEP LOAD SHEDDING

It is well known that continuously arriving stream data is often bursty [20, 43, 49]. During times of peak loads not all data items can be processed in a timely manner under resource constraints. As a result, part of the input load may have to be discarded (shedded), resulting in the system retaining only a subset of data items. The question that naturally arises is which queries should be preserved while others shedded? To answer this question, we introduce the notion of utility to quantify the "usefulness" of different queries.

3.1 A definition of utility

In CEP systems, different queries naturally have different real-world importance, where some query output may be more important than others. For example, in the inventory management application [4, 15], it is naturally more important to produce real-time query results that track expensive medicine/equipment than less expensive ones. Similarly, in the hospital hygiene compliance application [47], it is more important to produce real-time query results reporting serious hygiene violations with grave health consequences than the ones merely reporting routine compliance.

We define *utility weight* for each query to measure its importance.

DEFINITION 5. *Let* $\mathcal{Q} = \{Q_i\}$ *be the set of queries. Define the* ***utility weight*** *of query* Q_i, *denoted by* $w_i \in \mathbb{R}^+$, *as the perceived usefulness of reporting one instance of match of* Q_i.

A user or an administrator familiar with the application can typically determine utility weights. Alternatively, in a cloud environment, operators of multi-tenancy clouds may resort to service-level-agreements (SLAs) to determine utility weights. In this work we simply treat utility weights as known constants. We note that the notion of query-level weights has been used in other parts of data management literature (e.g., query scheduling [39]).

The total utility of a system is then defined as follows.

DEFINITION 6. *Let* $\mathcal{C}(Q_i, S)$ *be the number of distinct matches for query* Q_i *in* S. *The* ***utility*** *generated for query* Q_i *is*

$$U(Q_i, S) = w_i \cdot \mathcal{C}(Q_i, S) \tag{1}$$

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

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

Our definition of utility generalizes previous metrics like the *max-subset* [20] used in traditional stream load shedding literature. *Max-subset* aims to maximize the number of output tuples, and thus can be viewed as a special case of our definition in which each query has unit-weight.

We also note that although it is natural to define utility as a linear function of query matches for many CEP applications (e.g., [4, 47]), there may exist applications where utility can be best defined differently (e.g. a submodular function to model diminishing returns). Considering alternative utility functions for CEP load shedding is an area for future work.

EXAMPLE 3. *We continue with Example 2 using the event sequence* S *and queries* Q_1, Q_2 *and* Q_3 *to illustrate utility.*

Suppose the utility weight w_1 *for* Q_1 *is 1,* w_2 *is 2, and* w_3 *is 3. Since there are 2 matches of* Q_1, Q_2 *and* Q_3, *respectively, in* S, *the total utility is* $2 \times 1 + 2 \times 2 + 2 \times 3 = 12$.

3.2 Types of resource constraints

In this work, we study constraints on two common types of computing resources: CPU and memory.

Memory-bound load shedding. In this first scenario, memory is the limiting resource. Normally, arriving event data are kept in main memory for windowed joins until time-stamp expiry (i.e., when they are out of active windows). During peak loads, however, event arrival rates may be so high that the amount of memory needed to store all event data might exceed the available capacity. In such a case not every arriving event can be held in memory for join processing and some events may have to be discarded.

EXAMPLE 4. *In our running example the event sequence* $S = (A_1, B_2, C_3, D_4, E_5, A_6, B_7, C_8, D_9, E_{10}$ ...), *with queries* $Q_1 = SEQ(A, C)$, $Q_2 = SEQ(C, E)$ *and* $Q_3 = SEQ(A, B, C, D)$. *Because the sliding window of each query is 5, we know each event needs to be kept in memory for 5 units of time. Given that one event arrives in each time unit, a total of 5 events need to be simultaneously kept in memory.*

Suppose a memory-constrained system only has memory capacity for 3 events. In this case "shedding" all events of type B and D will sacrifice the results of Q_3 *but preserves A, C and E and meets the memory constraint. In addition, results for* Q_1 *and* Q_2 *can be produced using available events in memory, which amounts to a total utility of* $2 \times 1 + 2 \times 2 = 6$. *This maximizes utility, for shedding any other two event types yields lower utility.*

CPU-bound load shedding. In the second scenario, memory may be abundant, but CPU becomes the bottleneck. As a result, again only a subset of query results can be processed.

EXAMPLE 5. *We revisit Example 4, but now suppose we have a CPU constrained system. Assume for simplicity that producing each query match costs 1 unit of CPU. Suppose there are 2 unit of CPU available per 5 units of time, so only 2 matches can be produced every 5 time units.*

In this setup, producing results for Q_2 and Q_3 while shedding others yields a utility of $2 \times 2 + 2 \times 3 = 10$ given the events in S. This is utility maximizing because Q_2 and Q_3 have the highest utility weights.

Dual-bound load shedding. Suppose now the system is both CPU bound and memory bound (dual-bound).

EXAMPLE 6. *We continue with Example 5. Suppose now due to memory constraints 3 events can be kept in memory per 5 time units, and in addition 2 query matches can be produced every 5 time units due to CPU constraints.*

As can be verified, the optimal decision is to keep events A, C and E while producing results for Q_1 and Q_2, which yields a utility of $2 \times 1 + 2 \times 2 = 6$ given the events in S. Note that results for Q_3 cannot be produced because it needs four events in memory while only three can fit in memory simultaneously.

3.3 Types of shedding mechanisms

In this paper, we consider two types of shedding mechanisms, an *integral load shedding*, in which certain *types* of events or query matches are discarded altogether; and a *fractional load shedding*, in which a uniform sampling is used, such that a portion of event types or query matches is randomly selected and preserved.

Note that both the above mentioned load-shedding mechanisms are relevant in an *online* setting. That is, settings in which a shedding decision is made for the current event before the next arriving event is processed. This is in contrast to offline load shedding, where decisions are made after the whole event sequence has arrived. The reason we only focus on online load shedding is practicality – most stream applications demand real-time responsiveness; an offline algorithm that works after the entire event sequence has arrived is unlikely to be practically useful.

Performance of online algorithms is oftentimes measured against their offline counterparts to develop quality guarantees like *competitive ratios* [12]. However, we show in the following that meaningful competitive ratios cannot be derived for *any* online CEP load shedding algorithms.

PROPOSITION 1. *No online CEP load shedding algorithm, deterministic or randomized, can have competitive ratio better than $\Omega(n)$, where n is the length of the event sequence.*

A proof of this proposition can be found in Appendix A. Intuitively, to see why it is hard to bound the competitive ratio, consider the following adversarial scenario. Suppose we have a universe of $3m$ event types, $\Sigma = \{E_i\} \cup \{E_i'\} \cup \{E_i''\}$, $i \in [m]$. Let there be $2m$ queries $SEQ(E_i, E_i'')$ and $SEQ(E_i', E_i'')$, $\forall i \in [m]$, each with unit utility weight. The stream is known to be $(e_1, e_2, ..., e_m, X)$, where e_i is either of type E_i or E_i' with equal probability. In addition, X is drawn from the uniform distribution on $\{E_i'' : i \in [m]\}$. Lastly, suppose the system only has enough memory to hold two events. The optimal offline algorithm can look at the type of event X, denoted by E_k'', and keep the corresponding event e_k (of type E_k or E_k') that arrived previously, to produce a match (either (E_k, E_k'') or (E_k', E_k''), as the case may be) of utility of 1. In comparison, an online algorithm needs to select one event into memory before the event type of X is revealed. (Note that the offline algorithm cannot just output results based on the last event X given the form of the input, because e_k could be either E_k or E_k'.) Thus, the probability of producing a match is $\frac{1}{m}$, and the expected utility is also $\frac{1}{m}$.

This result essentially suggests that we cannot hope to devise online algorithms with good competitive ratios. In light of this result, in what follows, we will use *expected* number of query matches based on a characterization of the arriving event stream, and focus on optimizing the *expected utility* of online algorithms without further discussing competitive ratio bounds.

