Joins on Samples: A Theoretical Guide for Practitioners

Dawei Huang  Dong Young Yoon  Seth Pettie  Barzan Mozafari

University of Michigan, Ann Arbor
{hdawei, dyoon, pettie, mozafari}@umich.edu

ABSTRACT

Despite decades of research on approximate query processing (AQP), our understanding of sample-based joins has remained limited and, to some extent, even superficial. The common belief in the community is that joining random samples is futile. This belief is largely based on an early result showing that the join of two uniform samples is not an independent sample of the original join, and that it leads to quadratically fewer output tuples. However, unfortunately, this result has little applicability to the key questions practitioners face. For example, the success metric is often the final approximation’s accuracy, rather than output cardinality. Moreover, there are many non-uniform sampling strategies that one can employ. Is sampling for joins still futile in all of these settings? If not, what is the best sampling strategy in each case? To the best of our knowledge, there is no formal study answering these questions.

This paper aims to improve our understanding of sample-based joins and offer a guideline for practitioners building and using real-world AQP systems. We study limitations of offline samples in approximating join queries: given an offline sampling budget, how well can one approximate the join of two tables? We answer this question for two success metrics: output size and estimator variance. We show that maximizing output size is easy, while there is an information-theoretical lower bound on the lowest variance achievable by any sampling strategy. We then define a hybrid sampling scheme that captures all combinations of stratified, universe, and Bernoulli sampling, and show that this scheme with our optimal parameters achieves the theoretical lower bound within a constant factor. Since computing these optimal parameters requires shuffling statistics across the network, we also propose a decentralized variant where each node acts autonomously using minimal statistics. We also empirically validate our findings on popular SQL and AQP engines.

DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/4.0/. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 12, No. xxx
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

Approximate query processing (AQP) has regained significant attention in recent years due to major trends in the industry. Larger datasets and the rise of shared and hosted infrastructure have made it more expensive to achieve interactive-speed analytics. AQP presents itself as a viable alternative in scenarios where perfect decisions can be made with imperfect answers [9]. AQP is most appealing when negligible loss of accuracy can be traded for a significant gain in speedup or computational resources. Adhoc analytics [22], visualization [23, 27, 50, 51, 57], IoT [3], A/B testing [8], email marketing and customer segmentation [28], and real-time threat detection [2] are examples of such use cases.

Sampling and Joins— Sampling is one of the most widely-used techniques for general-purpose AQP [22]. The high level idea is to execute the query on a small sample of the original table(s) in order to provide a fast, but approximate, answer. While effective for simple aggregates, using samples for join queries has long remained an open problem [7]. There are two main approaches to AQP: offline or online. Offline approaches [6, 7, 9, 15, 27, 47, 49] build samples (or other synopses) prior to query arrival. Dynamic sample-selection At run time, they simply choose appropriate samples that can yield the best accuracy/efficiency for each incoming query. Online approaches, on the other hand, wander-join perform much of their sampling at run time based on the query at hand e.g., [13, 20, 33, 36, 46, 55]. Naturally, offline sampling leads to significantly higher speedup, while online techniques can support a much wider class of queries [56]. The same taxonomy applies to join approximation: offline techniques perform joins on previously-prepared samples [7, 11, 18, 47, 58], while online approaches seek to produce a sample of the output of the join at run time [25, 29, 40, 42]. As mentioned, the latter often means more modest speedups (e.g., $2 \times 55$) which may not be sufficient to justify approximation, or additional requirements (e.g., an index for each join column [40]) which may not be acceptable to many applications. Thus, our focus in this paper—and what is considered an open-problem—is the offline approach: joins on samples, not sampling the join’s output.

Joins on Samples— The simplest strategy is as follows. Given a large table $T_1$ and a smaller table $T_2$, create a uniform random sample of each, say $S_1$ and $S_2$ respectively, and then use $S_1 \bowtie S_2$ to approximate aggregate statistics of $T_1 \bowtie T_2$. This will lead to significant speedup if samples are much smaller than original tables, i.e., $|T_1| >> |S_1|$. One of the earliest results in this area shows that this simple strategy is futile for two reasons [4]. First, joining two uniform samples leads to quadratically fewer output tuples, i.e., joining two uniform samples that are each $p$ fraction ($0 \leq p < 1$) of the original tables will only produce $p^2$ of the output tuples of the original join (see Figure [4]). Second, joining uniform samples of two tables does not yield an independent sample of the join of those tables (see
Section 2.1 for details.\footnote{Prior work has stated this, as joining uniform samples is not a uniform sample of the join [7]. We avoid this terminology because uniform means equal probability of inclusion, and in this case each tuple does appear in the join of the uniform samples with equal probability, but not independently. In other words, joining two i.i.d. samples is an identical, but not independent, sample of the join tuples.}. The dependence of the output tuples can drastically lower the approximation accuracy [6,17],\footnote{Uniform Bernoulli Sampling. Given a column $\mathbf{J}$, a (perfect) hash function $h : \mathbf{J} \mapsto [0,1]$, and a sampling rate $p$, this strategy includes $J$ can also be a set of multiple columns.}

hashed-samples Prior Work—Universe sampling [31,36,47] addresses the first drawback of uniform sampling. Although universe sampling avoids quadratic reduction of output, it creates even more correlation in its output, leading to much lower accuracy (see Section 5.1).

Atserias et al. [12] show that computing exact joins with a small memory or time budget is hard, by providing a worst case lower bound for any query involving equi-joins on multiple relations. For instance, the maximum possible join size for any cyclic join on three $n$-tuple relations is $\Theta(n^{1.5})$. Thus, a natural question is whether approximating joins is also hard with small memory or time.

