Uncertainty Propagation in Complex Query Networks on Data Streams: A New Paradigm for Load Shedding

Kai Zeng Barzan Mozafari Shi Gao Carlo Zaniolo
Computer Science Department
University of California at Los Angeles, California, USA
{kzeng, barzan, gaoshi, zaniolo}@cs.ucla.edu

July 31, 2012

Abstract

Data Stream Management Systems (DSMS) are subject to bursty data arrivals. When overloaded, DSMS have to shed some of their load while minimizing quality of service losses. Thus, in this paper, we introduce an optimal load shedding strategy that minimizes the loss of accuracy in the query results. While previous studies have focused on minimizing the error per each window, in this paper we propose a new load shedding paradigm that models the distribution and propagation of the uncertainty itself, thus giving rise to an average-case error analysis over all windows. This treatment of uncertainty as a distribution enables us to solve the problem of optimal load shedding for the most general case of query networks and query operators. Therefore, our results significantly extend the state-of-the-art on DSMS load shedding, overcoming the significant limitations that all previous approaches encountered in terms of generality and applicability to complex queries. Our analytical conclusions are validated by the results of extensive experiments, which underscore the effectiveness of the new paradigm and techniques over a wide range of overload scenarios, including complex mining tasks on real-world and synthesized data sets.

1 Introduction

Many data-intensive applications need to continuously monitor terabyte to petabyte data streams such as click streams, web logs, and sensor data streams, and continuously produce results. As a result, finding more effective resource management techniques for Data Stream Management Systems (DSMS) has been the focus of much research [2, 3, 5, 7]. Data streams can arrive in bursts, and thus, the DSMS can easily become overloaded. When the system queues are building up due to the overload, the DSMS needs to shed some of the load in order to stay operational. This technique, called *load*

shedding, is the last resort for a DSMS. By skipping the full processing of parts of the input, the DSMS seeks to save the CPU time, memory or other limited resources.

Load shedding causes accuracy loss. The amount of accuracy loss, however, depends on *when, how much and where* the load is shed. Thus, the main challenge is to find load shedding strategies to minimize the degradation of Quality of Service (QoS). In practice, the QoS is often defined in terms of the accuracy of the query results, and occasionally in terms of performance metrics such as the response time or throughput. Finding the optimal load shedding strategy is not trivial. Thus far, numerous proposals have sought optimal load shedding policies. However, the generality of the existing techniques is largely impaired by several strong assumptions and limitations, as discussed next.

Previous work on load shedding can be categorized based on their goal function: (i) maximizing the system performance, or (ii) minimizing the accuracy loss. For example, the performance-based approaches in (i) have focused on maximizing the tuple utility [30] or system throughput [29]. These system-oriented approaches treat the query network as a black-box, and thus, can handle complex queries. However, the black-box approach makes impossible to reason about accuracy loss of queries and to provide the reliable QoS assurances required by critical applications.

On the other hand, the accuracy-based approaches in (ii) have aimed at minimizing metrics such as relative error [4] or the variance of the approximate answer on the sampled data [22, 24]. However, prior work in this category is severely impaired by the limited set of query operators and by the simplicity of the query networks that they can support. For instance, [4, 22, 16, 24] restrict the query networks to only those in which aggregates appear at the leaf level, i.e. the output of an aggregate cannot be fed into any other operator. However, in practice, almost all data mining algorithms are a result of several layers of complex operations including aggregations, comparisons and many other analytical functions.

Another limitation of previous load shedding approaches is that either they do no support user-defined functions and algebraic expressions [4, 22, 16], or they require the users to design and implement their own ad-hoc load shedders [24].

We believe that the inability of all previous approaches to support and provide accurate estimates for complex query operators and networks is primarily due to the error model they have used. Thus, in a sharp departure from previous work, our load shedding approach is based on modeling the distribution of the uncertainty that is caused by load shedding and how this uncertainty propagates throughout an arbitrary query network. At the best of our knowledge, this represents a very new load-shedding paradigm, and its difference from previous approaches can be formally characterized from the discussion that follows. Let D be the current window over the data stream, q a query that we would like to compute over D, and $\hat{D}$ be a random sample from D, i.e. due to load shedding, we have randomly discarded the rest of D. Using this terminology, the goal of the previous work can be stated as minimizing the approximation error *per each window*, or more precisely, ensuring that

$$E_{\hat{D}|D}(q(\hat{D})|D) = q(D) \tag{1}$$

where $E_{\hat{D}|D}$ denotes the expectation averaged over all possible samples given the window. Thus, equation 1 ensures that for every window the approximate answer is unbi-

ased. Our goal, instead, proposes to minimize the average-case error *over all possible windows*, or more precisely, we only require that

$$E_{D,\hat{D}|D}(q(\hat{D})|q(D) = \theta) = \theta \tag{2}$$

where $E_{D,\hat{D}|D}$ denotes the expectation averaged over all possible samples given the window, and also all possible windows. Informally, equation 2 states that the approximate query computed on the sampled data is an unbiased estimator of the original query.

The difference between (1) and (2) is subtle but very important as in this paper we make the following observation: ensuring (1) is not always feasible and can lead to over-allocation of resources while ensuring (2) is not only feasible and resource-efficient but also represents a more realistic approach given the unbounded nature of data streams.

The validity of this statement will become clear after the in-depth analytical treatment in the body of the paper; however, it can also be appreciated at the intuitive level from the following example:

Example 1 (Per-window-case vs. Average-case Error)**.** *Consider a data steam of real-valued tuples and a window size of* 100*. Assume that the current window,* D*, happens to contain* 97 *occurrences of* 1 *and* 3 *occurrences of* 20*, that is,* $t_i = 1$ *for* $i = 1, \cdots, 97$ *and* $t_i = 20$ *for* $i = 98, 99, 100$*. Assume that our query is to find the median of the window. Under load shedding, depending on the size and the content of our sample* $\hat{D}$*, we may get one of the following answers for* $q(\hat{D})$*:* 1*, if more than half of the sampled tuples are* 1*,* 10.5 *if we have the same number of* 1*'s and* 20*'s in* $\hat{D}$*, or even* 20 *if more than half of* $\hat{D}$ *are* 20*'s. In other words,* $q(\hat{D}) \in \{1, 10.5, 20\}$*. Since the* $Pr(q(\hat{D}|D) = 10.5 \text{ or } 20) > 0$*, the expected value of our approximation, namely* $E_{\hat{D}|D}(q(\hat{D})|D)$ *will always be larger than* 1*, and thus, we cannot have an unbiased approximation because the true value of the median for this particular window is* 1*. Thus,* $E_{\hat{D}|D}(q(\hat{D})|D) > q(D)$*.*

While this example presents an extreme case, achieving the goal of equation (2) is still feasible, as per the discussion in Section 3, and the following intuitive explanation. For some windows, such as the one in Example 1, our load shedding may lead to biased answers. But overall, by taking into consideration the probability of D, the biased cases may be very rare, and/or the bias in the answers can be canceled out over all the subsequent windows. For instance, there may be another window, say D', for which the true median is still the same, i.e. $q(D') = 1$, but due to its specific distribution[1], our approximation would lead to an underestimation, i.e. $q(\hat{D'}) < q(D')$. Thus, we could at least hope that, over all windows, our load shedding would achieve: $E_{D,\hat{D}|D}(q(\hat{D})|q(D) = \theta) = \theta$. Therefore, at least in principle, achieving (2) is not impossible, and in fact in this paper, we provide optimal load shedding strategies to deliver this guarantee.

Motivation. Using the new paradigm proposed in this paper, we can solve the most general case of load shedding, without restricting the number of layers, or the types of

[1]For instance, if in D' we have $t_i = 1$ for $i = 1, \cdots, 97$ and $t_i = -18$ for $i = 98, 99, 100$.

aggregates and algebraic functions that are allowed. The following examples illustrate the importance of such extensions.

Example 2 (Network traffic management [1])**.** *The network traffic management applications monitor network traffic and performance across a set of network infrastructures, e.g. routers and switches. Common queries keep track of the most recent traffic of each source IP, and each customer network through a backbone link. Such queries need to aggregate the traffic of each source IP, and group the aggregation result of each source by their corresponding customer networks to find out the total traffic of each network, which involves nested aggregates.*

Previous approaches do not support multiple layers of aggregates, though required in many decision support algorithms. The next example involves a complex function.

Example 3 (Astrophysical survey [28])**.** *Astrophysical surveys generate huge volumes of data every day. For example, the Large Synoptic Survey Telescope* [2] *produces nightly data rates of 20 TB. In these surveys, a kind of query that evaluates marked correlation function (MCF) is of particular interest, which in its simplest form is defined as:*

$$M(r) = \frac{\sum_i \sum_j w_i w_j \mathcal{I}(r_{ij} = r)}{\sum_i \sum_j \mathcal{I}(r_{ij} = r)}$$

where $\mathcal{I} = 1$ *if the separation* r_{ij} *between galaxy* i *and galaxy* j *is* r*, and* $\mathcal{I} = 0$ *otherwise.* w_i *is the luminosity of galaxy* i*. In a DSMS,* $M(r)$ *can simply be implemented as a* `SUM` *aggregate divided by a* `COUNT`*.*

Again, in the presence of complex functions, current load shedders cannot guarantee any QoS. For instance, questions such as 'how much will the accuracy of the MCF query be affected once we shed 70% of the astrophysical data stream?', are simply unanswered by the previous research.

Problem Statement. In this paper, we tackle the problem of optimal load shedding, where:

1. We do not want to restrict the query graphs to only contain aggregates at the leaf nodes. Aggregates can appear anywhere, and can be fed into one another. We also want to allow aggregates to take more than one stream as input.
2. We want to allow arbitrary functions throughout the network as long as they have continuous first-order derivatives.

The extensions above allow for load shedding over most general query networks.

Contributions. In summary:

1. We propose and study a load shedding paradigm that is based on modeling the distribution and propagation effect of the uncertainty.
2. We solve the problem of optimal load shedding for the most general case of query networks and operators. In particular, we provide closed-form formulae for arbitrary combinations of aggregates, rank statistics and arbitrary differentiable functions. We also provide use-cases of our results by considering complex mining tasks such as Bayesian Networks.
3. We empirically validate our theoretical results over both real-world and synthesized data.