Σ	The set of all possible event types
E_j	Event of type j
λ_j	The number of arrived events E_j in a unit time (event arrival rate)
m_j	The memory cost of keeping each event of type E_j
$\mathcal{Q}$	The set of query patterns
Q_i	Query pattern i
$\|Q_i\|$	The number of event types in Q_i
w_i	The utility weight associated with Q_i
n_i	The number of matches of Q_i in a unit time
c_i	The CPU cost of producing each result for Q_i
C	The total CPU budget
M	The total memory budget
x_j	The binary selection decision of event E_j
$\overline{x}_j$	The fractional sampling decision of event E_j
y_i	The selection decision of query Q_i
$\overline{y}_i$	The fractional sampling decision of query Q_i
p	The max number of queries that one event type participates in
f	The fraction of memory budget that the largest query consumes
d	The maximum number of event types in any one query

Table 2: Summary of the symbols used

3.4 Modeling CEP systems

At a high level, the decision of which event or query to shed should depend on a number of factors, including utility weights, memory/CPU costs, and event arrival rates. Intuitively, the more important a query is, the more desirable it is to keep constituent events in memory and produce results of this query. Similarly the higher the cost is to keep an event or to produce a query match, the less desirable it is to keep that event or produce that result. The rate at which events arrive is also important, as it determines CPU/memory costs of a query as well as utility it can produce.

In order to study these trade-offs in a principled way, we consider the following factors in a CEP system. First, we assume that the utility weight, w_i, which measures the importance of query Q_i installed in a CEP system, is provided as a constant. We also assume that the CPU cost of producing each result of Q_i is a known constant c_i, and the memory cost of storing each event instance of type E_j is also a fixed constant m_j. Note that we do not assume uniform memory/CPU costs across different events/queries, because in practice event tuples can be of different sizes. Furthermore, the arrival rate of each event type E_j, denoted by λ_j, is assumed to be known. This is typically obtained by sampling the arriving stream [17, 41]. Note that characteristics of the underlying stream may change periodically, so the sampling procedure may be invoked at regular intervals to obtain an up-to-date estimate of event arrival rates.

Furthermore, we assume that the "expected" number of matches of Q_i over a unit time, denoted by n_i, can also be estimated. A simple but inefficient way to estimate n_i is to sample the arriving event stream and count the number of matches of Q_i in a fixed time period. Less expensive alternatives also exist. For example, assuming an independent Poisson arrival process, which is a standard assumption in the performance modeling literature [35], the number of query matches can be analytically estimated. The full version of the paper [31] discusses one possible approach to estimate the number of query matches using event arrival rates. In this work we will simply treat n_i as known constants without further studying the orthogonal issue of estimating n_i.

Lastly, note that since n_i here is the expected number of query matches, the utility we maximize is also optimized in an expected sense. In particular, it is not optimized under arbitrary arrival event

Approximation Ratio	
IMLS	$p/(1-f)$ [Theorem 4]
IMLS$^{\text{m}}$ (loss minimization)	$(\frac{1}{\tau}, \frac{1}{1-\tau})$ bi-criteria approximation, for any $\tau \in (0,1)$ [Theorem 3]
ICLS	$1+\epsilon$, for $\epsilon > 0$ [Theorem 10]
IDLS	$p/(1-f)$ [Theorem 12]
IDLS$^{\text{m}}$ (loss minimization)	$(\frac{1}{\tau}, \frac{1}{1-\tau}, \frac{1}{1-\tau})$ tri-criteria approximation, for any $\tau \in (0,1)$ [Theorem 11]
Relative Approximation Ratio (see Definition 8)	
FMLS	$1 - O\left(\lvert\Sigma\rvert^{-\frac{d-2}{2}}(t^2+1)^{-\frac{d}{2}}\right)$ where $t = \min\left\{\min_{E_j}\{\frac{\lambda_j m_j}{M}\}, \frac{1}{\sqrt{\lvert\Sigma\rvert}}\right\}$ [Theorem 7]
FMLS (under some assumptions)	$O\left(1 - \frac{k!}{(k-d)!k^d}\right)$ where $k > d$ [Theorem 8]
Absolute Approximation Ratio (see Definition 7)	
FMLS	$(1-\beta(\frac{k!}{(k-d)!k^d}))$-approximation, where $\beta = \min\left(\min_j\left\{\frac{\lambda_j m_j}{M}\right\}, 1\right)^d$, and $k > d$ controls approximation accuracy [Theorem 9].

Table 3: Summary of approximation results

strings (e.g., an adversarial input). While considering load shedding in such settings is interesting, Proposition 1 already shows that we cannot hope to get any meaningful bounds against certain adversarial inputs.

The symbols used in this paper are summarized in Table 2, and our main approximation results are listed in Table 3.

4. RELATED WORK

Load shedding has been recognized as an important problem, and a large body of work in the stream processing literature (e.g., [9, 10, 20, 26, 33, 43, 44, 48, 49]) has been devoted to this problem. However, existing work in the context of traditional stream processing predominantly considers the equi-join of two streaming relations. This is not directly applicable to CEP joins, where each join operator typically involves multi-relational non-equi-join (on time-stamps). For example, the authors in [33] are among the first to study load shedding for equi-joins operators. They proposed strategies to allocate memory and CPU resources to two joining relations based on arrival rates, so that the number of output tuples produced can be maximized. Similarly, the work [20] also studies the problem of load shedding while maximizing the number of output tuples. It utilizes value distribution of the join columns from the two relations to produce optimized shedding decisions for tuples with different join attribute values.

However, the canonical two-relation equi-join studied in traditional stream systems is only a special case of the multi-relational, non-equi-join that dominates CEP systems. In particular, if we view all tuples from R (resp. S) that have the same join-attribute value v_i as a virtual CEP event type R_i (resp. S_i), then the traditional stream load shedding problem is captured as a very special case of CEP load shedding we consider, where each "query" has exactly two "event types" (R_i and S_i), and there are no overlapping "event types" between "queries". Because of this equi-join nature, shedding one event has limited ramification and is intuitively easy to solve (in fact, it is shown to be solvable in [20]). In CEP queries, however, each event type can join with an arbitrary number of other events, and different queries use overlapping events. This significantly complicates the optimization problem and makes CEP load shedding hard.

In [10], sampling mechanisms are proposed to implement load shedding for aggregate stream queries (e.g., SUM), where the key technical challenge is to determine, in a given operator tree, where to place sampling operators and what sampling rates to use, so that query accuracy can be maximized. The work [44] studies the similar problem of strategically placing drop operator in the operator tree to optimize utility as defined by QoS graphs. The authors in [43] also consider load shedding by random sampling, and propose techniques to allocate memory among multiple operators.

The works described above study load shedding in traditional stream systems. The growing popularity of the new CEP model that focuses on multi-relational non-equi-join calls for another careful look at the load-shedding problem in the new context of CEP.

5. MEMORY-BOUND LOAD SHEDDING

Recall that in the memory-bound load shedding, we are given a fixed memory budget M, which may be insufficient to hold all data items in memory. The problem is to select a subset of events to keep in memory, such that the overall utility can be maximized.

5.1 The integral variant (IMLS)

In the integral variant of the memory-bound load shedding problem, a binary decision, denoted by x_j, is made for each event type E_j, such that event instances of type E_j are either all selected and kept in memory ($x_j = 1$), or all discarded ($x_j = 0$). The event selection decisions in turn determine whether query Q_i can be selected (denoted by y_i), because output of Q_i can be produced only if all constituent event types are selected in memory. We formulate the resulting problem as an optimization problem as follows.

$$\text{(IMLS)} \qquad \max \sum_{Q_i \in \mathcal{Q}} n_i w_i y_i \tag{3}$$

$$\text{s.t.} \sum_{E_j \in \Sigma} \lambda_j m_j x_j \leq M \tag{4}$$

$$y_i = \prod_{E_j \in Q_i} x_j \tag{5}$$

$$y_i, x_j \in \{0,1\} \tag{6}$$

The objective function in Equation (3) says that if each query Q_i is selected ($y_i = 1$), then it yields an expected utility of $n_i w_i$ (recall that as discussed in Section 3, n_i models the expected number of query matches of Q_i in a unit time, while w_i is the utility weight of each query match). Equation (4) specifies the memory constraint. Since selecting event type E_j into memory consumes $\lambda_j m_j$ memory, where λ_j is the arrival rate of E_j and m_j is the memory cost of each event instance of E_j, Equation (4) guarantees that total memory consumption does not exceed the memory budget M. Equation (5) ensures that Q_i can be produced if and only if all participating events E_j are selected and preserved in memory ($x_j = 1$, for all $E_j \in Q_i$).