Our Goal—In this paper, we specifically focus on understanding the limitation of using offline samples in approximating join queries. Given a sampling budget, how well can we approximate the join of two tables using their offline samples? To answer this question, we must first define what constitutes a “good” approximation of a join. We consider two metrics: (1) output cardinality and (2) aggregation accuracy. The former is the number of tuples of the original join that also appear in the join of the samples, whereas the latter is the error of the aggregates estimated from the sample-based join with respect to their true values, if computed on the original join. Because in this paper we only consider unbiased estimators, we measure approximation error in terms of the variance of our estimators.

For the first metric, we provide a simple proof showing that universe sampling is optimal, i.e., no sampling scheme with the same sampling rate can outperform universe sampling in terms of the (expected) output cardinality. However, as we show in Section 5.1 retaining a large number of join tuples does not imply accurate aggregates. It is therefore natural to also ask about the lowest variance that can be achieved given a sampling rate. To the best of our knowledge, this has remained an open problem to date. For the first time, we formally study this problem and offer an information-theoretical lower bound to this question. We also present a hybrid sampling scheme that matches this lower bound within a constant

factor. This scheme involves a centralized computation, which can become prohibitive for large tables due to large amounts of statistics that need to be shuffled across the network. Thus, we also propose a decentralized variant that only shuffles a minimal amount of information across the nodes—such as the table size and maximum frequency—but still achieves the same worst case guarantees. Finally, we generalize our sampling scheme to accommodate a priori information about filters (i.e., WHERE clause).

In this paper, we make the following contributions:

1. We discuss two metrics—output size and estimator’s variance—for measuring the quality of join approximation, and show that universe sampling is optimal for output size and there is an information-theoretical lower bound for variance (Section 5).

2. We formalize a hybrid sampling scheme, called Stratified-Universe-Bernoulli Sampling (SUBS), which allows for different combinations of stratified, universe, and Bernoulli sampling. We derive optimal sampling parameters within this scheme, and show that they achieve the theoretical lower bound of variance within a constant factor (Section 5.3). We also extend our analysis to accommodate additional information regarding the WHERE clause (Section 6).

3. Through extensive experiments, we also empirically show that our optimal sampling parameters achieve lower error than existing sampling schemes in both centralized and decentralized scenarios (Section 7).

2. BACKGROUND

In this section, we provide the necessary background on sampling-based join approximation. We also formally state our problem setting and assumptions.

2.1 Sampling in Databases

The following are the three main popular sampling strategies (operators) used in AQP engines and database systems.

1. Uniform/Bernoulli Sampling. Any strategy that samples all tuples with the same probability is considered a uniform (random) sample. Since enforcing fixed-size sampling without replacement is expensive in distributed systems, Bernoulli sampling is considered a more efficient strategy [50]. In Bernoulli sampling, each tuple is included in the sample independently, with a fixed sampling probability $p$. In this paper, for simplicity, we use “uniform” and “Bernoulli” interchangeably. As mentioned in Section 1 joining two uniform samples leads to quadratically fewer output tuples. Further, it does not guarantee an i.i.d. sample of the original join [7]: the output is a uniform sample of the join but not an independent one. Consider an arbitrary tuple of the join, say $(t_1, t_2)$ where $t_1$ is from the first table and $t_2$ is from the second one.

The probability of this tuple appearing in the join of the samples is always the same value, i.e., $p^2$. The output is therefore a uniform sample. However, the tuples are not independent: consider another tuple of the join, say $(t_1, t'_2)$ where $t'_2$ is another tuple from the second table joining with $t_1$. If $(t_1, t'_2)$ appears in the output, the probability of $(t_1, t'_2)$ also appearing becomes $p$ instead of $p^2$, which would be the probability if they were independent.

2. Universe Sampling. Given a column $\mathbf{J}$, a (perfect) hash function $h : \mathbf{J} \mapsto [0,1]$, and a sampling rate $p$, this strategy includes
a tuple $t$ in the table if $h(t,J) \leq p$. This strategy is often used for equi-joins, in which case the same $p$ value and hash function $h$ are applied to the join columns in both tables. This ensures that when a tuple $t_1$ is sampled from one table, any matching tuple $t_2$ from the other table is also sampled, simply because $t_1.J = t_2.J \Rightarrow h(t_1,J) = h(t_2,J)$. This is why joining two universe samples with a sampling rate of $p$ produces $p$ fraction of the original join output in expectation. The output is a uniform sample of the original join, as each join tuple appears with the same probability $p$. However, there is more dependence among the output tuples. Consider two join tuples $(t_1, t_2)$ and $(t'_1, t'_2)$ where $t_1, t'_1, t_2, t'_2$ all share the same join key. Then, if $(t_1, t_2)$ appears, the probability of $(t'_1, t'_2)$ also appearing will be 1. Likewise, if $(t_1, t_2)$ does not appear, the probability of $(t'_1, t'_2)$ appearing will be 0. Higher dependence means lower accuracy (see Section 3.1).

3. Stratified Sampling. The goal of stratified sampling is to ensure that minority groups are sufficiently represented in the sample. Groups are defined according to one or multiple columns, called the stratified columns. A group (a.k.a. a stratum) is a set of tuples that share the same value under those stratified columns. Given a set of stratified columns $C$ and an integer parameter $k_{\text{tuple}}$, a stratified sampling is a scheme that guarantees at least $k_{\text{tuple}}$ tuples are sampled uniformly at random from each group. When a group has fewer than $k_{\text{tuple}}$ tuples, all of them are retained.

2.2 Quality Metrics

Different metrics can be used to assess the quality of a join approximation. In this paper, we focus on the following two, which are used by most AQP systems.

Output Size/Cardinality—This metric is the number of tuples of the original join that also appear in the join of the samples. This metric is mostly relevant for exploratory usecases, where users visualize or examine a subset of the output. In other cases, where an aggregate is computed from the join output, retaining a large number of output tuples does not guarantee accurate answers (we show this in Section 3.1).