[2] LSST. http://www.lsst.org

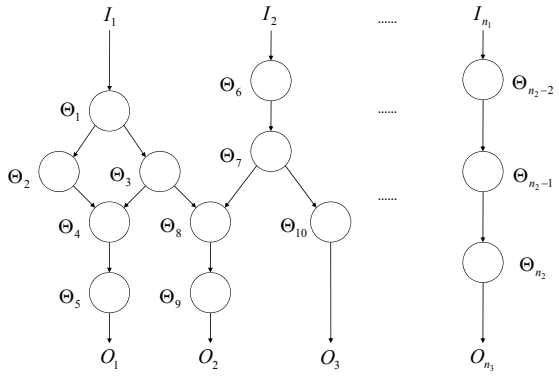


Figure 1: Query Network Diagram

2 Problem Formulation

A typical query network in a DSMS is a *directed acyclic graph* $G(V, \mathcal{E})$, such as the one shown in Figure 1. V is the set of vertices defined as $V = V_s \cup V_q \cup V_o$, where the nodes in V_s represent the incoming data streams $I_1, \cdots, I_{n_1}$, the nodes in V_q represent the query operators $\Theta_1, \cdots, \Theta_{n_2}$, and nodes in V_o are the output streams $O_1, \cdots, O_{n_3}$. The set of directed edges, $\mathcal{E} \subseteq (V_s \cup V_q) \times (V_q \cup V_o)$, represents the data flow between two vertices. For each node v in V, we define its *query subgraph* to be the subgraph induced by the set of all vertices that can reach v, including v itself, and denote it by $G(v)$. This path represents the necessary nodes to compute the results of v. For instance, in Figure 1, $G(\Theta_4)$ is the subgraph induced by $\{I_1, \Theta_1, \Theta_2, \Theta_3, \Theta_4\}$.

We categorize query operators in a DSMS into four types:

1. *Aggregate operators*, defined as a function that maps a set of values into a single value. Examples include the SQL aggregates (i.e., `SUM`, `COUNT` and `AVG`) and order statistics (i.e. quantiles) [3].
2. *Tuple operators*, defined as a function that maps a tuple to a value. Examples include user-defined functions and all algebraic expression defined on the tuple's attributes[4].
3. *Filtering operators*, which drops tuples that mismatch a given condition. An example is the selection operator.
4. *Structural operators*, which affect the structure of the query network. Examples are projection, join, split and union.

Since aggregate and tuple operators both generate new values, we refer to them as *computational operators*.

As data streams are unbounded, in a DSMS, aggregates are often defined over finite portions of the data stream, called *windows*. Windows are either *logical* (e.g., all the tuples arrived in the past 10 minutes), or *physical* (e.g., the last 1000 tuples). A *slide* is defined as the distance between the starting points of two adjacent windows. The slide size can be smaller than or equal to the window size. When they are equal, it is called

[3] In this paper, when referring to SQL aggregates and order statistics, we exclude `MAX` and `MIN` as extreme cases which will be discussed in Sections 3.2.

[4] In this paper, we require that a tuple operator function have continuous first order derivative.

a *tumbling window*. For ease of presentation, in this paper we only present our results on physical and tumbling windows. However, in Appendix C, we show that our results can be easily generalized to other forms of windows.

Load shedding. In a DSMS, the arrival rates of input tuples, as well as the resource usage (CPU, buffer length, etc.) of each query operator, are continuously monitored. When the DSMS is overloaded, that is, when the arrival rate of data exceeds the system's (processing or memory) capacity, the last resort is to discard portions of the input data to cast off the excessive load. In this paper, we limit our scope to the CPU usage problem. This operation, called *load shedding* [30, 4, 31, 22, 24], often consists of two steps: (i) deciding how much and where to shed the load, such that the degradation of a given QoS (Quality of Service) metric is minimized, and (ii) placing *load shedders* on the edges in the query network in order to randomly sample the incoming tuples, based on their derived shedding ratios. In other words, each load shedder is assigned a sampling rate $0 < p \leq 1$ (or a shedding rate of $1 - p$).

Given p, there are two common sampling methods that could be used for load shedding:

Bernoulli sampling For each incoming tuple, the load shedder passes it to the downstream vertex with probability p, and with probability $(1 - p)$ discards it.

Simple random sampling [34] (a.k.a. fixed-size sampling) For a window of size n, we randomly sample exactly $p \cdot n$ tuples from the window, without replacement.

For brevity, throughput the paper we only present our results for Bernoulli sampling. However, the same results hold for fixed-sized sampling (as shown in Section 3.2).

2.1 Problem Statement

Two types of error caused by load shedding. Load shedding can introduce two different types of error in the output results:

(i) Subset error. In the absence of aggregate operators, it is guaranteed that all the output tuples are a subset of the original output tuples. In other words, due to load shedding, some tuples might be missing in the output but at least the ones that are delivered are accurate, i.e. they would be still part of the output if there were no load shedding in place. Thus, in this case, the goal of an optimal load shedding strategy would be to deliver the largest possible subset. This problem, known as *subset error*, has been intensively studied [30, 31, 29, 15].

(ii) Uncertainty error. When the query network contains aggregates, load shedding can also lead to inaccurate query results. For instance, if we shed some of the input to an `AVG` aggregate, the final results is an approximate average which would not necessarily appear in the output if the were no load shedding. Obviously, when aggregates are fed into other operators, this uncertainty will propagate through the query network. The challenge here is to minimize the inaccuracy of our approximate query results, for a user-given error metric. This, known as the *uncertainty error*, leads to a very different paradigm from the subset error model.

This paper addresses general query networks which may contain many complex aggregate operators, and therefore, *we focus on the uncertainty error*. However, our model can also incorporate the subset error through a simple reduction (see Appendix C).

Thus, we define the problem of optimal load shedding as that of optimizing the placement and sampling rates of the load shedders in the query network to minimize the uncertainty error in the query answers (e.g. the variance of the approximation) while keeping the number of such answers the same as in the case of no load shedding,

As a result, in this paper, we exclude from our analysis the cases where a filtering or join operator is applied to the output of an aggregate, because the uncertainty in the input of filtering or join (caused by the previous aggregate), can change the cardinality of their output, and hence lead to subset error.

Load shedding as optimization. In its most general form, optimal load shedding can be formalized as the task of minimizing a weighted error of the approximate answers in the output, once a certain amount of load has to be shed in the input.

Let $\boldsymbol{E} = [e_1, \cdots, e_{n_2}]$ be the error vector[5], where e_i is the error in our approximate results for output stream O_i. Similarly, let $\boldsymbol{V} = [v_1, \cdots, v_{n_2}]$ represent the importance of each O_i. Under a given load shedding scenario, let the resource costs of all query operators be $\boldsymbol{C} = [c_1, \cdots, c_{n_3}]$, where c_i is the unit cost of Θ_i under load shedding. Thus, for a given query network, $\boldsymbol{V}$ is specified by the user, while $\boldsymbol{E}$ and $\boldsymbol{C}$ are both functions of the applied sampling rates, say $\boldsymbol{p} = [p_1, \cdots, p_{|\mathcal{E}|}]$ (i.e., one p_i per edge). Now, the problem of optimal load shedding can be stated [24] as choosing $\boldsymbol{p}$ such that the weighted error (the scalar product of $\boldsymbol{E} \cdot \boldsymbol{V}$) is minimized, subject to a resource constraint, where the overall cost has to be under system's capacity L.

$$\text{minimize} \quad \boldsymbol{E} \cdot \boldsymbol{V} = \sum_{O_i \in V_O} e_i \cdot v_i \tag{3}$$

$$\text{subject to} \quad \sum_{\Theta_i \in V_q} c_i \leq L \ \text{ and } \ \forall j, 0 < p_j \leq 1 \tag{4}$$

3 Uncertainty Propagation

We define the uncertainty (error) of a computational operator as

Definition 1. *The uncertainty of a computational operator Θ is defined as $\epsilon = \hat{\theta} - \theta$, where θ and $\hat{\theta}$ denote the output value of Θ, respectively, before and after load shedders are inserted in $G(\Theta)$.*

We first introduce our superpopulation model of the data streams. In the rest of the section, we present our analytical results on the distribution of the uncertainty and its propagation in the network by each category of operators.

3.1 Models for Data Streams and Uncertainty

Superpopulation Model. In this paper, we model a data stream as a population drawn from a *superpopulation* [6]: we treat each tuple as an *independent instantiation of a random vector* $\mathbf{X} = [X_1, \cdots, X_k]$, where component X_i decides the value of

[5]Throughout this paper we use the bold font $\boldsymbol{A}$ to represent vectors.

the i'th attribute of the tuple. $\mathbf{X}$ follows a joint distribution with probability density function (p.d.f.) $f_{\mathbf{X}}$. We use f_{X_i} to refer to the marginal distribution of X_i. The superpopulation models the distribution of the data stream tuples, taking into account any possible states of the stream in the future. This model allows us to develop feasible and resource-efficient load shedding plans, as shown in Example 1.

In some scenarios, tuples in a stream might be correlated. Inter-tuple correlations are either stationary (i.e. the correlation does not change over time) or non-stationary (i.e., the correlation changes over time) [26]. The stationary inter-tuple correlation has often been ignored [20, 33] due to the stringent performance requirements of DSMSs which do not allow for the costly process of detecting inter-tuple correlations. However, the non-stationary nature of inter-tuple correlation can be treated as a *concept shift*. In other words, one can assume that the inter-tuple correlation is stationary, and each time a concept shift is detected, we can assume a new stationarily correlated data stream with a new joint distribution [6].

Gaussian Mixture Model. A *Gaussian Mixture Model* (GMM) is a weighted sum of Gaussian distributions. Later in this section, we show that (under quite general conditions) query uncertainties can be modeled as GMMs, and for which we can give close form formulation and derive efficient load shedding algorithms.

Definition 2. *A multivariate Gaussian Mixture Model for a random vector $\mathbf{X}$ is a convex mixture of m Gaussian random vectors $\mathbf{X}_1, \cdots, \mathbf{X}_n$. The p.d.f. of $\mathbf{X}$ is $f_{\mathbf{X}}(\mathbf{x}) = \sum_{i=1}^{m} p_i f_{\mathbf{X}_i}(\mathbf{x})$, where $\sum_{i=1}^{m} p_i = 1$ and each $\mathbf{X}_i$ follows a Gaussian distribution with mean $\boldsymbol{\mu}_i$ and covariance matrix Σ_i, abbreviately denoted by $\mathbf{X}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$.*

3.2 Aggregate Operators

We first study the uncertainty of a single aggregate under load shedding which we then, in Section 3.2.1, generalize to horizontal composition of multiple aggregates. The vertical composition of aggregates is discussed in Section 3.4.

Let A_n denote the output of an `AVG` aggregate on a window of size n. Define X as the attribute being averaged. Let $\mathcal{D} = \{X_1, X_2, \cdots, X_n\}$ denote the current window of input values to the `AVG` operator. As described in the superpopulation model, $\mathcal{D}$ is a set of independent and identically distributed (i.i.d.) samples drawn from some underlying distribution $f_{\mathbf{X}}$. We can think of load shedding on the input of `AVG` as a partitioning of $\mathcal{D}$ into two parts: $\mathcal{D}_1$ of size n_1 and $\mathcal{D}_2$ of size n_2, where $\mathcal{D}_1$ is the sampled window and $\mathcal{D}_2$ is the set of tuples discarded.