5.1.1 A general complexity analysis

We first give a general complexity analysis. We show that this shedding problem is NP-hard and hard to approximate by a reduction from the *Densest k-Sub-Hypergraph (DKSH)*.

THEOREM 1. *The problem of utility maximizing integral memory-bound load shedding (IMLS) is NP-hard.*

A proof of the theorem can be found in Appendix B. We show in the following that IMLS is also hard to approximate.

THEOREM 2. *The problem of IMLS with n event types cannot be approximated within a factor of $2^{(\log n)^\delta}$, for some $\delta > 0$, unless 3SAT $\in$ DTIME($2^{n^{3/4+\epsilon}}$).*

This result is obtained by observing that the reduction from DKSH is approximation-preserving. Utilizing an inapproximability result in [29], we obtain the theorem above (a proof is in Appendix C).

It is worth noting that DKSH and related problems are conjectured to be very hard problems with even stronger inapproximability than what was proved in [29]. For example, authors in [24] conjectured that *Maximum Balanced Complete Bipartite Subgraph (BCBS)* is n^ϵ hard to approximate. If this is the case, utilizing a reduction from BCBS to DKSH [29], DKSH would be at least n^ϵ hard to approximate, which in turn renders IMLS n^ϵ hard to approximate given our reduction from DKSH.

While it is hard to solve or approximate IMLS efficiently in general, in the following sections we look at constraints that may apply to real-world CEP systems, and investigate special cases that enable us to approximate or even solve IMLS efficiently.

5.1.2 A general bi-criteria approximation

We reformulate the integral memory-bound problem into an alternative optimization problem (IMLS^l) with linear constraints as follows.

$$\text{(IMLS}^l\text{)} \qquad \max \sum_{Q_i \in \mathcal{Q}} n_i w_i y_i \tag{7}$$

$$\text{s.t.} \sum_{E_j \in \Sigma} \lambda_j m_j x_j \leq M$$

$$y_i \leq x_j, \forall E_j \in Q_i \tag{8}$$

$$y_i, x_j \in \{0,1\}$$

Observing that Equation (5) in IMLS is essentially morphed into an equivalent Equation (8). These two constraints are equivalent because y_i, x_j are all binary variables, y_i will be forced to 0 if there exists $x_j = 0$ with $E_j \in Q_i$.

Instead of maximizing utility, we consider the alternative objective of minimizing utility loss as follows. Set $\hat{y}_i = 1 - y_i$ be the complement of y_i, which indicates whether query Q_i is un-selected. We can change the utility gain maximization IMLS^l into a utility loss minimization problem IMLS^m. Note that utility gain is maximized if and only if utility loss is minimized.

$$\text{(IMLS}^m\text{)} \qquad \min \sum_{Q_i \in \mathcal{Q}} n_i w_i \hat{y}_i \tag{9}$$

$$\text{s.t.} \sum_{E_j \in \Sigma} \lambda_j m_j x_j \leq M$$

$$\hat{y}_i \geq 1 - x_j, \forall E_j \in Q_i \tag{10}$$

$$\hat{y}_i, x_j \in \{0,1\} \tag{11}$$

In IMLS^m Equation (10) is obtained by using $\hat{y}_i = 1 - y_i$ and Equation (8). Using this new minimization problem with linear structure, we prove a bi-criteria approximation result. Let OPT be the optimal loss with budget M in a loss minimization problem, then an (α, β)-bi-criteria approximation guarantees that its solution has at most $\alpha \cdot OPT$ loss, while uses no more than $\beta \cdot M$ budget. Bicriteria approximations have been extensively used in the context of resource augmentation (e.g., see [42] and references therein), where the algorithm is augmented with extra resources and the benchmark is an optimal solution without augmentation.

THEOREM 3. *The problem of IMLS^m admits a $(\frac{1}{\tau}, \frac{1}{1-\tau})$ bi-criteria-approximation, for any $\tau \in [0,1]$.*

For concreteness, suppose we set $\tau = \frac{1}{2}$. Then this result states that we can efficiently find a strategy that incurs at most 2 times the optimal utility loss with budget M, while using no more than $2M$ memory budget.

PROOF. Given a parameter $\frac{1}{\tau}$, we construct an event selection strategy as follows. First we drop the integrality constraint of IMLS^m to obtain its LP-relaxation. We solve the relaxed problem to get an optimal fractional solutions x_j^* and $\hat{y}_i^*$.

We can then divide queries $\mathcal{Q}$ into two sets, $\mathcal{Q}^a = \{Q_i \in \mathcal{Q} | \hat{y}_i^* \leq \tau\}$ and $\mathcal{Q}^r = \{Q_i \in \mathcal{Q} | \hat{y}_i^* > \tau\}$. Since $\hat{y}_i$ denotes whether query Q_i is un-selected, intuitively a smaller value means the query is more likely to be accepted. We can accordingly view $\mathcal{Q}^a$ as the set of "promising" queries, and $\mathcal{Q}^r$ as "unpromising" queries.

The algorithm works as follows. It preserves every query with $\hat{y}_i^* \leq \tau$, by selecting constituent events of Q_i into memory. So the set of query in $\mathcal{Q}^a$ is all accepted, while $\mathcal{Q}^r$ is all rejected.

We first show that the memory consumption is no more than $\frac{1}{1-\tau} M$. From Equation (10), we know the fractional solutions must satisfy

$$x_j^* \geq 1 - \hat{y}_i^*, \forall E_j \in Q_i \tag{12}$$

In addition, we have $\hat{y}_i^* \leq \tau, \forall Q_i \in \mathcal{Q}^a$. So we conclude

$$x_j^* \geq 1 - \tau, \forall Q_i \in \mathcal{Q}^a, E_j \in Q_i \tag{13}$$

Since we know x_j^* are fractional solutions to IMLS^m, we have

$$\sum_{E_j \in \mathcal{Q}^a} m_j \lambda_j x_j^* \leq \sum_{E_j \in \mathcal{Q}^a \cup \mathcal{Q}^r} m_j \lambda_j x_j^* = M \tag{14}$$

Here we slightly abuse the notation and use $E_j \in \mathcal{Q}^a$ to denote that there exists a query $Q \in \mathcal{Q}^a$ such that $E_j \in Q$.

Combining (13) and (14), we have

$$\sum_{E_j \in \mathcal{Q}^a} m_j \lambda_j (1 - \tau) \leq M$$

Notice that $\sum_{E_j \in \mathcal{Q}^a} m_j \lambda_j$ is the total memory consumption of our rounding algorithm, we have

$$\sum_{E_j \in \mathcal{Q}^a} m_j \lambda_j \leq \frac{M}{1-\tau}$$

Thus total memory consumption cannot exceed $\frac{M}{1-\tau}$.

We then need to show that the utility loss is bounded by a factor of $\frac{1}{\tau}$. Denote the optimal loss of IMLS^m as l^*, and the optimal loss with LP-relaxation as $\bar{l}^*$. We then have $\bar{l}^* \leq l^*$ because any feasible solution to IMLS^m is also feasible to the LP-relaxation of IMLS^m. In addition, we know

$$\sum_{Q_i \in \mathcal{Q}^r} n_i w_i \hat{y}_i^* \leq \sum_{Q_i \in \mathcal{Q}^a \cup \mathcal{Q}^r} n_i w_i \hat{y}_i^* = \bar{l}^* \leq l^*$$

So we can obtain

$$\sum_{Q_i \in \mathcal{Q}^r} n_i w_i \hat{y}_i^* \leq l^* \tag{15}$$

Based on the way queries are selected, we know for every rejected query

$$\hat{y}_i^* \geq \tau, \forall Q_i \in \mathcal{Q}^r \tag{16}$$

Combining (15) and (16), we get

$$\sum_{Q_i \in \mathcal{Q}^r} n_i w_i \tau \leq l^*$$

Observing that $\sum_{Q_i \in \mathcal{Q}^r} n_i w_i$ is the utility loss of the algorithm, we conclude that

$$\sum_{Q_i \in \mathcal{Q}^r} n_i w_i \leq \frac{l^*}{\tau}$$

This bounds the utility loss from optimal l^* by a factor of $\frac{1}{\tau}$, thus completing the proof. □

Note that since our proof is constructive, this gives an LP-relaxation based algorithm to achieve $(\frac{1}{\tau}, \frac{1}{1-\tau})$ bi-criteria-approximation of utility loss.