Variance—In scenarios where an aggregate function needs to be calculated from the join output, the error of the aggregate approximation is more relevant than the number of intermediate tuples generated. For most non-extreme statistics, there are readily available unbiased estimators, e.g., Horvitz-Thompson estimator [34]. Thus, a popular indicator of accuracy is the variance of the estimator [9], which determines the size of the confidence interval given a sample size.

2.3 Problem Statement

In this section, we formally state the problem of sample-based join approximation. The notations used throughout the paper are listed in Table 1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1, T_2$</td>
<td>Two tables for the join</td>
</tr>
<tr>
<td>$S_i$</td>
<td>A sample generated from table $T_i$</td>
</tr>
<tr>
<td>$J$</td>
<td>Column(s) used for the join between $T_1$ and $T_2$</td>
</tr>
<tr>
<td>$W$</td>
<td>Column being aggregated (e.g., SUM, AVG)</td>
</tr>
<tr>
<td>$C$</td>
<td>Column(s) used for filters (i.e., WHERE clause)</td>
</tr>
<tr>
<td>$U$</td>
<td>Set of all possible values of $J$</td>
</tr>
<tr>
<td>$a, b$</td>
<td>Frequency vectors for $T_1$ and $T_2$ w.r.t. its join column resp.</td>
</tr>
<tr>
<td>$a_v, b_v$</td>
<td>Number of tuples with join value $v$ in $T_1$ and $T_2$, resp.</td>
</tr>
<tr>
<td>$\hat{J}_{agg}$</td>
<td>Estimator for a join query with aggregate function $agg$</td>
</tr>
<tr>
<td>$k_{\text{tuple}}$</td>
<td>Minimum number of tuples to be kept per group in stratified sampling</td>
</tr>
<tr>
<td>$k_{\text{key}}$</td>
<td>Minimum number of join keys per group to apply universe sampling (universe sampling is not applied to groups with fewer join keys)</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>Sampling rate of universe sampling</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Sampling rate of uniform sampling</td>
</tr>
</tbody>
</table>

Table 1: Notations.

and $S_i$ that minimize $\hat{J}_{agg}$’s variance or maximize its output size such that $E[|S_1| + |S_2|] \leq \epsilon \times (|T_1| + |T_2|)$.

Note that we define the sampling budget in terms of an expected size (rather than a strict one), since sampling schemes are probabilistic in nature and may slightly over- or under-use the budget.

To formally study this problem, we first need to define a class of reasonable sampling strategies. In Section 4, we define a hybrid scheme that can capture different combinations of stratified, universe, and uniform sampling.

2.4 Scope and Limitations

To simplify our analysis, we limit our scope in this paper.

Flat Equi-joins—We focus on equi (inner) joins as the most common form of joins in practice. We also support both WHERE and GROUPBY clauses. Because our focus is on the join itself, we ignore nested queries and only consider flat (or flattened) queries. We primarily focus on two-way joins. However, our results extend to multi-way joins with the same join column(s).

Aggregate Functions—Most AQP systems do not support extreme statistics, such as Min or Max [44]. Likewise, we only consider non-extreme aggregates, and primarily focus on the three basic functions, COUNT, SUM, and AVG. However, we expect our techniques to easily extend to other mean-like statistics as well, such as VAR, STDEV, and PERCENTILE.

3. HARDNESS

In this section, we explain why providing a large output size is insufficient for approximating joins, and formally show the hardness of approximating common aggregates based on the theory of communication complexity.

3.1 Output Size

Uniform sampling leads to small output size. If we sample at a rate $q$ from both table $T_1$ and table $T_2$, the join of samples contains only $q^2$ fraction of $T_1 \bowtie T_2$ in expectation. Moreover, the join of two independent samples of the original tables is in general not an
independent sample of $T_1 \bowtie T_2$, which hurts the sample quality. In contrast, universe sampling \[31, 36\] with sample rate $p$ can in expectation sample a $p$ fraction of $T_1 \bowtie T_2$. We prove that this is optimal (all omitted proofs are deferred to Appendix A).

**Theorem 1.** No sampling scheme with sample rate $α$ can guarantee more than $α$ fraction of $T_1 \bowtie T_2$ in expectation for all possible inputs.

However, a large number of tuples retained in the join does not imply that the original join query can be accurately approximated. As pointed out in \[13\], universe sampling shows poor performance in approximating queries when the frequencies of keys are concentrated on a few elements. Consider the following extreme example with tables $T_1$ and $T_2$, each comprised of $n$ tuples with a single value 1 in their join key. In this example, universe sampling with the sampling rate $p$ produces an estimator of variance $n^2/p$, while uniform sampling with rate $q$ has a variance of $n^2/q^2$, which is much lower when $p = q$ and $n$ is large. Thus, a larger output size does not necessarily lead to a better approximation of the query.

### 3.2 Approximating Aggregate Queries

In this section, we focus on the core question: why is approximating common aggregates (e.g., COUNT, SUM and AVG) hard when using a small sample (or more generally, a small summary)? We address this question using the theory of communication complexity. Specifically, to show that computing COUNT on a join is hard, we reduce it to set intersection, a canonically hard problem in communication complexity. Assume that both Alice and Bob each hold a set of size $k$, say $A$ and $B$, respectively. They aim to estimate the size of $t = |A \cap B|$. Pagh et al. \[45\] show that if Alice only sends a small summary to Bob, any unbiased estimator that Bob uses will have a large variance.

**Theorem 2 (See \[45\]).** Any one-way communication protocol that estimates $t$ within relative error $δ$ with probability at least $2/3$ must send at least $Ω(k/(δ^2))n$ bits.