Let p be the sampling rate. If we use Bernoulli sampling, according to *Large Numbers Theorem* [11], $n_1/n \to p$ and $n_2/n \to (1-p)$. In the case of fixed-size sampling, we have $n_1/n = p$ and $n_2/n = (1-p)$. Therefore, fixed-size sampling can be viewed as a special case of Bernoulli sampling. Hence, in the rest of this paper, we omit fixed-size sampling and only present our analysis based on Bernoulli sampling.

Let $\hat{A}_n^{(1)}$ and $\hat{A}_n^{(2)}$ be the `AVG` computed on $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. Both of $\hat{A}_n^{(1)}$ and $\hat{A}_n^{(2)}$ are approximations of A_n. Clearly,

$$A_n = \frac{n_1}{n}\hat{A}_n^{(1)} + \frac{n_2}{n}\hat{A}_n^{(2)}$$

[6] Incorporating static inter-tuple correlations would make an interesting future extension of our models.

Assume that X has the finite mean μ_X and the variance σ_X^2, as it is always the case in practice given that mean and variance are measured empirically. By applying *Central Limit Theorem* [11], $n_1^{1/2}\hat{A}_n^{(1)}$ and $n_2^{1/2}\hat{A}_n^{(2)}$ are independent and both converge to a Gaussian distribution $\mathcal{N}\left(\mu_X, \sigma_X^2\right)$. Since any linear transformation of a normal distribution is still normal, we can derive the following joint distribution:

$$\begin{bmatrix} A_n \\ \epsilon \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_X \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{\sigma_X^2}{n} & 0 \\ 0 & \frac{1-p}{p}\frac{\sigma_X^2}{n} \end{bmatrix}\right)$$

where ϵ is the uncertainty of A_n. To save space, we use

$$(A_n; \epsilon) \sim \mathcal{N}\left((\mu_X; 0), \left(\frac{\sigma_X^2}{n}; \frac{1-p}{p}\frac{\sigma_X^2}{n}\right)\right)$$

as a compact representation throughout the rest of the paper. From this distribution, we can make the following important observations about the uncertainty of `AVG`:

1. The aggregate result and its uncertainty (i.e., A_n and ϵ) are independent and converge to normal distributions.
2. The mean of ϵ is 0, which implies that the approximate answer $\hat{A}_n^{(1)}$ is an unbiased estimator.
3. The variances of A_n and ϵ have the relationship $\mathbf{Var}(\epsilon) = \frac{1-p}{p}\mathbf{Var}(A_n)$, where p is the sampling rate of the load shedder.

As we show in the rest of this paper, these properties play a fundamental role in our analysis of uncertainty propagation, which make the problem of optimal load shedding tractable even for very complex query networks. Fortunately, these properties hold for a very large class of statistics, including almost all commonly used aggregates. We recognize this class as all statistics that satisfy two conditions: *mean-like* property and *asymptotic normality* [11], which are defined next.

Definition 3 (Mean-like Statistic)**.** *Let θ_n be the output of Θ on $\mathcal{D} = \{X_1, \cdots, X_n\}$, where X_i's are i.i.d. random variables. Consider a random partitioning of $\mathcal{D}$ into k subsets $\mathcal{D}_1, \cdots, \mathcal{D}_k$. Let n_i be the cardinality of $\mathcal{D}_i$ and $\hat{\theta}_n^{(i)}$ be the output of Θ on $\mathcal{D}_i$. We call Θ a mean-like statistic if, when $n_i/n \to \lambda_i$ as $n \to \infty$, we have:*

$$\sum_{i=1}^{k} \lambda_i \hat{\theta}_n^{(i)} \to \theta_n \ , \ \ as \ \ n \to \infty$$

Definition 4 (Asymptotic Normality)**.** *Let θ_n be the output of Θ on $\mathcal{D} = \{X_1, \cdots, X_n\}$, X_i's are i.i.d. random variables with finite mean and variance. Θ is asymptotic normal if*

$$\theta_n \sim \mathcal{N}\left(\mu, n^{-1}\sigma^2\right) \ , as \ \ n \to \infty$$

where μ and σ^2 are some constants.

Next, we observe that all SQL aggregates and order statistics (except `MIN` and `MAX`) are both mean-like and asymptotic normal, and thus have the three aforementioned properties.

`SUM` and `COUNT`. A `SUM` aggregate can be defined as $S_n = nA_n$, and thus, can be approximated as $\hat{S}_n^{(i)} = n\hat{A}_n^{(i)}$. Clearly, `SUM` is a mean-like statistic with asymptotic normality. Note that a `COUNT` aggregate is a special case of `SUM`. A `COUNT` without any selection condition, is trivial as it always returns n (i.e. the size of the window). A `COUNT` that only counts those tuples that satisfy a specific condition, say c, is the same as a `SUM` on a window that is defined as $t_i' = 1$ if c holds for t_i and otherwise $t_i' = 0$.

`Order Statistics`. The *k-th order statistic* of $\mathcal{D}$ is defined as its k'th smallest element, which is also known as the q-th *quantile*, where $k = \lceil nq \rceil$. Let Q_n be the q-th quantile of $\mathcal{D}$, and $\hat{Q}_n^{(i)}$ be the q-th quantile of $\mathcal{D}_i$, where $\mathcal{D}_i$ is one of the partitions defined in Definition 3. It is known [11] that Q_n is asymptotic normal, i.e.

$$Q_n \sim \mathcal{N}\left(x_q, \frac{1}{n}\frac{q(1-q)}{f_X^2(x_q)}\right)$$

where x_q is the q-th quantile of the underlying distribution f_X. Moreover, Knight [21] proved that for $0 < q < 1$, Q_n is a mean-like statistic.

However, `MIN` and `MAX` are extreme order statistics for which, in general, it is very difficult (if not impossible) to find unbiased approximations. Thus, we do not apply load shedding on `MIN` and `MAX`.

In general, for an aggregate operator Θ and its uncertainty under load shedding, ϵ, we have the following theorem [7]:

Theorem 1. *If an aggregate operator Θ is mean-like with asymptotic normality, then under load shedding with the sampling rate p, its approximate answer is unbiased, i.e. $\mathbf{E}(\hat{\theta}|\theta) = \theta$. θ and ϵ are independent and asymptotic normal:*

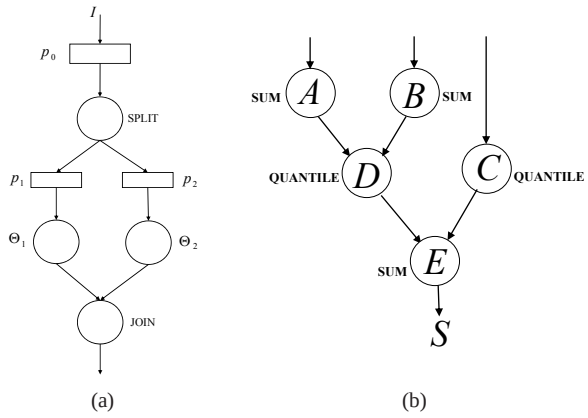
$$(\theta; \epsilon) \sim \mathcal{N}\left((\mu; 0), \left(\sigma^2; \tilde{\sigma}^2\right)\right), \text{ where } \tilde{\sigma}^2 = \frac{1-p}{p}\sigma^2$$

where μ and σ^2 are some constants determined by the aggregate and the superpopulation.

3.2.1 Correlations Between Multiple Aggregates

Multiple aggregates on the same attribute or correlated attributes might have correlations. Under load shedding, such correlations affect the distribution of the uncertainty. For instance, as shown in Example 3, we count pairs of galaxies that are of some distance of each other, and sum up their luminosity products. Let $X = \mathcal{I}(r_{ij} = r)$ and $Y = w_i w_j \mathcal{I}(r_{ij} = r)$. Assume in a window we have p percent non-zero X. If we sample more than p percent non-zero X in the window, we will also have more non-zero Y in the same sample, which means we overestimate both the `SUM` and `COUNT` aggregates. In complex query networks, multiple aggregates and user-defined functions can be fed into other computational operators. Such correlation will propagate through the

[7] All the omitted proofs are provided in Appendix B.



(a) (b)

Figure 2: (a) Correlations between Multiple Aggregates, (b) Placement of Load Shedders.

network and affect downstream operators. However, in the context of load shedding, such important correlations have never been studied before.

Let us consider a simple case where two aggregates Θ_1 and Θ_2 are applied to the same window over the same data stream, as shown in Figure 2(a). Then, we have several alternatives for the placement of the load shedders: (i) **Shared.** Insert a single load shedder right before the split operator, in which case both Θ_1 and Θ_2 are computed on the same sample of the tuples. (ii) **Independent.** Insert a different load shedder right before each aggregate, so that Θ_1 and Θ_2 are computed using two independent samples of the tuples. (iii) **Hybrid.** Insert load shedders right before the split operator and also before each aggregate, in which case the input sets to Θ_1 and Θ_2 are different but correlated. Let the sampling rates in this case be p_0, p_1 and p_2, as shown in Figure 2(a). It is clear that the shared placement is equivalent to a hybrid one where $p_1 = p_2 = 1$, and the independent placement is when we set $p_0 = 1$.

Under the hybrid placement, the current window $\mathcal{D}$ is partitioned into four parts, i.e. tuples used by both Θ_1 and Θ_2, tuples used by one of them, and tuples used by none of them. Θ_1 and Θ_2 computed on each partition are correlated, but are independent between partitions. This idea of partitioning the input into independent partitions is the intuition behind the following theorem that characterizes the joint distribution of the uncertainty for aggregates with correlated inputs.

Theorem 2. *For two aggregates Θ_1 and Θ_2 under a hybrid load shedding plan with the shared sampling rate p_0 and the independent sampling rates p_i for Θ_i, their results $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$ and uncertainties $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2]^T$ are independent and asymptotic normal:*

$$(\boldsymbol{\theta}; \boldsymbol{\epsilon}) \sim \mathcal{N}\left((\boldsymbol{\mu}; 0), \left(\Sigma; \widetilde{\Sigma}\right)\right)$$

where $\boldsymbol{\mu}$, Σ are some constants (resp., a vector and a matrix), and

$$\widetilde{\Sigma}_{ij} = \frac{1 - p_{ij}}{p_{ij}} \Sigma_{ij} \;\; \textit{where for} \;\; 1 \leq i, j \leq 2, \;\; p_{ij} = \begin{cases} p_0 p_i, & \textit{if } i = j \\ p_0, & \textit{if } i \neq j \end{cases}$$

Theorem 2 can be viewed as a high-dimensional generalization of Theorem 1. The importance of Theorem 2 becomes clearer when we make the following interesting observations.