5.1.3 An approximable special case

Given that memory is typically reasonably abundant in today's hardware setup, in this section we will assume that the available memory capacity is large enough such that it can hold at least a few number of queries. If we set $f = \frac{\max_{Q_i} \sum_{E_j \in Q_i} m_j \lambda_j}{M}$ to be the ratio between the memory requirement of the largest query and available memory M. We know if M is large enough, then each query uses no more than fM memory, for some $f < 1$.

In addition, denote by $p = \max_j |\{Q_i | E_j \in Q_i, Q_i \in \mathcal{Q}\}|$ as the maximum number of queries that one event type participates in. We note that in practice there are problems in which each event participates in a limited number of queries. In such cases p will be limited to a small constant.

Assuming both p and f are some fixed constants, we obtain the following approximation result.

THEOREM 4. *Let p be the maximum number of queries that one event can participate in, and f be the ratio between the size of the largest query and the memory budget defined above, IMLS admits a $\frac{p}{1-f}$-approximation.*

The idea here is to leverage the fact that the maximum query-participation p is a constant to simplify the memory consumption constraint, so that a knapsack-like heuristic yields utility guarantees. We present a proof of this theorem in the full version of the paper [31].

5.1.4 A pseudo-polynomial-time solvable special case

We further consider the multi-tenant case where multiple CEP applications are consolidated into one single server or into one cloud infrastructure where the same set of underlying computing resources is shared across applications.

In this multi-tenancy scenario, since different CEP applications are interested in different aspects of real-world event occurrences, there typically is no or very limited sharing of events across different applications (the hospital hygiene system HyReminder and hospital inventory management system CIMS as mentioned in the Introduction, for example, have no event types in common. So do a network intrusion detection application and a financial application co-located in the same cloud). Using a hyper-graph model, a multi-tenant CEP system can be represented as a hyper-graph H, where each event type is represented as a vertex and each query as a hyper-edge. If there is no sharing of event types among CEP applications, then each connected component of H corresponds to one CEP application. Let k be the size of the largest connected component of H, then k is essentially the maximum number of event types used in any one CEP application, which may be limited to a small constant (the total number of event types across multiple CEP applications is not limited). Assuming this is the case, we have the following special case that is pseudo-polynomial time solvable.

THEOREM 5. *In a multi-tenant CEP system where each CEP tenant uses a disjoint set of event types, if each CEP tenant uses no more than k event types, the problem of IMLS can be solved in time $O(|\Sigma||\mathcal{Q}|M2^{k^2})$.*

Our proof utilizes a dynamic programming approach developed for Set-union Knapsack problem [27]. A detailed proof of this theorem can be found in Appendix D.

We note that we do not assume the total number of event types across multiple CEP tenants to be limited. In fact, the running time grows linearly with the total number of event types and queries across all CEP tenants.

Lastly, we observe that the requirement that events in each CEP tenant are disjoint can be relaxed. As long as the sharing of event types between different CEP tenants are limited, such that the size of the largest component of H mentioned above is bounded by k, the result in Theorem 5 holds.

5.2 The fractional variant (FMLS)

In this section, we consider the fractional variant of the memory-bound load shedding problem. In this variant, instead of taking an all-or-nothing approach to either include or exclude *all* event instances of certain types in memory, we use a random sampling operator [32], which samples *some* arriving events uniformly at random into the memory. Denote by $\overline{x}_j \in [0, 1]$ the sampling probability for each event type E_j. The fractional variant memory-bound load shedding (FMLS) can be written as follows.

$$\text{(FMLS)} \qquad \max \sum_{Q_i \in \mathcal{Q}} n_i w_i \overline{y}_i \tag{17}$$

$$\text{s.t.} \sum_{E_j \in \Sigma} \lambda_j m_j \overline{x}_j \leq M \tag{18}$$

$$\overline{y}_i = \prod_{E_j \in Q_i} \overline{x}_j \tag{19}$$

$$0 \leq \overline{x}_j \leq 1 \tag{20}$$

The integrality constraints are essentially dropped from the integral version of the problem, and are replaced by constraints in (20). We use fractional sampling variables $\overline{x}_j$ and $\overline{y}_i$ to differentiate from binary variables x_j, y_i. Note that Equation (19) states that the probability that a query result is produced, $\overline{y}_i$, is the cross-product of sampling rates of constituent events since each event is sampled randomly and independently of each other.

5.2.1 A general complexity analysis

In the FMLS formulation, if we fold Equation (19) into the objective function in (17), we obtain

$$\max \sum_{Q_i \in \mathcal{Q}} n_i w_i \prod_{E_j \in Q_i} \overline{x}_j \tag{21}$$

This makes FMLS a polynomial optimization problem subject to a knapsack constraint (18).

Since we are maximizing the objective function in Equation (21), it is well known that if the function is concave, then convex optimization techniques [14] can be used to solve such problems optimally. However, we show that except the trivial case where each query has exactly one event (i.e., (21) becomes linear), in general Equation (21) is not concave.

LEMMA 1. *If the objective function in Equation* (21) *is non-trivial (that is, at least one query has more than one event), then* (21) *is non-concave.*

We show a proof of Lemma 1 in Appendix E.

Given this non-concavity result, it is unlikely that we can hope to exploit special structures of the Hessian matrix to solve FMLS. In particular, convex optimization techniques like KKT conditions or gradient descent [14] can only provide local optimal solutions, which may be far away from the global optimal.

On the other hand, while the general polynomial program is known to be hard to solve [11, 45], FMLS is a special case where all coefficients are positive, and the constraint is a simple knapsack constraint. Thus the hardness results in [11, 45] do not apply to FMLS. We show the hardness of FMLS by using the Motzkin-Straus theorem [38] and a reduction from the Clique problem.

THEOREM 6. *The problem of fractional memory-bound load shedding (FMLS) is NP-hard. FMLS remains NP-hard even if each query has exactly two events.*

A full proof of this theorem can be found in Appendix F.

So despite the continuous relaxation of the decision variables of IMLS, FMLS is still hard to solve. However, in the following we show that FMLS has special structure that allows it to be solved approximately under fairly general assumptions.

5.2.2 Definitions of approximation

We will first describe two definitions of approximation commonly used in the numerical optimization literature.

The first definition is similar to the approximation ratio used in combinatorial optimization.

DEFINITION 7. *[30] Given a maximization problem P that has maximum value $v_{max}(P) > 0$. We say a polynomial time algorithm has an absolute approximation ratio $\epsilon \in [0,1]$, if the value found by the algorithm, $v(P)$, satisfies $v_{max}(P) - v(P) \leq \epsilon\, v_{max}(P)$.*

The second notion of *relative approximation ratio* is widely used in the optimization literature [22, 30, 40, 45].

DEFINITION 8. *[30] Given a maximization problem P that has maximum value $v_{max}(P)$ and minimum value $v_{min}(P)$. We say a polynomial time algorithm has a relative approximation ratio $\epsilon \in [0,1]$, if the value found by the algorithm, $v(P)$, satisfies $v_{max}(P) - v(P) \leq \epsilon(v_{max}(P) - v_{min}(P))$.*

Relative approximation ratio is used to bound the quality of solutions relative to the possible value range of the function. We refer to this as *ϵ-relative-approximation* to differentiate from *ϵ-approximation* used Definition 7.

Note that in both cases, ϵ indicates the size of the gap between an approximate solution and the optimal value. So smaller ϵ values are desirable in both definitions.

5.2.3 Results for relative approximation bound

In general, the feasible region specified in FMLS is the intersection of a unit hyper-cube, and the region below a scaled simplex. We first normalize (18) in FMLS using a change of variables. Let $\bar{x}'_j = \frac{\lambda_j m_j \bar{x}_j}{M}$ be the scaled variables. We can obtain the following formulation FMLS'.

$$\text{(FMLS')} \qquad \max \sum_{Q_i \in \mathcal{Q}} n_i w_i \prod_{E_j \in Q_i} \frac{M}{\lambda_j m_j} \bar{x}'_j \tag{22}$$

$$\text{s.t.} \sum_{E_j \in \Sigma} \bar{x}'_j = 1 \tag{23}$$

$$0 \leq \bar{x}'_j \leq \frac{\lambda_j m_j}{M} \tag{24}$$

Note that the inequality constraint (18) in FMLS is now replaced by an equality constraint (23) in FMLS'. This will not change the optimal value of FMLS as long as $\sum_{E_j} \frac{\lambda_j m_j}{M} \geq 1$ (otherwise, although $\sum_{E_j \in \Sigma} \bar{x}'_j = 1$ is unattainable, the memory budget becomes sufficient and the optimal solution is trivial). This is because all coefficients in (17) are positive, pushing any x_i to a larger value will not hurt the objective value. Since the constraint (18) is active for at least one global optimal point in FMLS, changing the inequality in the knapsack constraint to an equality in (23) will not change the optimal value.