**Corollary 3.** Any estimator to $|A \cap B|$ produced by Bob that is based on an $s$-bits summary by Alice must have a variance of at least $Ω(k/s)$.

Any sample of size $s$ can be encoded using $O(\log (\frac{n}{s}))$ bits, implying that any estimator to COUNT that is based on a sample of size $s$ from one of the tables must have a variance of at least $Ω(k/s)$.

Estimating SUM queries is at least as hard as estimating COUNT queries, since any COUNT can be reduced to a SUM by setting all entries in the SUM column to 1.

From the hard instance of set intersection, we can also derive a hard instance for AVG queries. Based on Theorem 2, any summary of $T_1$ that can distinguish between intersection size $t(1 + δ)$ and $t(1 - δ)$ must be at least of size $Ω(k/(tδ^2))$ bits. Now we reduce this problem to estimating an AVG query.

Here, the two tables consist of $k + \sqrt{t}$ tuples each. The first $k$ tuples of $T_1$ and $T_2$ are from the hard instance of set intersection, and the values of their AVG column are set to 2. The join column of the last $\sqrt{t}$ tuples is set to some common key $v_i$ that is in the first $k$ tuples, and their AVG column is set to 0. Therefore, the intersection size from the first $k$ tuples is at least $t(1 + δ)$ (or at most $t(1 - δ)$) if and only if the result of the AVG query is at least $\frac{2t(2t+1)}{t(2t+1)} = (1 + O(δ))r$ (or at most $\frac{2t(2t+1)}{t(2t+1)} = (1 - O(δ))r$). By re-scaling $δ$ by a constant factor, we can get the following theorem:

**Theorem 4.** Any summary of $T_1$ that can estimate an AVG query with precision $δ$ with probability at least $2/3$ must have a size of at least $Ω(n/(tδ^2))$.

### 4. GENERIC SAMPLING SCHEME

To formally argue about the optimality of a sampling strategy, we must first define a class of sampling schemes. As discussed in Section 2.1, there are three well-known sampling operators: stratified, universe, and Bernoulli (uniform). However, these atomic operators can themselves be combined. For example, one can apply universe sampling of rate $0.1$ and then Bernoulli sampling of rate $0.2$ for an overall effective sampling rate of $0.02$. To account for such hybrid schemes, we define a generic scheme that combines universe and Bernoulli sampling, called UBS. We also define a more generic scheme that combines all three of stratified, universe and Bernoulli sampling, called SUBS. It is easy to show that the basic sampling operators are a special case of SUBS. First, we define the effective sample rate.

**Definition 5 (Effective sampling rate).** We define the effective sampling rate of a sampling scheme as the expected ratio of the size of the resulting sample to that of the original table.

**Definition 6 (Uniform-Bernoulli Sampling (UBS Scheme)).** Given a table $T$ and a column (or set of columns) $J$ in $T$, a UBS scheme is defined by a pair $(p, q)$, where $0 < p \leq 1$ is a universe sampling rate and $0 < q \leq 1$ is a Bernoulli (or uniform) sampling rate. Let $h: \mathcal{U} \rightarrow [0, 1]$ be a perfect hash function. Then, a sample of $T$ produced by this scheme, $S = UBS_{p,q}(T, J)$, is produced as follows:

**Algorithm 1** $UBS_{p,q}(T, J)$

$S \leftarrow \emptyset$ for each tuple $t$ do
  if $h(t,J) < p$ then
    Include $t$ in $S$ independently w/ prob. $q$.
  end

It is easy to see that the effective sampling rate of a UBS scheme $(p, q)$ is $p \cdot q$. Thus, the effective sampling rate of is independent of the actual distribution of the values in the table (and column(s) $J$).

The goal of this sampling paradigm is to optimize the trade-off between universe sampling and Bernoulli sampling in different instances. At one extreme, when each join value appears exactly once in both table, universe sampling leads to lower variance than Bernoulli sampling. This is because independent Bernoulli sampling has trouble matching tuples with the same join value, while universe sampling guarantees that when a tuple is sampled, all matching tuples in other table are also sampled. At the other extreme, if all tuples have the same join value in both tables (i.e., the join becomes a Cartesian product of the two tables), universe sampling will either sample the entire join, or sample nothing at all, while uniform sampling will have a sample size concentrated around $qN$, thus giving an estimator of much lower variance. In section 5.1 to 5.3, we give a comprehensive discussion on how to optimize $p$ and $q$ for different tables and different queries.
The Stratified-Universe-Bernoulli Sampling Scheme applies to a table $T$ that is divided into $K$ groups (i.e., strata), denoted as $G_1, G_2, \ldots, G_K$.  

**Definition 7 (Stratified-Universe-Bernoulli Sampling (SUBS) Scheme).** Given a table $T$ of $N$ rows and a column (or set of columns) $J$ in $T$, a SUBS scheme is defined by a tuple $(p_1, p_2, \ldots, p_K, q_1, q_2, \ldots, q_K)$, where $0 < p_i, q_i \leq 1$ are the universe sampling rate and Bernoulli sampling rate. Let $h : \mathcal{U} \rightarrow [0, 1]$ be a perfect hash function. Then, a sample of $T$ produced by this scheme, $S = \text{UBS}_{p,q}(T, J)$, is produced as follows:

**Algorithm 2 SUBS**$_{p_1, \ldots, p_K, q_1, \ldots, q_K}(T, G, J)$

$S \leftarrow \emptyset$

for each group $G_i$

for each tuple $t$ in $G_i$

if $h(t.J) < p_i$ then

| Include $t$ in $S$ independently w/ prob. $q_i$.

end

end

end

Let $|G_i|$ denote the number of tuples in group $G_i$. Then the effective sampling rate of a SUBS scheme is $\sum_i p_i \cdot q_i \cdot |G_i|/N$. We call $\epsilon_i = p_i \cdot q_i$ the effective sampling rate for group $G_i$.