Observation 1. *In a hybrid placement, the effective sampling rate for each aggregate is the product of the sampling rates of the shared load shedder and the corresponding independent load shedder. For instance, in Figure 2(a), the effective sampling rate for* Θ_1 *is* $p_0 p_1$*. Therefore, the variance of* Θ_1*'s uncertainty is* $\widetilde{\Sigma}_{11} = \dfrac{1 - p_0 p_1}{p_0 p_1} \Sigma_{11}$.

Observation 2. *In a hybrid placement, the effective sampling rate for each pair of aggregates is the sampling rate of the shared load shedder. For instance, in Figure 2(a), the effective sampling rate for* Θ_1 *and* Θ_2 *is* p_0*. Therefore, the covariance of their uncertainties is* $\widetilde{\Sigma}_{12} = \dfrac{1 - p_0}{p_0} \Sigma_{12}$.

In some scenarios, various aggregates have different window sizes, causing their input windows to overlap partially. Often, the overlap pattern is also varying with time. For instance, if Θ_1 and Θ_2 use windows of size 5000 and 7000, respectively, the overlap between their inputs could range from 1000 to 5000 tuples. To deal with such scenarios, for each overlapping pattern, we can model the aggregates' results $\boldsymbol{\theta}$ and their uncertainty $\boldsymbol{\epsilon}$ as a Gaussian distribution. Thus, the final distribution will be a GMM, where each Gaussian distribution comes from a different overlapping pattern.

3.3 Tuple Operators

As discussed in Section 2.1, tuple operators, by themselves, do not introduce uncertainty error. However, when load shedding is applied on the upstream operators in a tuple operator's query subgraph, the uncertainty in the tuple operator's input will propagate to its output. Returning to Example 3, if load shedders are applied on the `SUM` and `COUNT` aggregates, the output of MCF function will also contain uncertainty. In this section, we study the propagation of uncertainty from the input of a tuple operator to its output.

Consider a tuple operator $\Theta = \psi(t)$, where ψ is a function with continuous first-order derivative, and t is a tuple composed of several attributes. Attribute can be virtual, i.e. be the result of another aggregate or tuple operator. We divide the attributes of t into two categories: (i) deterministic attributes, modeled by $\mathbf{X}^d$, which have no uncertainty under load shedding, and (ii) nondeterministic attributes, modeled by $\mathbf{X}^u$, which have uncertainties when load shedding is applied in $G(\Theta)$. Thus, each t is an instantiation of $\mathbf{X}^d \cup \mathbf{X}^u$. For instance, Wal-mart might want to estimate its daily net operating profit after tax, in which case $\mathbf{X}^u$ could be the total daily revenue from all Wal-mart's store in a state, while $\mathbf{X}^d$ could be the state-specific tax rate.

We employ a *Conditional View* to facilitate a closed-form modeling of ψ and its uncertainty ϵ_ψ. Below, we formally state the definition of conditional view.

Definition 5. *For a tuple composed of attribute* $(\mathbf{X}^d, \mathbf{X}^u)$*, where* $\mathbf{X}^d$ *is a vector of deterministic attributes, and* $\mathbf{X}^u$ *is a vector of nondeterministic attributes, a conditional*

view of $\mathbf{X}^u$ and its uncertainty $\boldsymbol{\epsilon}$ given $\mathbf{X}^d$, denoted by $V(\mathbf{X}^u, \boldsymbol{\epsilon}|\mathbf{X}^d)$, is the collection of all conditional distributions of $\mathbf{X}^u$ and $\boldsymbol{\epsilon}$ for all values of $\mathbf{X}^d$, characterized by

$$\{f_{\mathbf{X}^u,\boldsymbol{\epsilon}}(\mathbf{x}^u, \boldsymbol{\epsilon}|\mathbf{X}^d = \mathbf{x}^d), \forall \mathbf{x}^d\}$$

By utilizing $V(\mathbf{X}^u, \boldsymbol{\epsilon}|\mathbf{X}^d)$, the propagation effect of tuple operators on uncertainty can be evaluated in two steps:

Step 1. For each $\mathbf{x}^d$, we use the following theorem, to map the GMM f_i of $\mathbf{X}^u$ and $\boldsymbol{\epsilon}$ into another GMM f_o. In each Gaussian distribution from f_i, $\mathbf{X}^u$ and $\boldsymbol{\epsilon}$ are independent. This independence is kept through the mapping. Therefore, in each Gaussian distribution from f_o, ψ and ϵ_ψ are still independent.

Theorem 3. *If $(\mathbf{X}^u; \boldsymbol{\epsilon}|\mathbf{x}^d) \sim \mathcal{N}\left((\boldsymbol{\mu}; 0), (\Sigma; \widetilde{\Sigma})\right)$, then $\psi(\mathbf{x}^u; \mathbf{x}^d)$ and its uncertainty ϵ_ψ are independent and asymptotic normal,*

$$(\psi(\mathbf{x}^u; \mathbf{x}^d); \epsilon_\psi) \sim \mathcal{N}\left((\psi_0; 0), (\nabla\psi_0^T \Sigma \nabla\psi_0; \nabla\psi_0^T \widetilde{\Sigma} \nabla\psi_0)\right)$$

where $\psi_0 = \psi(\boldsymbol{\mu}; \mathbf{x}^d), \nabla\psi_0 = \nabla\psi(\boldsymbol{\mu}; \mathbf{x}^d)$

Theorem 3 can be easily proved using Cramér's Theorem [11], and thus the proof is omitted here.

Step 2. We group the GMMs of $\psi(\mathbf{x}^u; \mathbf{x}^d)$ and ϵ_ψ, for all values of $\mathbf{x}^d$, into a new GMM. Assume $\mathbf{X}^d$ has possible values $\{\mathbf{x}_j^d, j = 1, \cdots, k\}$. For each $\mathbf{x}_j^d$, $\psi(\mathbf{x}^u; \mathbf{x}^d)$ and ϵ_ψ follow a GMM of m_j Gaussian distributions, each with weight w_{i_j} characterized by $\mathcal{N}\left((\boldsymbol{\mu}_{i_j}; 0), (\Sigma_{i_j}; \widetilde{\Sigma}_{i_j})\right)$. Then $\psi(\mathbf{x}^u; \mathbf{x}^d)$ and ϵ_ψ follow a GMM with all these $\sum_{j=1}^{k} m_j$ Gaussian distributions, each with weight $(f_{\mathbf{X}^d}(\mathbf{x}_j^d) w_{i_j})$.

3.4 Aggregates Cascading

In this section, we study the uncertainty propagation when aggregates are arranged vertically, i.e. the output of an aggregate (possibly transformed by other operators in between) feeds into another. The uncertainty in the input of an aggregate Θ causes uncertainty in its output. Let the input of Θ without uncertainty be $\mathcal{D} = \{X_1, \cdots, X_n\}$. Then, its input with uncertainty can be represented as $\mathcal{D}' = \{X_1 + \varepsilon_1, \cdots, X_n + \varepsilon_n\}$. If we furhter apply load shedding to Θ's input, namely $\mathcal{D}'$, we are actually computing Θ on a sample of $\mathcal{D}'$, say $\hat{\mathcal{D}}'$.

In this case, the uncertainty of Θ is $\epsilon_\theta = \Theta(\hat{\mathcal{D}}') - \Theta(\mathcal{D})$, which originates from two different sources: (i) the uncertainty propagated from the query path, and (ii) the uncertainty introduced by load shedding on Θ's input. For the most general set of aggregates, $\Theta(\hat{\mathcal{D}}')$ is not always an unbiased estimator of $\Theta(\mathcal{D})$. However, for the set of aggregates which are also *linear statistics*, $\Theta(\hat{\mathcal{D}}')$ is guaranteed to be an unbiased estimator of $\Theta(\mathcal{D})$. This set of aggregates subsume SQL aggregates. The following theorem states the uncertainty propagation in vertical arrangement of aggregates by using $\Theta =$ `AVG` as an example.

Definition 6 (Linear Statistics)**.** *A linear statistic of n i.i.d. observations $\{X_1, \cdots, X_n\}$ is a linear function $\sum_{i=1}^{n} c_i X_i$, where c_i are constants.*

Theorem 4. *For an* AVG *aggregate* Θ *under load shedding with the sampling rate* p*. If the input of* Θ *and its uncertainty (denoted as* $\mathbf{X}$ *and* $\boldsymbol{\epsilon}$*, respectively) follow a GMM of* m *Gaussian distributions, each with weight* w_i *characterized by* $\mathcal{N}\left((\mu_i;0),(\sigma_i^2;\tilde{\sigma}_i^2)\right)$*. then* θ *and* ϵ_θ *are independent and asymptotic normal, i.e.* $(\theta;\epsilon_\theta) \sim \mathcal{N}\left((\mu;0),(n^{-1}\sigma^2;n^{-1}\tilde{\sigma}^2)\right)$*, where*

$$\mu = \sum_{i=1}^{m} w_i\mu_i \ , \ \sigma^2 = \sum_{i=1}^{m} w_i\sigma_i^2 + \left(\sum_{i=1}^{m} w_i\mu_i^2 - \mu^2\right),$$
$$\tilde{\sigma}^2 = \frac{1-p}{p}\sigma^2 + \frac{1}{p}\sum_{i=1}^{n} w_i\tilde{\sigma}_i^2$$

Theorem 4 can be proved by following the fact that X_i's are independent from ϵ_i's, and applying Theorem 1 on X_i's and ϵ_i's separately. Generalization of Theorem 4 to SUM and COUNT is straightforward and is omitted here. However, Theorem 4 is not applicable to order statistics, as order statistics are not linear statistics. The following example gives an intuitive view why computing order statistics on uncertain data will give out biased results.

Example 4. *Consider the case of computing the* 0.95*-quantile of the output stream of an* AVG*. Assume the output of* AVG *follows Gaussian distribution* $\mathcal{N}(0,1)$*. The* 0.95*-quntile would be around 2. If we shed half of* AVG*'s input, the output of* AVG *would instead follow* $\mathcal{N}(0,2)$*. In this case the* 0.95*-quntile would be around 2.82, which is biased compared to 2.*

This leads us to an important observation,

Observation 3. *Load shedding cannot be applied on other aggregates in the query subgraph of an order statistics aggregate.*

For example, in Figure 2(b), we cannot apply load shedding on A or B, because A and B are in the query subgraph of some order statistics aggregate (i.e. D). However, one can apply load shedding on C, D and E, as they are not in any order statistics aggregate's query subgraph.

3.5 Filtering and Structural Operators

We next consider the effect of filtering and structural operators on uncertainty propagation. Since projection and split are straightforward, we focus on filtering and join (when applied to deterministic attributes), as well as union.