Denote by $d = \max\{|Q_i|\}$ the maximum number of event types in a query. We will assume in this section that d is a fixed constant. Note that this is a fairly realistic assumption, as d tends to be very small in practice (in HyReminder [47], for example, the longest query has 6 event types, so $d = 6$). Observe that d essentially corresponds to the degree of the polynomial in the objective function (22).

An approximation using co-centric balls. Using a randomized procedure from the optimization literature [30], we show that FMLS' can be approximated by bounding the feasible region using two co-centric balls to obtain a (loose) relative-approximation ratio as follows.

THEOREM 7. *The problem FMLS' admits a relative approximation ratio ϵ, where $\epsilon = 1 - O\left(|\Sigma|^{-\frac{d-2}{2}}(t^2+1)^{-\frac{d}{2}}\right)$ and $t = \min\left(\min_{E_j}\left(\frac{\lambda_j m_j}{M}\right), \frac{1}{\sqrt{|\Sigma|}}\right)$.*

A proof of this result can be found in Appendix G. Note that this is a general result that only provides a loose relative approximation bound, which is a function of the degree of the polynomial d, the number of event types $|\Sigma|$, and $\frac{\lambda_j m_j}{M}$, and cannot be adjusted to a desirable level of accuracy.

An approximation using simplex. We observe that the feasible region defined in FMLS' has special structure. It is a subset of a simplex, which is the intersection between a standard simplex (Equation (23)) and box constraints (Equation (24)).

There exists techniques that produces relative approximations for polynomials defined over simplex. For example, [22] uses a grid-based approximation technique, where the idea is to fit a uniform grid into the simplex, so that values on all nodes of the grid are computed and the best value is picked as an approximate solution. Let Δ_n be an n dimensional simplex, then $(x \in \Delta_n | kx \in Z_+^n)$ is defined as a k-grid over the simplex. The quality of approximation is determined by the granularity the uniform grid: the finer the grid, the tighter the approximation bound is.

The result in [22], however, does not apply here because the feasible region of FMLS' represents a subset of the standard simplex. We note that there exist a special case where if $\frac{\lambda_j m_j}{M} \geq 1, \forall j$ (that is, if the memory requirement of a single event type exceeds the memory capacity), the feasible region degenerates to a simplex, such that we can use grid-based technique for approximation.

THEOREM 8. *In FMLS', if for all j we have $\frac{\lambda_j m_j}{M} \geq 1$ then the problem admits a relative approximation ratio of ϵ, where*

$$\epsilon = O\left(1 - \frac{k!}{(k-d)!k^d}\right)$$

for any $k \in \mathbb{Z}^+$ such that $k > d$. Here k represents the number of grid points along one dimension.

Note that as $k \to \infty$, approximation ratio $\epsilon \to 0$.

A detailed proof of this result can be found in Appendix H. This result can provide increasingly accurate relative approximation bound for a larger k. It can be shown that for a given k, a total of $\binom{|\Sigma|+k-1}{|\Sigma|-1}$ number of evaluations of the objective function is needed.

5.2.4 Results for absolute approximation bound

Results obtained so far use the notion of relative approximation (Definition 8). In this section we discuss a special case in which FMLS' can be approximated relative to the optimal value (Definition 7).

We consider a special case in which queries are *regular* with no repeated events. That is, in each query, no events of the same type occur more than once (e.g., query SEQ(A,B,C) is a query without repeated events while SEQ(A,B,A) is not because event A appears twice). This may be a fairly general assumption, as queries typically detect events of different types. HyReminder [47] queries, for instance, use no repeated events in the same query (two such example Q1 and Q2 are given in the Introduction). In addition, we require that each query has the same length as measured by the number of events.

With the assumption above, the objective function Equation (22) becomes a homogeneous multi-linear polynomial, while the feasible region is defined over a sub-simplex that is the intersection of a

cube and a standard simplex. We extend a random-walk based argument in [40] from standard simplex to the sub-simplex, and show an (absolute) approximation bound.

THEOREM 9. *In FMLS', if $\frac{\lambda_j m_j}{M}$s are fixed constants, in addition if every query has no repeated event types and is of same query length, d, then a constant-factor approximation can be obtained for FMLS' in polynomial time. In particular, we can achieve a $(1-\beta(\frac{k!}{(k-d)!k^d}))$-approximation, by evaluating Equation (21) at most $\binom{|\Sigma|+k-1}{|\Sigma|-1}$ times, where $\beta = \min\left(\min_j\left(\frac{\lambda_j m_j}{M}\right), 1\right)^d$, and $k > d$ is a parameter that controls approximation accuracy.*

We use a scaling method to extend the random-walk argument in [40] to the sub-simplex in order to get the desirable constant factor approximation. A detailed proof can be found in [31]. Note that, by selecting $k = O(d^2)$ we can get $\frac{k!}{(k-d)!k^d}$ close to 1. Also note that if $\frac{\lambda_j m_j}{M} \geq 1$ for all j, $\beta = 1$ and we can get an approximation arbitrarily close to the optimal value by using large k.

6. CPU-BOUND LOAD SHEDDING

In this section we consider the scenario where memory is abundant, while CPU becomes the limiting resource that needs to be budgeted appropriately. CPU-bound problems turn out to be easy to solve.

6.1 The integral variant (ICLS)

In the integral variant CPU load shedding, we again use binary variables y_i to denote whether results of Q_i can be generated. For each query Q_i, at most n_i number of query matches can be produced. Assuming the utility weight of each result is w_i, and the CPU cost of producing each result is c_i, when Q_i is selected ($y_i = 1$) a total of $n_i w_i$ utility can be produced, at the same time a total of $n_i c_i$ CPU resources are consumed. That yields the following ICLS problem.

$$\text{(ICLS)} \quad \max \sum_{Q_i \in \mathcal{Q}} n_i w_i y_i \tag{25}$$

$$\text{s.t.} \sum_{Q_i \in \mathcal{Q}} n_i c_i y_i \leq C \tag{26}$$

$$y_i \in \{0,1\} \tag{27}$$

ICLS is exactly the standard 0-1 knapsack problem, which has been studied extensively. We simply cite a result from [46, Chapter 5] for completeness.

THEOREM 10. *[46] The integral CPU-bound load shedding problem (ICLS) is NP-complete. It admits a fully polynomial approximation scheme (FPTAS).*

ICLS is thus an easy variant among the load shedding problems studied in this work.

6.2 The fractional variant (FCLS)

Similar to the memory-bound load shedding problems, we also investigate the fractional variant where a sampling operator is used to select a fixed fraction of query results. Instead of using binary variables y_i, we denote by $\overline{y}_i$ the percentage of queries that are sampled and processed by CPU for output. The integrality constraints in ICLS are again dropped, and we can have the following FCLS formulation.

$$\text{(FCLS)} \quad \max \sum_{Q_i \in \mathcal{Q}} n_i w_i \overline{y}_i \tag{28}$$

$$\text{s.t.} \sum_{Q_i \in \mathcal{Q}} n_i c_i \overline{y}_i \leq C \tag{29}$$

$$0 \leq \overline{y}_i \leq 1, \text{for all } i \tag{30}$$

Since n_i, w_i, c_i are all constants, this is a simple linear program problem that can be solved in polynomial time. We conclude that FCLS can be efficiently solved.

7. DUAL-BOUND LOAD SHEDDING

Lastly, we study the dual-bound load shedding problem, where both CPU and memory resources can be limited.

7.1 The integral variant (IDLS)

The integral dual-bound load shedding (IDLS) can be formulated by combining CPU and memory constraints.

$$\begin{aligned} \text{(IDLS)} \quad & \max \sum_{Q_i \in \mathcal{Q}} n_i w_i y_i \\ & \text{s.t.} \sum_{E_j \in \Sigma} \lambda_j m_j x_j \leq M \\ & \sum_{Q_i \in \mathcal{Q}} n_i c_i y_i \leq C \\ & y_i \leq \prod_{E_j \in Q_i} x_j \qquad (31) \\ & x_j, y_i \in \{0,1\} \end{aligned}$$

Binary variables x_j, y_i again denote event selection and query selection, respectively. Note that in order to respect the CPU constraint, not all queries whose constituent events are available in memory can be produced. This is modeled by an inequality in Equation (31).