In both UBS and SUBS schemes, the user specifies $\epsilon$ as his/her desired sampling budget, given which our goal is to determine optimal sampling parameters $p_i$ and $q_i$ values such that the variance of our join estimator is minimized. In Section 5, we derive the optimal $p$ and $q$ for UBS. For SUBS, in addition to $\epsilon$, the user also provides two additional parameters $k_{\text{key}}$ and $k_{\text{tuples}}$ (explained below). Next, we show how to determine the effective sampling rate $\epsilon_i$ for each group $G_i$ based on these parameters in SUBS. Given $\epsilon_i$ for each group, the problem is then reduced to finding the optimal parameters for UBS for that group (i.e., $p_i$ and $q_i$). Moreover, as we will show in Sections 5.1, 5.2, and 5.3, the universe sampling rate for every group must be the same, and must be the same as the universe sampling rate of the other table in two-way joins. Hence, we use a single universe sampling rate $p = p_1 = \ldots = p_k$ across all groups.

As mentioned in Section 2.1, $k_{\text{tuples}}$ is a user-specified lower bound on the minimum number of tuples in each group the sample must retain. $k_{\text{key}}$ is an additional user-specified parameter required for the SUBS scheme. It specifies a threshold as when to activate the universe sampler. In particular, if a group contains too few (i.e., less than $k_{\text{key}}$) join keys, we do not perform any universe sampling as it will have a high chance of filtering out all tuples. Hence, we apply universe sampling only to those groups with $k_{\text{key}}$ join keys. For groups with fewer than $k_{\text{key}}$ join keys, we will only apply Bernoulli sampling with rate $\epsilon_i$.

We call a group large if it contains at least $k_{\text{key}}$ join keys, otherwise, we call it a small group. We use $N_i$ to denote the total number of tuples in all large groups, and $N_s$ to denote the total number of tuples in all small groups. Similarly, let $M_i$ and $M_s$ denote the number of large and small groups respectively. Then, we decide the sampling budget $\epsilon_i$ for each group $G_i$ as follows:

1. If $M_s k_{\text{tuples}} > \epsilon_i N_s$ or $M_i k_{\text{tuples}} > \epsilon_i N_s$, we notify the user that creating a sample given their parameters is infeasible.
2. Otherwise,

$\epsilon_i = \frac{K_i k_{\text{tuples}}}{N_s}$ and let $\epsilon_i'' = \epsilon - \epsilon_i'$. Then for each small group $G_i$, the sampling budget is $\epsilon_i = \frac{k_{\text{tuples}}}{M_i} + \epsilon_i''$.

$\epsilon_i = \frac{K_i k_{\text{tuples}}}{N_s}$ and let $\epsilon_i' = \epsilon - \epsilon_i''$. Then for each large group $G_i$, the sampling budget is $\epsilon_i = \frac{k_{\text{tuples}}}{M_i} + \epsilon_i'$.

Once $\epsilon_i$ is determined for each group, the problem of deciding optimal SUBS parameters is reduced to deciding the optimal SUBS parameters for $K$ separate groups. This effective sampling rate $\epsilon_i$ guarantees that each large group will have at least $t$ tuples in the sample on average, and the remaining budget is divided evenly. Thus, the corresponding uniform sampling rate for each large group is $q_i = \epsilon_i/p$. Moreover, we pose the constraint that the universe sampling rate $p$ should be at least $1/s$ to guarantee that, on average, there is at least one join key passing through the universe sampler.

For small groups, we simply apply uniform sampling with rate $\epsilon_i$. This is equivalent to setting $p = 1$ for these groups.

Overall, this strategy provides the following guarantees:

1. Each group will have at least $t$ tuples in the sample, on average.
2. The probability of each group being missed is at most $(1 - 1/s)^t < 0.367$. In general, if we set $p > \epsilon/s$ for some constant $c > 1$, this probability will become $0.367^c$.
3. The approximation of the original query will be optimal in terms of its variance (see Sections 5.1, 5.2).

### 5. OPTIMAL SAMPLING

As shown in Section 4, finding the optimal sampling parameters within the SUBS scheme can be reduced to that within the UBS scheme. Thus, in this section, we focus on deriving the UBS parameters that minimize error for each aggregation type (COUNT, SUM, and AVG). Initially, we also assume there is no WHERE clause. Later, in Section 6, we show how to handle WHERE conditions and how to create a single sample instead of creating one per each aggregation type and WHERE condition.

**Centralized vs. Decentralized** — For each aggregation type, we analyze two scenarios: centralized and decentralized. Centralized setting is when the frequencies of the join keys in both tables are known. This represents situations where both tables are stored on the same server, or each server communicates its full frequency statistics to other parties. Decentralized setting represents a scenario where the two tables are each stored on a separate server, and exchanging full frequency statistics across the network is costly.  

**Decentralized Protocols** — In a decentralized setting, each party (i.e., server) only has access to full statistics of its own table (e.g., frequencies, join column distribution). The goal then is for each party to determine its sampling strategy, while minimizing communications with the other party. Depending on the amount of information exchanged, one can pursue different protocols for achieving this goal. In this paper, we study a simple sampling protocol, which we call DICTATORSHIP. Here, one server, say party 1, is chosen as the dictator. We also assume that the other’s sampling budgets and table sizes ($c_1, c_2, |T_1|$, and $|T_2|$). The dictator observes the distributional information of its own table, say $T_1$, and decides a shared universe sampling rate $p$ between $\max (c_1, c_2)$ and 1. This $p$ is sent to the other server (party 2) and both servers use $p$ as their universe sampling rate. Their uniform sampling rates will thus be $q_1 = c_1/p$ and $q_2 = c_2/p$, respectively.