Filtering. Let t be a tuple drawn from a joint distribution $f_{\mathbf{X}^d,\mathbf{X}^u}$. A filtering operator applies a range predicate $\mathcal{I}$ to the deterministic attributes $\mathbf{X}^d$. Let $\bar{t}$ denote the output tuple. Therefore, the distribution of $\bar{t}$ is:

$$\bar{f}_{\mathbf{X}^d,\mathbf{X}^u}(\mathbf{x}^d,\mathbf{x}^u) = \begin{cases} 0, & \text{if } \mathcal{I}(\mathbf{x}^d) = 0 \\ f_{\mathbf{X}^d,\mathbf{X}^u}(\mathbf{x}^d,\mathbf{x}^u)/q, & \text{if } \mathcal{I}(\mathbf{x}^d) = 1 \end{cases}$$

where $q = \int_{\mathcal{I}(\mathbf{x}^d)=1} f_{\mathbf{X}^d,\mathbf{X}^u}(\mathbf{x}^d, \mathbf{x}^u)$.

This shows that applying a filtering operator to a data stream yields a truncated distribution. Thus, our model of uncertainty propagation can be generalized to filtering, simply by applying our previous results on the truncated distribution.

Join. Given two independent streams S_1 and S_2, which are composed of attributes $(\mathbf{X}_1^d, \mathbf{X}_1^u)$ and $(\mathbf{X}_2^d, \mathbf{X}_2^u)$ respectively. $\mathbf{X}_1^d$,$\mathbf{X}_2^d$ are deterministic attributes, while $\mathbf{X}_1^u$,$\mathbf{X}_2^u$ are nondeterministic attributes. Consider a join of S_1 and S_2 based on the deterministic attributes $S_1 \bowtie_{\mathbf{X}_1^d,\mathbf{X}_2^d} S_2$. The join condition can be either equality or inequality comparison. We use conditional views $V(\mathbf{X}_1^u, \boldsymbol{\epsilon}_1|\mathbf{X}_1^d)$ and $V(\mathbf{X}_2^u, \boldsymbol{\epsilon}_2|\mathbf{X}_2^d)$ to facilitate computing the distribution of the join output. We define a join operation between two conditional views as follows,

Definition 7. *$V(\mathbf{X}_1^u, \boldsymbol{\epsilon}_1|\mathbf{X}_1^d) \bowtie V(\mathbf{X}_2^u, \boldsymbol{\epsilon}_2|\mathbf{X}_2^d)$ yields a new conditional view $V(\mathbf{X}_1^u, \boldsymbol{\epsilon}_1, \mathbf{X}_2^u, \boldsymbol{\epsilon}_2|\mathbf{X}_1^d, \mathbf{X}_2^d)$, where*

$$\begin{aligned} &f_{\mathbf{X}_1^u,\boldsymbol{\epsilon}_1,\mathbf{X}_2^u,\boldsymbol{\epsilon}_2}(\mathbf{x}_1^u, \boldsymbol{\varepsilon}_1, \mathbf{x}_2^u, \boldsymbol{\varepsilon}_2|\mathbf{X}_1^d = \mathbf{x}_1^d, \mathbf{X}_2^d = \mathbf{x}_2^d) \\ &= f_{\mathbf{X}_1^u,\boldsymbol{\epsilon}_1}(\mathbf{x}_1^u, \boldsymbol{\varepsilon}_1|\mathbf{X}_1^d = \mathbf{x}_1^d) f_{\mathbf{X}_2^u,\boldsymbol{\epsilon}_2}(\mathbf{x}_2^u, \boldsymbol{\varepsilon}_2|\mathbf{X}_2^d = \mathbf{x}_2^d) \end{aligned}$$

The join result distribution can be evaluated in two steps:
Step 1. Compute $V(\mathbf{X}_1^u, \boldsymbol{\epsilon}_1|\mathbf{X}_1^d) \bowtie V(\mathbf{X}_2^u, \boldsymbol{\epsilon}_2|\mathbf{X}_2^d)$.
Step 2. Group $V(\mathbf{X}_1^u, \boldsymbol{\epsilon}_1|\mathbf{X}_1^d) \bowtie V(\mathbf{X}_2^u, \boldsymbol{\epsilon}_2|\mathbf{X}_2^d)$ into a mixture distribution. Each $f_{\mathbf{X}_1^u,\boldsymbol{\epsilon}_1,\mathbf{X}_2^u,\boldsymbol{\epsilon}_2}(\mathbf{x}_1^u, \boldsymbol{\varepsilon}_1, \mathbf{x}_2^u, \boldsymbol{\varepsilon}_2|\mathbf{X}_1^d = \mathbf{x}_1^d, \mathbf{X}_2^d = \mathbf{x}_2^d)$ is weighted by $f(\mathbf{x}_1^d, \mathbf{x}_2^d)$, the probability of joining $\mathbf{x}_1^d$ and $\mathbf{x}_2^d$ together.
Clearly, the join result distribution is also a GMM.

Union. A union operator merges k streams $S_i, i = 1, \cdots, k$ that all conform to the same schema. We define the *mixture weight* of S_i, denoted by ϕ_i, as the portion of tuples in the merged stream that come from S_i. Let f_i be the distribution of the tuples in stream S_i. Then, the tuples in the merged stream follow a distribution which is a mixture of all f_i, each weighted by ϕ_i.

4 Optimal Load Shedding

In this section, we present our algorithm for generating the load shedding plan. We formulate the problem of finding the optimal load shedding plan into a signomial program.

4.1 Practical Condition for Model Accuracy

Our Gaussian mixture model for uncertainty propagation is based on the asymptotic behavior of the uncertainties. The theoretical distribution predicted by our model approaches the true distribution of the uncertainty with enough accuracy only when the model is applied on a sufficiently large input. For example, Theorem 1 holds for an aggregate when the size of its input window is sufficiently large (e.g. larger than 30).

We evaluate the model accuracy by using the distance between the theoretical distribution predicted by our model and the true distribution of the uncertainty. We use the *Kolmogorov-Smirnov (KS) distance* as a metric to measure this distance defined next.

Definition 8 (Kolmogorov-Smirnov Distance)**.** *[10] The Kolmogorov-Smirnov distance of two one-dimensional c.d.f. F and G is defined as $D_{KS}(F,G) = \sup_x |F(x) - G(x)|$.*

We dynamically determine whether our Gaussian mixture model satisfies the accuracy requirements by conducting *Kolmogorov-Smirnov (KS) Tests* [10]. KS test is a well-known method for testing whether a set of observations are coming from a hypothesized distribution, which is based on the KS distance between the empirical distribution of the observations and the hypothesized distribution. Specifically, for each query in the query network, users can specify a significance level α. Then for each operator Θ in the query subgraph of the query output, we conduct a KS test on the approximate query output $\hat{\theta}$ at the significance level α. The null hypothesis is that $\hat{\theta}$ follows the GMM predicted by our model. The alternative hypothesis is that $\hat{\theta}$ does not follow such distribution. If the KS test rejects the null hypothesis, we simply take out all the load shedders in the query subgraph of Θ.

4.2 Parameter Estimation

As discussed in Section 3, the uncertainties in a complex query network can be modeled as GMMs. Each Gaussian distribution in a GMM is fully characterized by its mean and variance. Thus, the error e_i of output stream O_i is a function of the means and variances of all the Gaussian distributions in O_i's GMM. To formulate the load shedding problem into an optimization problem, we need to estimate these means and variances as input parameters of the load shedding problem.

Aggregates. As shown in Section 3.2 (more detailed discussion can be found in Appendix A), for a SQL aggregate, the distribution of its output is determined by the mean and variance of its input, which can be simply estimated by computing the first and second order moments (i.e., $\mathbf{E}(X), \mathbf{E}(X^2)$) of the input data; for an order statistics aggregate, such as the q-th quantile, the distribution of its output depends on the probability density of the input at the q-th quantile point, which can be estimated by the kernel density estimation method [35].

Tuple Operators, Filtering and Join. Conditional views are essential to facilitate the modeling of tuple operators and join. A joint multi-dimension histogram of deterministic and nondeterministic attributes can be used to build the conditional views. Such multi-dimension histograms are also necessary for a filtering operator in order to compute the truncated distribution. There is much research on developing optimal summary histograms, both for one-dimensional and multi-dimensional data [19, 17, 9]. Our proposed techniques are independent from the specific estimating techniques, i.e. one can plug in better estimating techniques to yield higher accuracy. Moreover, for tuple operators, the derivative of its tuple function at a specific point can be estimated by numerical differentiation [23].

Union. For union operators, the mixture weights of input streams need to be collected.

Every operator Θ_i is characterized by its selectivity s_i and its processing CPU time t_i per tuple. An operator's selectivity is defined as the ratio of the number of results produced by the operator to the number of tuples consumed by the operator before load shedding. Also every input stream I_i is associated with its tuple arriving rate r_i. All the s_i's, t_i's and r_i's are periodically monitored.

4.3 Optimal Load Shedding Planning

Placement of Load Shedders. A load shedder can be treated as a special 'filtering' operator. As a result, pushing a load shedder 'upstream' – that is, more closer to the input stream – across a filtering/tuple operator doesn't change the uncertainties of 'downstream' operators. This 'early shedding' drops tuples as early as possible and saves the unnecessary computation. Therefore, we introduce the concept of *segments* as directed edges in the query network after we take out all the fitering/tuple operators. Clearly, we only need to put a single load shedder at the beginning of each segment.

However, for a segment whose tail is a join/union operator, we do not place any load shedder. This is because shedding the input of join may overly reduce the join output rate [15]. Also, shedding the input streams of a union operator separately, might change their mixture weights.

Optimization Formulation. We model the uncertainty by its distribution, which gives users much flexibility in choosing an error metric of their choice in the error vector $\boldsymbol{E}$, such as variance, absolute error, relative error, etc. We first present our formulation for minimizing variances of uncertainties. For using other error metrics, users could follow similar approaches and thus the formulation is omitted here. Below we use the words 'error' and 'variance' interchangeably. We show that the optimal load shedding can be formulated into a signomial program, which has been intensively studied and has efficient algorithms to find the global optimal solution [36, 25].

We first introduce the definition of a signomial function.

Definition 9. *A signomial function of $x_1, x_2, \cdots, x_n$ is a function with the form*

$$f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{M} \left(c_i \prod_{j=1}^{n} x_j^{a_{ij}} \right)$$

To formulate the error e of an output stream O, users can apply our theoretical analysis shown in Section 3 recursively on each operator along $G(O)$. We have Lemma 1, showing that the error $e(\Theta_i)$ of a computational operator Θ_i is a signomial function of the sampling rates of the load shedders in $G(\Theta_i)$.

Lemma 1.

$$e(\Theta_i) = \sum_{P_j \in \mathcal{P}(G(\Theta))} \left(\frac{a_j}{\prod_{p_k \in P_j} p_k} - a_j \right)$$

where $\mathcal{P}(G(\Theta_i))$ is a subset of the powerset of all the sampling rates of load shedders in $G(\Theta_i)$ and a_j's are constants.