We show that the loss minimization version of IDLS admits a tri-criteria approximation.

THEOREM 11. *Denote by M the given memory budget C the given CPU budget, and l^* the optimal utility loss with that budget. IDLS admits $(\frac{1}{\tau}, \frac{1}{1-\tau}, \frac{1}{1-\tau})$ tri-criteria-approximation for any $\tau \in [0,1]$. That is, for any $\tau \in [0,1]$, we can compute a strategy that uses no more than $\frac{1}{1-\tau}M$ memory $\frac{1}{1-\tau}C$ CPU, and incur no more than $\frac{l^*}{\tau}$ utility loss.*

The idea of the proof is to use LP-relaxation and round the resulting fractional solution, which is similar to Theorem 3. In addition, we show that the approximable special case of IMLS (Theorem 4) also holds for IDLS. Detailed proofs of both theorems can be found in [31].

THEOREM 12. *Let p be the maximum number of queries that one event can participate in, and $f = \frac{\max_{Q_i} \sum_{E_j \in Q_i} m_j \lambda_j}{M}$ be the ratio between the size of the largest query and the memory budget. IDLS is $\frac{p}{1-f}$-approximable in pseudo polynomial time.*

7.2 The fractional variant (FDLS)

The fractional dual-bound problem once again relaxes the integrality constraints in IDLS to obtain the following FDLS problem.

$$\text{(FDLS)} \quad \max \sum_{Q_i \in \mathcal{Q}} n_i w_i \overline{y}_i \tag{32}$$

$$\text{s.t.} \sum_{E_j \in \Sigma} \lambda_j m_j \overline{x}_j \leq M$$

$$\sum_{Q_i \in \mathcal{Q}} n_i c_i \overline{y}_i \leq C \tag{33}$$

$$\overline{y}_i \leq \prod_{E_j \in Q_i} \overline{x}_j \tag{34}$$

$$\overline{x}_j, \overline{y}_i \in [0,1]$$

First of all, we note that the hardness result in Theorem 6 still holds for FDLS, because FMLS is simply a special case of FDLS without constraint (33).

The approximation results established for FMLS, however, do not carry over easily. If we fold (34) into Equation (32), and look at the equivalent minimization problem by taking the inverse of the objective function, then by an argument similar to Lemma 1, we can show that objective function is non-convex in general. In addition, we can show that except in the trivial linear case, the constraint in Equation (33) is also non-convex using a similar argument. So we are dealing with a non-convex optimization subject to non-convex constraints.

It is known that solving or even approximating non-convex problems with non-convex constraints to global optimality is hard [14, 21]. Techniques dealing with such problems are relatively scarce in the optimization literature. Exploiting special structure of FDLS to optimize utility is an interesting area for future work.

8. CONCLUSIONS AND FUTURE WORK

In this work we study the problem of load shedding in the context of the emerging Complex Event Processing model. We investigate six variants of CEP load shedding under various resource constraints, and demonstrate an array of complexity results. Our results shed some light on the hardness of load shedding CEP queries, and provide some guidance for developing CEP shedding algorithms in practice.

CEP load shedding is a rich problem that so far has received little attention from the research community. We hope that our work will serve as a springboard for future research in this important aspect of the increasingly popular CEP model.

Acknowledgment. We thank anonymous reviewers for their valuable comments on the paper, including a suggested improvement of the proof for Theorem 1, among many other things.

9. REFERENCES

[1] Microsoft StreamInsight: http://www.microsoft.com/sqlserver/2008/en/us/r2-complex-event.aspx.

[2] StreamBase: http://www.streambase.com.

[3] Sybase Aleri streaming platform high-performance CEP: http://www.sybase.com/files/data_sheets/ sybase_aleri_streamingplatform_ds.pdf.

[4] Wavemark system: Clinical inventory management solution www.wavemark.net/healthcare.action.

[5] A. Adi, D. Botzer, G. Nechushtai, and G. Sharo. Complex event processing for financial services. *Money Science*.

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

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

[8] L. Aniello, G. Lodi, and R. Baldoni. Inter-domain stealthy port scan detection through complex event processing. In *Proceedings of EWDC*, 2011.

[9] B. Babcock, S. Babu, R. Motwani, and M. Datar. Chain: Operator scheduling for memory minimization in data stream systems. In *proceedings of SIGMOD*, 2003.

[10] B. Babcock, M. Datar, and R. Motwani. Load shedding techniques for data stream systems. In *Workshop on Management and Processing of Data Streams*, 2003.

[11] M. Bellare and P. Rogaway. The complexity of approximating a nonlinear program. In *Mathematical Programming 69*, 1993.

[12] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[13] J. M. Boyce and D. Pittet. Guideline for hand hygiene in healthcare settings. *CDC Recommendations and Reports*, 51, 2002.

[14] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[15] K. Carter. Optimizing critical medical supplies in an acute care setting. *RFID in Health Care*, 2009.

[16] B. Chandramouli, J. Goldstein, and D. Maier. High-performance dynamic pattern matching over disordered streams. In *proceedings of VLDB*, 2010.

[17] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xin, and S. Zdonik. Scalable distributed stream processing. In *proceedings of CIDR*, 2003.

[18] Y. Chi, H. J. Moon, and H. Hacigümüs. icbs: Incremental costbased scheduling under piecewise linear slas. In *proceedings of VLDB*, 2011.

[19] Y. Chi, H. J. Moon, H. Hacigümüs, and J. Tatemura. Sla-tree: a framework for efficiently supporting sla-based decisions in cloud computing. In *proceedings of EDBT*, 2011.

[20] A. Das, J. Gehrke, and M. Riedewald. Approximate join processing over data streams. In *proceedings of SIGMOD*, 2003.

[21] A. d'Aspremont and S. Boyd. Relaxations and randomized methods for nonconvex qcqps.

[22] E. de Klerk, M. Laurent, and P. Parrilo. A PTAS for the minimization of polynomials of fixed degree over the simplex. In *Theoretical Computer Science*, 2006.

[23] Y. Diao, N. Immerman, and D. Gyllstrom. Sase+: An agile language for kleene closure over event streams. Technical report, University of Massachusetts at Amherst, 2007.

[24] U. Feige and S. Kogan. Hardness of approximation of the balanced complete bipartite subgraph problem. Technical report, Weizmann Institute, 2004.

[25] B. Gedik, K.-L. Wu, and P. S. Yu. Efficient construction of compact shedding filters for data stream processing. In *proceedings of ICDE*, 2008.

[26] B. Gedik, K.-L. Wu, P. S. Yu, and L. Liu. A load shedding framework and optimizations for m-way windowed stream joins. In *proceedings of ICDE*, 2007.

[27] O. Goldschmidt, D. Nehme, and G. Yu. Note: On the set-union knapsack problem. In *Applied Probability & Statistics*, 2006.

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

[29] M. T. Hajiaghayi, K. Jain, K. Konwar, L. C. Lau, I. I. Mandoiu, A. Russell, A. Shvartsman, and V. V. Vazirani. The minimum k-colored subgraph problem in haplotyping and DNA primer selection. In *International Workshop on Bioinformatics Research and Applications*, 2006.

[30] S. He, Z. Li, and S. Zhang. General constrained polynomial optimization: an approximation approach. In *Technical report*, 2009.

[31] Y. He, S. Barman, and J. F. Naughton. On Load Shedding in Complex Event Processing. *Preprint arXiv:1312.4283*, 2013.

[32] T. Johnson, S. Muthukrishnan, and I. Rozenbaum. Sampling algorithms in a stream operator. In *proceedings of SIGMOD*, 2005.

[33] J. Kang, J. F. Naughton, and S. D. Viglas. Evaluating window joins over unbounded streams. In *proceedings of ICDE*, 2003.

[34] W. Kleiminger, E. Kalyvianaki, and P. Pietzuch. Balancing load in stream processing with the cloud. In *SMDB*, 2011.

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

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

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

[38] T. Motzkin and E. Straus. Maxima for graphs and a new proof of a theorem of turán. *Canadian Jornal of Mathamatics*, 17, 1965.