---

*Here, we focus on two servers, but the math can easily be generalized to decentralized networks of multiple servers.*

*Using the same universe sampling rate is justified by Lemma 10.*

---

5
Since party only has $T_1$’s frequency information, it chooses an optimal value of $p$ that minimizes the worst case variance of $\hat{J}_{avg}$, i.e., the variance when the frequencies in $T_2$ are chosen adversarially. This can be formulated as a robust optimization:

$$p^* = \arg\min_{\max\{s_1, s_2\} \leq p \leq 1} \max_{b \in \beta} B \text{Var}[\hat{J}_{avg}]$$

where $b$ ranges over all possible frequency vectors of $T_2$. There are more complex protocols that exchange additional information (e.g., voting, iterative convergence), which we defer to Appendix C and use DICTATORSHIP in our decentralized analysis.

5.1 Join Size Estimation: Count on Joins

We start by considering the following simplified query:

\[
\begin{align*}
\text{select} \quad \text{count}(\ast) \\
\text{from} \quad T_1 \text{ join } T_2 \text{ on } J
\end{align*}
\]

where $T_1$ and $T_2$ are two tables joined on column(s) $J$. Consider two samples, $S_1 = UBS_{p_1.q_1}(T_1, J)$ and $S_2 = UBS_{p_2.q_2}(T_2, J)$. Then, we can define an unbiased estimator for the above query, $E_{count} = |T_1 \bowtie J T_2|$, using $S_1$ and $S_2$ as follows. Observe that given any pair of tuples $t_1 \in T_1$ and $t_2 \in T_2$, where $t_1.J = t_2.J$, the probability that $(t_1, t_2)$ enters $S_1 \bowtie S_2$ is $p_{min}q_1q_2$, where $p_{min} = \min\{p_1, p_2\}$. Hence, the following is an unbiased estimator for $E_{count}$.

\[
\hat{J}_{count}(p_1, q_1, p_2, q_2, S_1, S_2) = \frac{1}{p_{min}q_1q_2} |S_1 \bowtie J S_2|
\]

When the arguments $p_1, q_1, p_2, q_2, S_1, S_2$ are clear from the context, we omit these arguments and simply write $\hat{J}_{count}$.

Definition 8 (Join Size Estimation). Given sampling budgets $\epsilon_1, \epsilon_2$, our goal is to find parameters $(p_1, q_1)$ and $(p_2, q_2)$ that minimize the variance of $\hat{J}_{count}$ subject to $p_1 q_1 = \epsilon_1$ and $p_2 q_2 = \epsilon_2$.

Lemma 9. Let $S_1 = UBS_{p_1.q_1}(T_1, J)$ and $S_2 = UBS_{p_2.q_2}(T_2, J)$. The variance of $\hat{J}_{count}$ is as follows:

\[
\text{Var}[\hat{J}_{count}] = \frac{1 - p}{p} \gamma_{2.2} + \frac{1 - q_2}{p_{q_2}} \gamma_{2.1} + \left(1 - \frac{q_1}{p_{q_1}}\right) \left(1 - q_2\right) \gamma_{1.1}
\]

where $\gamma_{i,j} = \sum a_i a_j b_i^j$.

To minimize \text{Var}[\hat{J}_{count}] under a fixed sampling budget, the two tables should always use the same universe sampling rate. If $p_1 > p_2$, the effective universe sampling rate is only $p_2$, i.e., only $p_2$ fraction of the join keys inside $T_1$ appear in the join of the samples, and the remaining $p_1 - p_2$ fraction is simply wasted. Then, we can change the universe sampling rate of $T_1$ to $p_2$ and increase its uniform sampling rate to obtain a lower variance.

Lemma 10. Given tables $T_1, T_2$ joined on column(s) $J$, a fixed sampling parameter $(p_1, q_1)$ for $T_1$, and a fixed effective sampling rate $\epsilon_2$ for $T_2$, the variance of $\hat{J}_{count}$ is minimized when $T_2$ uses $p_1$ as its universe sampling rate and correspondingly $\epsilon_2/p_1$ as its uniform sampling rate.

Note that Lemma 10 applies to both centralized and decentralized settings, i.e., it applies to any feasible sampling parameter $(p_1, q_1)$ and $(p_2, q_2)$, regardless of how the sampling parameter is decided. Next, we analyze each setting.

5.1.1 Centralized Sampling for Count

Theorem 11. When $T_1$ and $T_2$ use sampling parameters $(p, \epsilon_1/p)$ and $(p, \epsilon_2/p)$. $\hat{J}_{count}$’s variance is given by:

\[
\text{Var}[\hat{J}_{count}] = \left(\frac{1}{p} - 1\right) \gamma_{2.2} + \left(1 - \frac{1}{\epsilon_2}\right) \gamma_{2.1} + \left(1 - \frac{1}{\epsilon_1}\right) \gamma_{1.1} + \left(1 - \frac{1}{\epsilon_2}\right) \gamma_{1.2} + \left(\frac{p}{\epsilon_2} - 1\right) \gamma_{2.1} + \frac{1}{\epsilon_1} \gamma_{1.1}
\]

Since each term in Theorem 11 that depends on $p$ is proportional either to $p$ or $1/p$, to find a $p$ that minimizes the variance, one can simply set the first order derivatives (with respect to $p$) to 0.