Under our uncertainty error model, an aggregate is guaranteed to deliver the same number of results as the case with no load shedding. Therefore, for an operator Θ, only those load shedders which have a path to Θ with no aggregates in between will affect the effective sampling rate of Θ. We denote this set of load shedders as $LS(\Theta)$. We have Lemma 2 showing that the cost $c(\Theta_i)$ of an operator Θ_i is a signomial function of the sampling rates of the load shedders in $G(\Theta_i)$.

Lemma 2.

$$c(\Theta_i) = t_i \sum_{I_j \in V_I} \left[r_j \sum_{P \in P_{I_j}(G(\Theta_i))} \left(\prod_{\Theta_k \in P} s_k \prod_{LS_k \in P \cap LS(\Theta_i)} p_k \right) \right]$$

where $P_{I_j}(\Theta_i)$ is the set of pathes in $G(\Theta_i)$ each of which connects input stream I_j and Θ_i; LS_k's are load shedders.

The proof for this lemma is straightforward, and thus is omitted.

The fact that $\boldsymbol{E} \cdot \boldsymbol{V}$ and $\sum_{\Theta_i \in V_q} c_i$ are signomial functions directly follows from Lemma 1 and 2. Therefore, the optimal load shedding can be formulated into a signomial program.

5 Extension to Mining Algorithms

Our models for uncertainty of different operators can be easily combined to provide optimal load shedding for complex mining algorithms. In this section, we use Bayesian network as a concrete example to show how our models deliver optimal quality of mining results under limited computational resources.

A Bayesian network is a pair (G, Θ), where G is the network structure, and Θ is the network parametrization [8]. In G, each node represents a variable from the application domain. If variable X is dependent on a set of other variables, say $\mathbf{U}$, then there is a direct edge in G from every node corresponding to a variable in $\mathbf{U}$ to the node corresponding to X. In this case, $X\mathbf{U}$ is called a network family. The network parametrization, Θ, consists of the conditional probability distributions of all the network families in G, i.e. the conditional probability $\mathbf{Pr}(X = x|\mathbf{U} = \mathbf{u})$ ($\mathbf{U}$ can be $\emptyset$). Estimating Θ for a given network structure and a given dataset is one of the central tasks in learning a Bayesian network. A well-known estimation method is the maximum likelihood approach, which uses the empirical distribution $\mathbf{Pr}_{\mathcal{D}}(x|\mathbf{u})$ as the estimate of $\mathbf{Pr}(x|\mathbf{u})$. The empirical marginal/conditional distribution is defined as

$$\mathbf{Pr}_{\mathcal{D}}(x) = \frac{\mathcal{D}\#(x)}{|\mathcal{D}|}, \mathbf{Pr}_{\mathcal{D}}(x|\mathbf{u}) = \frac{\mathcal{D}\#(x, \mathbf{u})}{\mathcal{D}\#(\mathbf{u})},$$

where $\mathcal{D}\#(\alpha)$ is the frequency of α in dataset $\mathcal{D}$. Therefore, the maximum likelihood approach consists of counting the various instantiations of every family $X\mathbf{U}$ and various instantiations of variables $\mathbf{U}$, and finally applying the division function on the counts, say $f(a, b) = a/b$.

Consider a DSMS that is periodically learning and updating a Bayesian network. When the system undergoes a data burst, load shedding becomes necessary. As a result, the `COUNT` aggregates produce approximate results, and thus, degrade the quality/accuracy of the learned Bayesian network. The maximum likelihood approach seeks to maximize the log-likelihood of the learned network. Therfore, in order to maximize the quality of the network, one needs to minimize the uncertainty of the log-likelihood function [8]:

$$LL(\Theta|\mathcal{D}) = \sum_{X\mathbf{U}} \sum_{x\mathbf{u}} \mathbf{Pr}_{\mathcal{D}}(x\mathbf{u}) \log_2 \mathbf{Pr}_{\mathcal{D}}(x|\mathbf{u})$$

Using the models derived in this paper, we know that the uncertainty ϵ of all the COUNT aggregates follow Gaussian distributions, say $\mathcal{N}(0, T)$, where T is a function of the shedding rates. Denote $LL(\Theta|\mathcal{D})$ as a function of the COUNT results, say h. Based on our models, we know that its uncertainty will follow

$$\epsilon_{LL} \sim \mathcal{N}\left(0, \nabla h^T T \nabla h\right)$$

Therefore, by minimizing the variance of ϵ_{LL}, we can find the optimal load shedding plan even for this complex mining task. The load shedding plan generated by our model will apply different shedding rates to different families, according to their susceptibility to shedding. A naive load shedding approach would apply the same shedding rate to all of the COUNT aggregates, uniformly. In Section 6, we have implemented our optimal load shedding models on Bayesian networks. Our empirical results further confirm the accuracy and effectiveness of our theoretical guarantees.

6 Experiments

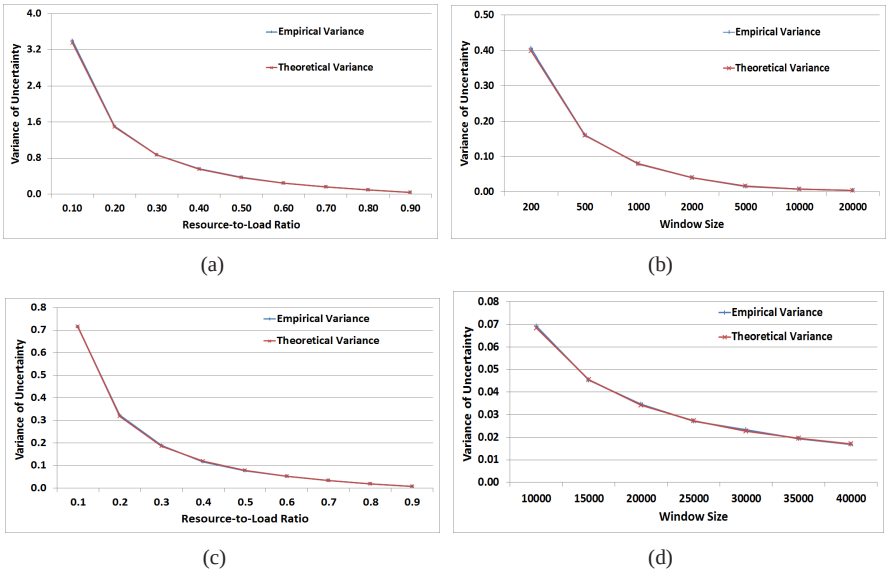


Figure 3: (a) AVG uncertainty vs. resource/load ratio. (b) AVG uncertainty vs. window size. (c) Quantile uncertainty vs. resource/load ratio. (d) Quantile uncertainty vs. window size.

In this section, we present our empirical results, validating (i) the accuracy of our analysis model, and (ii) the effectiveness of our optimal load shedding algorithm. We conducted all the experiments in MatLab, running on a Windows Machine with AMD Athlon 2.20 GHZ CPU and 2 GB memory. The optimization problems were solved using the cvx package [18]. We used both synthetic and real-world data sets.

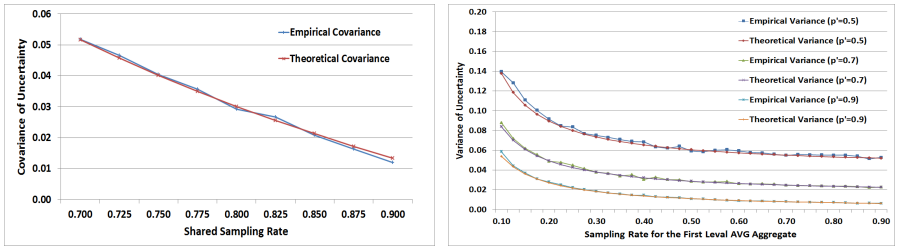
In Section 6.1, we compare the theoretical results predicted by our analysis model against the empirical results obtained from both synthetic and real-world data sets for

various query operators and network structures. Then in Section 6.2, we examine our optimal load shedding technique for the case of an OLAP query, using real-world data. Finally, in Section 6.3, we apply our optimal load shedding technique on learning Bayes networks as an example of complex mining algorithms.

6.1 Model Accuracy

The first set of our experiments study the accuracy of our proposed model for two aggregate operators using `AVG` and 0.75-qantile as two examples. We used real-world data (adult data from UCI data repository [12]) and synthetic data with controlled distributions to study the accuracy of our uncertainty model for these two aggregates, respectively. Our data generator produced a tuple stream with a single attribute. Each tuple is modeled by a mixture of two Gaussian distributions. The means of the two components are uniformly sampled from $[0, 5]$ and $[5, 50]$, and their standard deviation uniformly sampled from $[0.5, 1]$ and $[0, 1]$, respectively to model complex real-world distributions.

Figure 3(a) and Figure 3(c) demonstrate how different sampling rates of load shedding impact the empirical and theoretical distributions by plotting the variances of the distributions against different resource/load ratios. We fix the window sizes to be 500 and 20000 respectively, and let resource/load ratio changing within the range $[0.1, 0.9]$. The lines represent the variances of the empirical distribution obtained by experiments and the theoretical distribution predicted by our model. As expected, the theoretical values perfectly match the empirical experiments, which also implies that the variance of aggregate uncertainty is proportional to the odd of the shedding rate (odd is defined as $f(x) = (1 - x)/x$). Figure 3(b) and Figure 3(d) show the trend of the empirical and theoretical distributions changing with respect to different window sizes. We fixed the sampling rate of load shedding at 0.7 and let the window size change from 200 to 20000. It is clear that the empirical values coincide with the theoretical results. Therefore, the variance of aggregate uncertainty is reciprocal to the window size. In particular, the relative error of the theoretical variances predicted by our model is less than 2.7%.



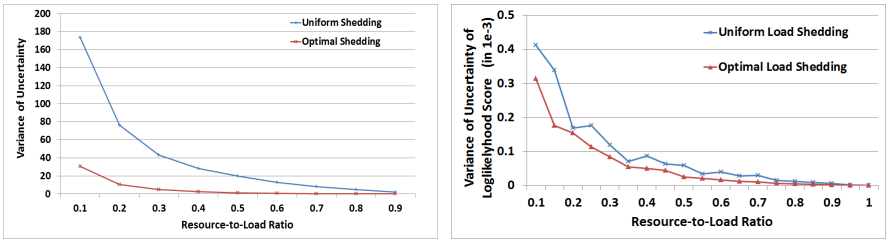
(a) (b)

Figure 4: (a) Correlation between multiple aggregates. (b) Uncertainty in aggregates cascading.