[39] S. Narayanan and F. Waas. Dynamic prioritization of database queries. In *proceedings of ICDE*, 2011.

[40] Y. Nesterov. Random walk in a simplex and quadratic optimization over simplex polytopes. *Center for Operations Research and Econometrics (CORE)*, 2003.

[41] O. Papaemmanouil, U. Cetintemel, and J. Jannotti. Supporting generic cost models for wide-area stream processing. In *proceedings of ICDE*, 2009.

[42] K. P. J. Sgall and E. Torng. *Handbook of scheduling: Algorithms, models, and performance analysis*. 2004.

[43] U. Srivastava and J. Widom. Memory-limited execution of windowed stream joins. In *proceedings of VLDB*, 2004.

[44] N. Tatbul, U. Cetintemel, S. B. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *proceedings of VLDB*, 2003.

[45] S. A. Vavasis. Approximation algorithms for concave quadratic programming. Technical report, Cornell University, 1990.

[46] V. Vazirani. *Approximation Algorithms*. Springer-Verlag Berlin Heidelberg, 2003.

[47] D. Wang and E. Rundensteiner. Active complex event processing over event streams. In *proceedings of VLDB*, 2011.

[48] M. Wei, E. A. Rundensteiner, and M. Mani. Utility-driven load shedding for XML stream processing. In *WWW*, 2008.

[49] M. Wei, E. A. Rundensteiner, and M. Mani. Achieving high output quality under limited resources through structure-based spilling in XML streams. In *proceedings of VLDB*, 2010.

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

[51] A. Widder, R. v. Ammon, G. Hagemann, and D. Schönfeld. An approach for automatic fraud detection in the insurance domain. In *AAAI Spring Symposium: Intelligent Event Processing*, 2009.

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

[53] Y. Xing, J.-H. Hwang, U. Cetintemel, and S. Zdonik. In *Providing Resiliency to Load Variations in Distributed Stream Processing*, 2006.

[54] A. Yao. Probabilistic computations: Toward a unified measure of complexity. In *proceedings of FOCS*, 1977.

APPENDIX

A. PROOF OF PROPOSITION 1

PROOF. First, we show that no deterministic online shedding algorithm can achieve a competitive ratio that is independent of the length of the event sequence. In order to establish this, it is sufficient to specify a distribution over event sequences for which the competitive ratio—defined to be the ratio of the expected utility obtained by the algorithm and the (offline) optimal utility—achieved by any deterministic algorithm must depend on the length of the sequence.

We construct such a distribution as follows. Suppose we have a universe of $3m$ event types, $\Sigma = \{E_i\} \cup \{E'_i\} \cup \{E''_i\}$, $i \in [m]$. Assume that there are $2m$ queries $SEQ(E_i, E''_i)$ and $SEQ(E'_i, E''_i)$, $\forall i \in [m]$, each with unit utility weight. The event sequence is of the form $(e_1, e_2, ..., e_m, X)$, where for all $i \in [m]$, e_i is set to be either E_i with probability $1/2$ or E'_i with probability $1/2$. In addition, X is drawn from the uniform distribution over $\{E''_i : i \in [m]\}$.

Suppose the system can only hold two events in memory due to memory constraints. The optimal offline algorithm has a utility of 1. This is because it can look at the type of event X, denoted by E''_k, and keep the corresponding event e_k which is of type either E_k or E'_k, and has arrived previously. This ensures that irrespective of the instantiation of the event stream one query (either $SEQ(E_k, E''_k)$ or $SEQ(E'_k, E''_k)$ match can be produced.

A deterministic online algorithm, on the other hand, needs to select one event into memory before seeing the event type of X. In particular, if X is E''_k then the deterministic algorithm is able to report the correct query (either (E_k, E''_k) or (E'_k, E''_k)) only if it stored e_k.[1]

Thus, if a deterministic algorithm stores e_k then it succeeds only if X is E''_k. The latter event happens with probability $\frac{1}{m}$; therefore, the probability that a deterministic algorithm produces a match is $\frac{1}{m}$. Its expected utility is also $\frac{1}{m}$ given that queries all have a unit weight.

Note that the competitive ration, i.e., the ratio of the utility produced by the optimal offline algorithm, and an online deterministic algorithm, is $\frac{1}{m}$. Here m is the size of the event stream minus one. Since an event stream can be unbounded, the ratio can be arbitrarily bad.

Next, we show that no randomized algorithm can do any better. Using Yao's principle [54], we know that the expected utility of a randomized algorithm against the worst case input stream, is no more than the expected utility of the best deterministic algorithm against the input distribution. Since we know any deterministic algorithm against the input distribution constructed above is $\frac{1}{m}$, it follows that the expected utility of a randomized algorithm against the worst case input stream from this input distribution is at most $\frac{1}{m}$, thus completing the proof. □

B. PROOF OF THEOREM 1

[1]Note that the algorithm can only report queries that have a match, i.e., it cannot report false positives. Hence, if X is E''_k, then the algorithm cannot simply guess and declare a match, say (E_k, E''_k), without any knowledge of e_k. Specifically, e_k could instead be E'_k and in such a case the declared match would be incorrect.

PROOF. We obtain the hardness result by a reduction from *Densest k-sub-hypergraph (DKSH)*. Recall that given a hypergraph $G = (V, H)$, the decision version of DKSH is to determine if there exists an induced subgraph $G' = (V', H')$, such that $|V'| \leq k$, and $|H'| \geq B$ for some given constant B.

Given any instance of the DKSH problem, we construct an IMLS problem as follows. We build a bijection $\phi : V \rightarrow \Sigma$, so that each vertex $v_j \in V$ corresponds to one event type $E_j \in \Sigma$. For each hyperedge $h_i \in H$, using vertices v_k that are endpoints of h_i to construct a corresponding query Q_i, such that $\forall v_k \in h_i$, $\phi(v_k) \in Q_i$. Set all E_j to have unit cost ($m_j \lambda_j = 1$) and all Q_i have unit weight ($w_i n_i = 1$), and lastly set the memory budget to k.

We first show the forward direction, that is if there exists a solution to DKSH, i.e., a k-sub-hypergraph with at least B hyper-edges, then there exists a solution to IMLS with no more than k memory cost but achieves B utility. Suppose the subgraph $G' = (V', H')$ is the solution to DKSH. In the corresponding IMLS problem, if we keep all event types $\phi(v'), \forall v' \in V'$, our solution has a memory cost of $|V'|$, which is no more than k since G' is a k-sub-hypergraph. This ensures the constructed solution is feasible. Furthermore, the utility of the IMLS solution is exactly $|H'|$ by our unit weight construction. Since we know $|H'| \geq B$, this completes the proof in this direction.

In the other direction, let the set of events selected in IMLS be $S \subseteq \Sigma$ while the set of dropped events be $D = \Sigma \setminus S$. The set of vertices V_s corresponding to S induces a subgraph $G_s = (V_s, H_s)$. S respects memory-bounds in IMLS implies $|V_s| \leq k$, so G_s is a valid k-sub-hyper-graph. In addition, S produces utility B in IMLS ensures that $|H_s| \geq B$, thus completing the hardness proof. □

C. PROOF OF THEOREM 2

PROOF. We note that the reduction from *Densest k-sub-hypergraph (DKSH)* discussed above is approximation preserving. Utilizing a hardness result of DKSH [29], which establishes that DKSH cannot be approximated within a factor of $2^{(\log n)^{\delta}}$ for some $\delta > 0$, we obtain the inapproximability result of IMLS. □

D. PROOF OF THEOREM 5

Proof Sketch. The IMLS problem studied in this work can be formulated as a Set-union Knapsack problem [27]. A dynamic programming algorithm is proposed in [27] for Set-union Knapsack, which however runs in exponential time. If we define an *adjacency graph* G by representing all events as graph vertices, and each pair of vertices are connected if the corresponding events co-occur in the same query. The exponent of the running time is shown to be no more than the cut-width of the induced adjacency graph G, $cw(G)$. Recall that cut-width of a graph G is defined as the smallest integer k such that the vertices of G can be arranged in a linear layout $[v_1, ..., v_n]$ such that for every $i \in [1, n-1]$, there are at most k edges with one endpoint in $\{v_1, ..., v_i\}$ and another endpoint in $\{v_{i+1}, ..., v_n\}$.