Theorem 12. Let $T_1$ and $T_2$ be two tables joined on column(s) $J$. Let $a_v$ and $b_v$ be the frequency of value $v$ in column(s) $J$ of tables $T_1$ and $T_2$, respectively. Given their sampling rates $\epsilon_1$ and $\epsilon_2$, the optimal sampling parameters $(p_1, q_1)$ and $(p_2, q_2)$ are given by:

\[
p_1 = \min\{1, \max\{\epsilon_1, \epsilon_2\} \sqrt[p_1q_1q_2]{\frac{\gamma_{2.2} - \gamma_{1.2} - \gamma_{2.1} + \frac{1}{\epsilon_1}\gamma_{1.1}}{\gamma_{1.1}}\}
\]

and $q_1 = \epsilon_1/p$, $q_2 = \epsilon_2/p$.

Substituting this into Lemma 8, the resulting variance is only a constant factor of Theorem 11’s, theoretical limit. For instance, consider a primary-key-foreign-key join query where $a_v \in \{0, 1\}$ and $b_v$ is smaller than some constant, say 5, and $\epsilon_1 = \epsilon_2 = \epsilon$ for any $\epsilon$. Then, Theorem 11 becomes $(1/e - 1)J$ where $J = \sum a_v b_v$ is the size of the join. Since $\epsilon$ is the expected ratio of the sample to table size, the expression $(1/e - 1)J$ matches the lower bound in Corollary 3 except for a constant factor.

5.1.2 Decentralized Sampling for Count

Motivated by Lemma 10, the DICTATORSHIP protocol uses the same universe sampling rate $p$ for both parties in the decentralized setting, by solving the following robust optimization problem:

\[
\arg\min_{\max\{s_1, s_2\} \leq p \leq 1} \max_{b \in \beta} B \text{Var}[\hat{J}_{count}]
\]

Based on Lemma 9 and 12, given the effective sampling rates $\epsilon_1$ and $\epsilon_2$, we can express \text{Var}[\hat{J}_{count}] as a function of frequencies \{a_v\} and \{b_v\}, and universe sampling rate $p$ as follows.

\[
\text{Var}[\hat{J}_{count}] = \left(\frac{1}{p} - 1\right) \gamma_{2.2} + \left(1 - \frac{1}{\epsilon_2}\right) \gamma_{2.1} + \left(1 - \frac{1}{\epsilon_1}\right) \gamma_{1.1} + \left(1 - \frac{1}{\epsilon_2}\right) \gamma_{1.2} + \left(\frac{p}{\epsilon_2} - 1\right) \gamma_{2.1} + \frac{1}{\epsilon_1} \gamma_{1.1}
\]

Lemma 13. Let $a_v$ be the maximum frequency in in table $T_1$, $v_*$ be any value that has that frequency, and $n_v$ be the total number of tuples in $T_2$. The optimal value for the problem $\max_{b \in \beta} \text{Var}[\hat{J}_{count}]$ is given by $(1/p - 1) a_v^2 n_v^2 + (1/p - 1) a_v^2 n_v + (1/p - 1) a_v n_v^2 + (1/p - 1) a_v n_v^2 + (p/\epsilon_2) - 1/a_v n_v + (\frac{1}{\epsilon_1} - \frac{1}{\epsilon_2})/a_v n_v$.

In equation 3, given \{a_v\} and a fixed $p$, the variance is a convex function of the frequency vector \{b_v\}. Thus, the frequency vector \{b_v\} that maximizes the variance, i.e., the worst case \{b_v\}, is one where exactly one join key has a non zero frequency. This join key should be the one with the maximum frequency in $T_1$. This is not a representative case and using it to decide a sampling rate might drastically hinder the performance on average. We therefore
require that both servers also share a simple piece of information regarding the maximum frequency of the join keys in each table, say \( F_a = \max_v a_v \) and \( F_b = \max_v b_v \). With this information, the new optimal sampling rate is given by:

**Theorem 14.** Given \( \epsilon_1 \) and \( \epsilon_2 \), the optimal UBS parameter \((p, q_1)\) and \((p, q_2)\) for \( \text{COUNT} \) in the decentralized setting are given by

\[
p = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\epsilon_1 \epsilon_2 (F_a F_b - F_a - F_b + 1)}\}\}
\]

and \( q_1 = \epsilon_1/p, q_2 = \epsilon_2/p \).

5.2 Sum on Joins

Let \( E_{\text{sum}} \) be the output of the following simplified query:

```sql
select sum(T1.W)
from T1 join T2 on J
```

Let \( F \) be the sum of column \( W \) in the joined samples \( S_1 \bowtie S_2 \). Then, the following is an unbiased estimator for \( E_{\text{sum}} \):

\[
\hat{J}_{\text{sum}} = \frac{1}{p_{\min} q_1 q_2} F
\]

where \( p_{\min} = \min\{p_1, p_2\} \).

**Lemma 15.** \( E[\hat{J}_{\text{sum}}] = E_{\text{sum}} \).

Let \( \mu_v \) and \( \sigma_v^2 \) be respectively the mean and variance of attribute \( W \) of the tuples in \( S_1 \) that have the join value \( v \). Further, recall that \( a_v \) is the number of tuples in \( T_1 \) with join value \( v \). The following lemma gives the variance of \( \hat{J}_{\text{sum}} \).

**Lemma 16.** The variance of \( \hat{J}_{\text{sum}} \) is given by:

\[
\text{Var}[\hat{J}_{\text{sum}}] = \frac{1 - \frac{q_2}{p q_1}}{p_{\min} q_1 q_2} \beta_1 + \frac{1 - q_1}{p q_1 q_2} \beta_2 + \frac{(1 - q_1)(1 - q_2)}{p q_1 q_2} \beta_3 + \frac{1 - \frac{q_2}{p}}{p} \beta_4
\]

where \( \beta_1 = \sum_v a_v^2 \mu_v^2 b_v, \beta_2 = a_v (\mu_v^2 + \sigma_v^2) b_v, \beta_3 = a_v (\mu_v^2 + \sigma_v^2) b_v, \) and \( \beta_4 = a_v^2 \mu_v^2 b_v \).