The second set of experiments evaluate the accuracy of our model for complex query network structures. We use synthetic data in this experiment. The two streams are sampled by a method similar to the one used in the first set of experiment but with

different parameter settings. The first experiment investigates our model accuracy of predicting aggregates correlation in the hybrid load shedding setting. We compute AVG on two different but correlated attributes of the same stream with the shared sampling rate ranging in $[0.7, 1]$. We fix the effective sampling rate for each AVG to 0.7. Figure 4(a) plots the covariances of the empirical and theoretical joint distributions of the uncertainties of the two AVG aggregates versus different shared sampling rates. The second experiment studies our model accuracy in the case of multiple cascading aggregates. We compute AVG on two different streams, union the two output streams with mixture weights $0.6 : 0.4$, and evaluate AVG on the merged stream. The window sizes of all the AVG aggregates are 200. We use sampling rate p for the AVG aggregates on two different streams, and p' for the AVG on the merged stream. Figure 4(b) shows the trend of the empirical and theoretical distribution of the uncertainty of the final output against p. Experimental results for different p' settings are demonstrated in Figure 4(b) using different lines. The theoretical results predicted by our model accurately coincide with the empirical values.

6.2 Effectiveness of Optimal Load Shedding



(a) (b)

Figure 5: (a) Optimal laod shedding in OLAP function. (b) Optimal laod shedding in learning Bayes network.

This section studies the efficiency of applying our model in real-world load shedding problem, such as computing an OLAP function. The first goal of our experiments is comparing our load shedding algorithm with the state-of-the-art counterpart.

We use US Census data set from UCI repository [12]. We evaluate an OLAP query by computing the average personal income for each state and then feeding these per-state average incomes into another query to compute the national average personal income. Our aim is to optimize the total variance of both state and national average personal incomes. Figure 5(a) shows the comparison of our algorithm with a uniform shedding plan which is the state-of-the-art available algorithm. Our algorithm improves the accuracy by approximately 8 times compared to the naive approach.

6.3 Mining Algorithm

As an example of a complex data mining task studied the performance of load shedding in Bayesian networks. This also serves as an example of the effectiveness of our optimal load shedding algorithm for complex functions. We used nursery data from

UCI data repository [12] and used Weka [37] to learn the Bayes network structure. We measured the uncertainty of the log-likelihood of the learned network parametrization under load shedding for a wide range of sampling rates. The counterpart algorithm used uniform shedding rate for the `COUNT` aggregates, while our algorithm used a different shedding rate for different families. As shown in Figure 5(b), our algorithm significantly outperforms our counterpart. For highly overloaded situations, e.g. resource/load ratio being 10%, our algorithm improves the counterpart by 25%. It exhibits that by exploring the correlations between different aggregate operators and the first-order derivative of the tuple operator function, one can greatly improve the accuracy of the query, even for complex data mining tasks.

7 Related Work

Researchers have proposed many techniques and approaches to perform load shedding in a DSMS. Many of these works, as well as this work, focus on the centralized server model [30, 4, 31, 22, 24]. However, load shedding proposals have also been presented to address the placement of filters at source nodes [16, 13], as well as load shedding in a distributed stream processing environment [29].

An early load shedding proposal presents an approach to load shedding in a DSMS that focuses on dropping both random tuples and tuples of lesser utility for operators that do not generate new values (filter, union, and join) [30]. Load shedding in windowed join queries has also been examined [14, 15]. Rather than load shedding by dropping tuples from the system, these approaches focus on reducing CPU load with techniques that perform 2-way [14] joins or m-way joins [15] using only a selected subset of tuples. [15] also seeks to leverage time correlations between streams in determining the subset to join.

Seeking to minimize query inaccuracy over all queries in the system, [4] presents an approach targeted for aggregation queries that places random sampling load shedding operators in the query network. This work only considers query trees where the aggregate operator is at the leaf. The work presented in [4] is extended in [22] which introduces an error model that improves query answer accuracy by utilizing statistics about previous answers.

Following the subset result model, where the largest subset of the original answer is delivered, [31] presents a window-based load shedding technique for aggregate queries where windows are treated as indivisible units and are probabilistically selected to be dropped. Although this work is similar to ours in that it supports arbitrarily nested aggregation queries, but that it focuses on optimizing system throughput by simply considering all aggregates the same, whereas we categorize aggregates based on their susceptibility to propagate incorrect-ness under load shedding.

Recently, [24] proposed a framework that allows for each query, including UDAs, to be weighted differently and have different error functions with the goal of minimizing weighted error. However, [24] does not provide methods to analyze the error of complex networks and queries, but relies on the user to provide the corresponding error results.

Another related work is in the area of uncertain data streams [33, 32], in particularly[33]

which uses GMM to model uncertain query results. However, the techniques in [33] are designed for passively describing the output distributions, and thus are not suitable for load shedding, where the goal is to predict the relationship between sampling rates and the output distributions in order to find the optimal shedding plan. Moreover, [33] requires that the distribution of the input stream be provided in terms of GMM, while our model does not require that. Finally, despite their various optimizations, the techniques in [33] involve costly computations on GMM, thus impractical for the load shedding scenario, where the resources are already insufficient. Third, the uncertainty caused by load shedding is often more complex than those introduced from an uncertain stream, such as the cases discussed in Section 3.2.1.

8 Conclusion

In this paper, we have proposed a new load shedding paradigm that is based on modeling the distribution and the propagation effect of the uncertainty. As a clear improvement on the state-of-the-art, our load shedding proposal achieves great generality in handling arbitrary query networks and complex query operators. Application of our algorithms to data mining tasks has been discussed and developed. Comprehensive experiments have empirically validated the accuracy of our theoretical guarantees as well as their superiority over the state-of-the-art load shedding methods.

References

[1] Stream query repository. http://www-db.stanford.edu/stream/sqr/.

[2] D. J. Abadi et al. Aurora: A data stream management system. In *SIGMOD*, 2003.

[3] A. Arasu et al. Stream: The stanford stream data manager. In *SIGMOD*, 2003.

[4] B. Babcock, M. Datar, and R. Motwani. Load shedding for aggregation queries over data streams. In *ICDE*, 2004.

[5] Y. Bai, H. Thakkar, H. Wang, C. Luo, and C. Zaniolo. A data stream language and system designed for power and extensibility. In *CIKM*, 2006.

[6] C. Cassel, C. Srndal, and J. Wretman. *Foundations of inference in survey sampling*. Wiley, New York [u.a.], 1977.

[7] S. Chandrasekaran et al. Telegraphcq: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.

[8] A. Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009.

[9] A. Deshpande, M. N. Garofalakis, and R. Rastogi. Independence is good: Dependency-based histogram synopses for high-dimensional data. In *SIGMOD*, 2001.

[10] W. Eadie and F. James. *Statistical methods in experimental physics*. World Scientific Publishing, 2006.

[11] T. S. Ferguson. *A Course in Large Sample Theory*. Chapman and Hall, 1996.

[12] A. Frank and A. Asuncion. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2010.

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

[14] B. Gedik, K. L. Wu, P. S. Yu, and L. Liu. Adaptive load shedding for windowed stream joins. In *CIKM*, 2005.

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

[16] B. Gedik, K.-L. Wu, P. S. Yu, and L. Liu. Mobiqual: Qos-aware load shedding in mobile cq systems. In *ICDE*, 2008.

[17] A. C. Gilbert et al. Fast, small-space algorithms for approximate histogram maintenance. In *STOC*, 2002.

[18] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming. http://cvxr.com/cvx, 2011.

[19] S. Guha, N. Koudas, and K. Shim. Data-streams and histograms. In *STOC*, pages 471–475, 2001.

[20] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE*, 2008.

[21] K. Knight. What are the limiting distributions of quantile estimators? In *Statistical Data Analysis Based on the L1-Norm and Related Methods*. Basel: Birkhauser, 2002.

[22] Y.-N. Law and C. Zaniolo. Improving the accuracy of continuous aggregates and mining queries on data streams under load shedding. *IJBIDM*, 3(1):99–117, 2008.

[23] J. N. Lyness and C. B. Moler. Numerical differentiation of analytic functions. *Siam Journal on Numerical Analysis*, 4, 1967.

[24] B. Mozafari and C. Zaniolo. Optimal load shedding with aggregates and mining queries. In *ICDE*, 2010.

[25] S. Qu, K. Zhang, and F. Wang. A global optimization using linear relaxation for generalized geometric programming. *European Journal of Operational Research*, 190(2), 2008.

[26] R. Shumway. *Applied statistical time series analysis*. Prentice-Hall series in statistics. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[27] M. M. Siddiqui and C. Butler. Asymptotic joint distribution of linear systematic statistics from multivariate distributions. *Journal of the American Statistical Association*, 64(325), 1969.

[28] D. Suciu, A. Connolly, and B. Howe. Embracing uncertainty in large-scale computational astrophysics. In *MUD*, 2009.

[29] N. Tatbul, U. Çetintemel, and S. B. Zdonik. Staying fit: Efficient load shedding techniques for distributed stream processing. In *VLDB*, 2007.

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

[31] N. Tatbul and S. B. Zdonik. Window-aware load shedding for aggregation queries over data streams. In *VLDB*, 2006.

[32] T. T. L. Tran, A. McGregor, Y. Diao, L. Peng, and A. Liu. Conditioning and aggregating uncertain data streams: Going beyond expectations. *PVLDB*, 3(1), 2010.

[33] T. T. L. Tran, L. Peng, B. Li, Y. Diao, and A. Liu. Pods: a new model and processing algorithms for uncertain data streams. In *SIGMOD*, 2010.

[34] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1), 1985.

[35] M. Wand and M. Jones. *Kernel smoothing*. Monographs on statistics and applied probability. Chapman & Hall, 1995.

[36] Y. Wang and Z. Liang. A deterministic global optimization algorithm for generalized geometric programming. *Applied Mathematics and Computation*, 168(1), 2005.

[37] Weka Machine Learning Project. Weka. URL http://www.cs.waikato.ac.nz/˜ml/weka.

A Background

Our Gaussian mixture model is based on the asymptotic normality property of aggregate operators. Also the means and variances of the Gaussian distributions are the essential input of our optimization problem. We introduce a set of statistics, named *linear order statistic*, which subsumes all SQL aggregates and non-extreme order statistics. This set of statistics is intensively studied by the statistics community, and closed form results for their converging Gaussian distributions are given.

Let $X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(n)}$ be the order statistics of $\mathcal{D}$. A linear combination $\sum_{j=1}^{n} a_j X_{(j)}$ of these order statistics is referred to as a linear order statistic of X, where a_j is generated by a function $B(x), 0 < x < 1$, i.e.

$$a_j = \Delta B\left(\frac{j}{n+1}\right) = B\left(\frac{j+\frac{1}{2}}{n+1}\right) - B\left(\frac{j-\frac{1}{2}}{n+1}\right)$$

Siddiqui [27] proved Theorem 5 that for n independent observations of a random vector $\mathbf{X} = [X_1, \cdots, X_m]$, an arbitrary finite number of linear order statistics $Z_{i\alpha}$ on component X_i, whether they correspond to the same component or different components of $\mathbf{X}$, follow asymptotic multivariate Gaussian distribution.