In the context of a multi-tenant CEP system, assuming each non-overlapping CEP tenant uses at most k event types, then the size of the largest component of the adjacency graph G is at most k. This, combines with the fact that the degree of each vertex in each component is at most k, ensures that $cw(G) \leq k^2$. The running time of the dynamic programming approach in [27] can then be bounded by $O(|\Sigma||\mathcal{Q}|M2^{k^2})$. Note that this result is pseudo-polynomial, because the running time depends on the value of memory budget M instead of the number of bits it needs to represent it. □

E. PROOF OF LEMMA 1

PROOF. Denote H as the Hessian matrix of (21) representing its second order partial derivatives. In the trivial case where (21) is just a linear function (each query has exactly one event type), H is an all-zero matrix with all-zero eigenvalues, which is trivially concave.

In general, (21) is a nonlinear polynomial (i.e., at least one query has more than one event). Since (21) is a polynomial with positive coefficients and positive exponents, and $\overline{x}_j \geq 0, \forall j$, we know all non-zero second order partial derivatives of (21) are positive, and thus all non-zero entries of H are positive. Denote by h_{ij} the ith row, jth column entry of H, given that $h_{ij} \geq 0, \forall i, j$, we know the trace of the Hessian matrix $tr(H) = h_{11} + h_{22} + ... + h_{|\Sigma||\Sigma|} \geq 0$. From linear algebra, we know that $tr(H)$ equals the sum of the eigenvalues of H.

Since H is a Hessian matrix, we know it is symmetric and its eigenvalues are all real. In addition, at least one eigenvalue is non-zero because H is not an all-zero matrix.

We show by contradiction that H must have at least one positive eigenvalue. Suppose this is not the case. Since H has at least one non-zero eigenvalue, all its non-zero eigenvalues have to be negative, which implies that the sum of all eigenvalues are negative, thus we have $tr(H) < 0$. This contradicts with the fact that $tr(H) \geq 0$. Therefore H must have at least one positive eigenvalue, which ensures that H is non-concave. □

F. PROOF OF THEOREM 6

PROOF. We show the hardness of this problem by a reduction from the Clique problem. Given a graph $G = (V, E)$, the decision version of the Clique problem is to determine if there exists a clique of size k in G.

From any instance of the Clique problem, we construct an instance of the FMLS problem as follows. We build a bijective function $\phi : V \to \Sigma$ to map each vertex $v_j \in V$ to an event $\phi(v_j) \in \Sigma$. We set a unit memory consumption for each event ($\lambda_j m_j = 1$), and a unit knapsack capacity. So we get $\sum_{E_j \in \Sigma} \overline{x}_j \leq 1$. Furthermore, given an edge $e_i = (v_l, v_k) \in E$, we build a query $(\phi(v_l), \phi(v_k))$ with unit utility weight ($n_i w_i = 1$). We then essentially have a unit-weight, unit-cost, length-two-query FMLS that corresponds to the graph G. This gives rise to the following bilinear optimization problem subject to a knapsack constraint.

$$\begin{aligned} \max \quad & \sum_{Q_i:(E_l,E_k)\in\mathcal{Q}} \overline{x}_l \overline{x}_k \\ \text{s.t.} \quad & \sum_{E_j \in \Sigma} \overline{x}_j \leq 1 \\ & 0 \leq \overline{x}_j \leq 1 \end{aligned} \tag{35}$$

Since we are dealing with a maximization problem, and the coefficients of the objective are all non-negative, increasing $\overline{x}_j$ values will not "hurt" the objective. Thus, the knapsack constraint in (35) can be changed into a standard simplex constraint $\sum_{E_j \in \Sigma} \overline{x}_j = 1$ without changing the optimal value of the problem.

Given the FMLS defined above, the decision version of FMLS is to decide if there exists a fractional strategy such that the total utility value is at least u. We first show that if there exists a clique of size k in G, then the value of the FMLS we construct is at least $\frac{k-1}{2k}$. To show this connection, we use the Motzkin-Straus theorem [38], which states that global maximum over the standard simplex is attained when values are distributed evenly among the largest clique from the graph. So if there is a clique of size k in graph G, then the k-clique has a total of $\binom{k}{2}$ number of edges. Since each edge produces a value of $(\frac{1}{k})^2$, the the optimal value of FMLS is at least $\binom{k}{2}(\frac{1}{k})^2 = \frac{k-1}{2k}$.

We now show the other direction, that is if the optimal value FMLS problem we construct is no less than $\frac{k-1}{2k}$, then the graph G has a clique of size no less than k. We show this by contradiction. Suppose the size of the maximum clique of G is c, $c < k$. Then by the Motzkin-Straus theorem [38] the global maximum of the FMLS we construct is at most $\frac{c-1}{2c}$, which is less than $\frac{k-1}{2k}$ because $c < k$. This contradicts with the fact that the optimal value is no less than $\frac{k-1}{2k}$, therefore, the graph must have a clique of size at least k.

We note that by using a reduction from Clique, we have shown that even in the restricted case where the objective function in Equation (21) is a bi-linear function (a summation of quadratic square-free monomials), or in other words each query has exactly two events, FMLS remains NP-hard. □

G. PROOF OF THEOREM 7

PROOF. Let $B(t) = \{x \in \mathcal{R}^n, \|x\|_2 \leq t\}$ be a ball constraint. We show that the feasible region of FMLS', denoted by S, satisfies $B(t) \subseteq S \subseteq B(1)$, where $t = \min(\min_{E_j}(\frac{\lambda_j m_j}{M}), \frac{1}{\sqrt{|\Sigma|}})$.

First, we know that any feasible solution satisfies $\|x\|_1 \leq 1$. It can be shown that $\forall x, \|x\|_2 \leq \|x\|_1$. Thus we know $\|x\|_2 \leq 1$, and $S \subseteq B(1)$.

On the other hand, the ball inside S is limited by the shortest edge of the hyper-rectangle, $\min_{E_j}(\frac{\lambda_j m_j}{M})$, and the largest possible ball inside standard simplex, which has a radius of $\frac{1}{\sqrt{|\Sigma|}}$. So we have $t = \min(\min_{E_j}(\frac{\lambda_j m_j}{M}), \frac{1}{\sqrt{|\Sigma|}})$. The largest ball inside S is thus $B(t)$.

Authors in [30] show that if a convex feasible region contains a ball constraint, and in addition is bounded by another ball constraint, the polynomial program can be approximated with a relative approximation ratio that is a function of the degree of the polynomial, and the radius of the ball constraints, namely the ratio is $1 - (d+1)!(2d)^{-2d}\,(|\Sigma|+1)^{-\frac{d-2}{2}}(t^2+1)^{-\frac{d}{2}}$. Given that d is assumed to be a fixed constant, thus our result in the theorem. □

H. PROOF OF THEOREM 8

PROOF. We again use the FMLS' formulation. Let $d = \max\{|Q_i|\}$ be the maximum number of events in any query. Monomials in (22) can be homogenized to degree d using the fact that $\sum_{E_j} \bar{x}'_j = 1$. So (22) is equivalent to the following formula that is degree-d homogeneous

$$\sum_{Q_i \in \mathcal{Q}} n_i w_i \Big(\sum_{E_j} (\bar{x}'_j)\Big)^{d-|Q_i|} \prod_{E_j \in Q_i} \frac{M}{\lambda_j m_j} \bar{x}'_j \tag{36}$$

Under the assumption that for all j we have $\frac{\lambda_j m_j}{M} \geq 1$, the problem then is to optimize a degree-d homogeneous polynomial program on the standard simplex $\Delta_{|\Sigma|}$.

Authors in [22] show that using a uniform k-grid, $\Delta_{|\Sigma|}(k) = \{x \in \Delta_{|\Sigma|} | kx \in \mathbb{Z}^+\}$, polynomials of fixed degree d can be approximated by evaluating (22) for all $\binom{|\Sigma|+k-1}{|\Sigma|-1}$ number of points on the k-grid $\Delta_{|\Sigma|}(k)$. Utilizing approximation bound obtained in [22] (see Theorem 1.3 in [22]), FMLS' in this special case can be approximated with a relative approximation ratio $\epsilon = (1 - \frac{k!}{(k-d)!k^d})\binom{2d-1}{d}d^d$. Given that d is assumed to be a fixed constant, we get the result stated in the Theorem. □