Theorem 5. *[27] Consider a collection of linear rank statistics of the form*

$$Z_{i\alpha} = \sum_{j=1}^{n} X_{i(j)} \Delta B_{i\alpha}\left(\frac{j}{n+1}\right), i = 1, \cdots, m, \alpha = 1, \cdots, n_i$$

Let $\zeta_i(t) = F_{X_i}^{-1}(t), 0 < t < 1, h_i(t) = f_{X_i}(\zeta_i(t))$. Let $F_{ij}(x, y)$ be the marginal c.d.f. of (X_i, X_j), and $G_{ij}(s, t) = F_{ij}(\zeta_i(s), \zeta_j(t)), 0 < s, t < 1$. The asymptotic distribution of the vector $\{Z_\alpha^{(i)}, i = 1, \cdots, m, \alpha = 1, \cdots, n_i\}$ is jointly Gaussian, with

$$\begin{aligned} \mathbf{E}(Z_{i\alpha}) &\sim \int_0^1 \zeta_i(t) dB_{i\alpha}(t) \\ n\mathbf{Cov}(Z_{i\alpha}, Z_{j\beta}) &\sim \int_0^1 \int_0^1 \frac{G_{ij}(s,t) - st}{h_i(s) h_j(t)} dB_{i\alpha}(s) dB_{j\beta}(t) \end{aligned}$$

In particular, when $i = j$,

$$n\mathbf{Var}(Z_{i\alpha}) \sim 2 \int_0^1 \int_0^t \frac{s(1-t)}{h_i(s) h_i(t)} dB_{i\alpha}(s) dB_{i\beta}(t)$$

For example, an `AVG` is a linear order statistic with $B(x) = x, 0 < x < 1$; a q-th quantile is a linear order statistic with $B(x) = \mathcal{I}(x \geq q), 0 < x < 1$. Central Limit Theorem for `AVG` and the asymptotic result for q-th quantile are special cases of Theorem 5.

B Proofs

of Theorem 1. Let θ be the output of Θ computed on $\mathcal{D} = \{X_1, X_2, \cdots, X_n\}$. Under Bernoulli sampling with the rate p, $\mathcal{D}$ is randomly partitioned into two data sets. $\mathcal{D}_1$ is the sampled values, while $\mathcal{D}_2$ is the discarded set. The sizes of $\mathcal{D}_1$ and $\mathcal{D}_2$ are n_1 and n_2 respectively. According to Large Sample Theory, $n_1/n \xrightarrow{\mathcal{P}} p$, $n_2/n \xrightarrow{\mathcal{P}} (1-p)$. The ouputs of Θ computed on $\mathcal{D}_1$ and $\mathcal{D}_2$ are denoted by $\hat{\theta}_1$ and $\hat{\theta}_2$ respectively. Θ has asymptotic normality, thus

$$n_1^{1/2}(\hat{\theta}_1 - \mu) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sigma_0^2) \ , \ n_2^{1/2}(\hat{\theta}_2 - \mu) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sigma_0^2)$$

Since Θ is a mean-like statistic,

$$(n_1/n)\hat{\theta}_1 + (n_2/n)\hat{\theta}_2 \xrightarrow{\mathcal{L}} \theta \ , \ (n_2/n)\hat{\theta}_1 - (n_2/n)\hat{\theta}_2 \xrightarrow{\mathcal{L}} \epsilon$$

Therefore,

$$n^{1/2}(\theta-\mu;\epsilon-0) \xrightarrow{\mathcal{L}} \mathcal{N}\left((0;0),(\sigma_0^2;\frac{1-p}{p}\sigma_0^2)\right)$$

Let $\sigma^2=\sigma_0^2/n$. The theorem is proved. □

of Theorem 2. Under the hybrid placement, the current window $\mathcal{D}$ is partitioned into four parts,i.e. tuples used by Θ_1 or Θ_2, tuples used by both of them, and tuples used by none of them, denoted by $\mathcal{D}_i$ with size n_i,$i=1,\cdots,4$ respectively. We have

$$n_1/n \xrightarrow{\mathcal{P}} p_0p_1(1-p_2) \quad , \quad n_2/n \xrightarrow{\mathcal{P}} p_0(1-p_1)p_2$$
$$n_3/n \xrightarrow{\mathcal{P}} p_0p_1p_2 \quad , \quad n_4/n \xrightarrow{\mathcal{P}} (1-p_0)+p_0(1-p_1)(1-p_2)$$

Also following Θ_1 and Θ_2's joint asymptotic normality, we have

$$n_i^{1/2}(\hat{\boldsymbol{\theta}}_{\boldsymbol{i}}-\boldsymbol{\mu}) \xrightarrow{\mathcal{L}} \mathcal{N}(0,\Sigma_0)$$

$\boldsymbol{\theta}$, ϵ_1 and ϵ_2 are all linear combinations of Gaussian random variables, as

$$\sum(n_i/n)\hat{\boldsymbol{\theta}}_{\boldsymbol{i}} \xrightarrow{\mathcal{L}} \boldsymbol{\theta} \ , \ \frac{n_1}{n_1+n_3}\hat{\theta}_{1,1}+\frac{n_3}{n_1+n_3}\hat{\theta}_{3,1}-\theta_1 \xrightarrow{\mathcal{L}} \epsilon_1$$
$$\frac{n_2}{n_2+n_3}\hat{\theta}_{2,2}+\frac{n_3}{n_2+n_3}\hat{\theta}_{3,2}-\theta_2 \xrightarrow{\mathcal{L}} \epsilon_2$$

By following the property of linear combination of Gaussian random variables,

$$n^{1/2}(\boldsymbol{\theta}-\boldsymbol{\mu};\boldsymbol{\epsilon}-0) \xrightarrow{\mathcal{L}} \mathcal{N}\left((0;0),(\Sigma_0;\Sigma_0')\right)$$

where

$$\Sigma_{0,11}'=\frac{1-p_0p_1}{p_0p_1}\Sigma_{0,11} \ , \ \Sigma_{0,22}'=\frac{1-p_0p_2}{p_0p_2}\Sigma_{0,22} \ ,$$
$$\Sigma_{0,12}'=\Sigma_{0,21}'=\frac{1-p_0}{p_0}\Sigma_{0,12}=\frac{1-p_0}{p_0}\Sigma_{0,21}$$

Let $\Sigma=n^{-1}\Sigma_0$, $\widetilde{\Sigma}=n^{-1}\Sigma_0'$. The theorem is proved. □

Sketch of Lemma 1. Prove by induction for each computational operator along a query subgraph. Clearly it holds for the first aggregate (the first aggregates on each query path jointly) by following Theorem 1 (Theorem 2). Assume it holds for all upstream computational operators in the query subgraph of operator Θ. (i) If Θ is a tuple operator, as shown in Theorem 3, $e(\Theta)$ would be a linear combination of $e(\Theta_i)$'s for some Θ_i's in $G(\Theta)$. Therefore, Lemma 1 still holds. (ii) If Θ is a linear statistic aggregate, assume for Θ_i in $G(\Theta)$ $e(\Theta_i)=\sum_j\frac{1-q_{ij}}{q_{ij}}c_{ij}$, where q_i is a product of some sampling rates. By following Theorem 4,

$$e(\Theta)=\sum_i\sum_j w_i\frac{1-pq_{ij}}{pq_{ij}}c_{ij}+\frac{1-p}{p}\left(\sum_i w_i\mu_i^2-\mu^2\right)$$

Clearly, Lemma 1 still holds. For the joint case of multiple linear statistic aggregates, one can prove by similar methods used in the proof of Theorem 2. □

C Generalizations of Our Model

Our model can be generalized to aggregates using logical windows and sliding windows. (i)**Logical Window**. Logical windows do not have a fixed window size. We maintain a histogram of the window size n to approximate n's distribution. For each bin in the histogram, an aggregate Θ are roughly using the same window size and the results in this paper still hold. The distribution of θ and ϵ would be a weighted collection of the GMMs for all the bins. (ii)**Sliding Window**. The output of an aggregate Θ on sliding windows are correlated. Thus a downstream linear statistic aggregate are no longer computed on i.i.d. input. However, one can apply Central Limit Theorem for m-dependent stationary series [11] to obtain similar results as shown in this paper.

As discussed in Section 2.1, in the absence of aggregate operators, load shedding introduces subset error. For minimize such error, an common selected optimization goal would be to deliver the largest possible subset. This problem has been intensively studied [30, 31, 29, 15]. However, under certain circumstances, maximizing the delivered tuple set may not be interesting. In another word, delivering the most useful tuples, instead of delivering the most tuples, may be more desirable to the users. For example, compared to the case when the temperature varies dramatically, a temperature sensor could take a much lower sampling rate when the temperature stay relatively stable, while still achieve the same monitoring accuracy. This intuitive example justifies our goal of delivering the most "useful" tuples.

Similar to the superpopulation model we use to model a data stream, we can also construct a distribution using the sampled tuples, which we assume the sampled tuples are drawn from. Therefore, instead of pursuing delivering the largest possible subset, we aim to deliver the set of sampled tuples whose distribution match the original distribution the most. Under such new optimization goal, the subset error of a load shedding scheme is defined in terms of the distance between the original distribution and the reconstructed one (e.g. measured by a user-defined function), which is again an uncertainty error and thus can be handled by our framework. Such user-defined measure could be very flexible. For instance, the distance can be measured by a weighted sum of the difference between the moments of the distributions, e.g. the simplest case is to use the mean and the variance (first and second order moments) of the sampled tuples.

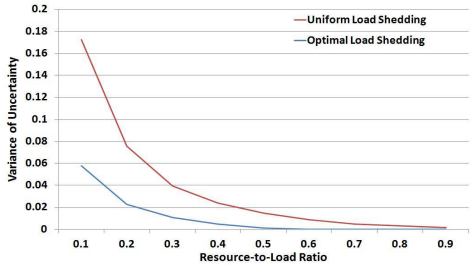


Figure 6: Subset error (the empirical variance of f) vs. resource/load ratio.

Figure 6 shows the result of an experiment, demonstrating our model's effectiveness in minimizing the distribution distance. In the experiment, we have two equal costly queries without any aggregate operators. The two queries produces tuples all

following Gaussian distributions $\mathcal{N}(0,1)$ and $\mathcal{N}(0,5)$, respectively. As known, a Gaussian distribution is fully characterized by its mean and variance. Therefore, we use the difference between a weighted sum of the mean and variance, i.e. $f = \mu + \sigma^2$, which is also known as the first and second order central moments, as the measure of the distance between the original and reconstructed distributions. We vary the resource/load ratio within the range $[0.1, 0.9]$, and plot the variance of the empirically computed f. Clearly, as shown in the figure, our method use optimal shedding plan, generate sampled tuples which give more stable f, which outperforms the naive approach (which would sample equally for the two queries) in terms of our new definition of subset set error.