Analogous to Lemma 10, we have the following result.

**Lemma 17.** Given tables \( T_1, T_2 \) joined on column(s) \( J \), fixed sampling parameters \((p_1, q_1)\) for \( T_1 \), and a fixed effective sampling rate \( \epsilon_2 \leq p_1 \) for \( T_2 \), the variance of \( \hat{J}_{\text{sum}} \) is minimized when \( T_2 \) also uses \( p_1 \) as its universe sampling rate and correspondingly, \( \epsilon_2/p_1 \) as its uniform sampling rate.

5.2.1 Centralized Sampling for Sum

Based on Lemma 17, we use the same universe sampling rate \( p \geq \epsilon_1, \epsilon_2 \) for both tables, with their corresponding uniform sampling rates being \( q_1 = \epsilon_1/p \) and \( q_2 = \epsilon_2/p \). Then we can further simplify equation (5) into:

**Theorem 18.** When \( T_1 \) and \( T_2 \) both use the universe sampling rate \( p \) and respectively use the uniform sampling parameters \( q_1 = \epsilon_1/p \) and \( q_2 = \epsilon_2/p \), the variance of \( \hat{J}_{\text{sum}} \) is given by:

\[
\text{Var}[\hat{J}_{\text{sum}}] = \sum_v \left( (\frac{1}{\epsilon_2} - \frac{1}{p}) \beta_1 + \left( \frac{1}{\epsilon_1} - \frac{1}{p} \right) \beta_2 \right) + \left( \frac{p}{\epsilon_1 \epsilon_2} - \frac{1}{\epsilon_1} - \frac{1}{\epsilon_2} + \frac{1}{p} \right) \beta_3 + (1 - \frac{1}{p}) \beta_4
\]

**Theorem 19.** Given effective sampling rates \( \epsilon_1, \epsilon_2 \), the optimal sampling parameters for \( \text{SUM} \) in a centralized setting are given by

\[
p = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\epsilon_1 \epsilon_2 (\frac{\beta_1 + \beta_2 + \beta_3 - \beta_4}{\beta_3})}\}\}, q_1 = \frac{\epsilon_1}{p}, \text{and } q_2 = \frac{\epsilon_2}{p}
\]

5.2.2 Decentralized Sampling for Sum

Lemma 17 implies that, in a decentralized setting for \( \text{SUM} \) estimation, the universe sampling rate \( p \) must be decided by the party that has \( T_1 \), i.e., the table with the aggregate column.

Given a fixed \( T_1 \) and \( p \), \( \text{Var}[\hat{J}_{\text{sum}}] \) is a strictly convex function of \( T_2 \)'s frequency vector. Hence, the worst case instance is a point distribution where all tuples in \( T_2 \) share the same join key. However, for \( \text{SUM} \), the worst case distributions in \( T_2 \) are not the same for all possible sampling parameters \( p \). Define \( h_v(p) \) to be \( \text{Var}[\hat{J}_{\text{sum}}] \) as a function of \( p \) where \( T_2 \)'s frequency vector is all concentrated on the join key \( v \), and define \( h^*(p) = \max_v h_v(p) \). Since all \( h_v(p) \)'s are convex in \( p \), \( h^*(p) \) is still convex and its exact minimum can be computed using a sweep line algorithm (see [21] §8 for details).

In a nutshell, the algorithm sweeps all possible values of \( p \) and uses a data structure to keep track of \( \max_v h_v(p) \) at that particular value. The data structure uses \( O(|\mathcal{U}|) \) memory, which can be costly in practice.

Therefore, we propose a simple sampling scheme whose worst case variance is at most twice the variance of the optimal scheme. Instead of using \( h^*(p) \) to keep track of the maximum of all \( h_v(p) \), we use an approximate \( h^*(p) = \max\{h_{v_1}(p), h_{v_2}(p)\} \), where \( v_1 = \arg\max_v a_v^2 \mu_v^2 \) and \( v_2 = \arg\max_v a_v (\mu_v^2 + \sigma_v^2) \) to approximate \( h^*(p) \). Since \( h_v(p) \) is a function in the form of \( h(p) = Ap + B/p + C \) for some constant \( A, B, C > 0 \), the value of \( p^* = \arg\min_{1 \leq p \leq \frac{1}{\epsilon_1 \epsilon_2}} h^*(p) \) can be easily solved using quadratic equations and basic case analysis. For more details on the algorithm, refer to Appendix B.

Let \( p^* = \arg\min \hat{h}^*(p) \) and \( p^* = \arg\min h^*(p) \). We claim that choosing \( p^* \) as our sampling parameter can only increase the optimal worst case variance by a factor of 2. This is follows from the simple fact that \( h_{v_1}(p) \) upper bounds the terms in \( h_v(p) \) that depends on \( a_v^2 \mu_v^2 \), and \( h_{v_2}(p) \) upper bounds the terms that depend on \( a_v (\mu_v^2 + \sigma_v^2) \). Hence their maximum is at least half of \( h_v(p) \) for any \( v \) and \( p \).

**Lemma 20.** For any \( p \geq \epsilon_1 \epsilon_2 \), we have \( h^*(p) \leq 2 h^*(p) \).

**Corollary 21.** We have: \( h^*(p) \leq 2 h^*(p) \).

5.3 Average on Joins

Let \( E_{\text{avg}} \) be the output of the following simplified query:

```sql
select avg(T1.W)
from T1 join T2 on J
```

In general, producing an unbiased estimator for \( \text{AVG} \) is hard. Instead, we define and analyze the following estimator. Let \( S \) and \( C \) be the \( \text{SUM} \) and \( \text{COUNT} \) of column \( W \) in \( S_1 \bowtie S_2 \). We define our estimator as \( \hat{J}_{\text{avg}} = S/C \). There are two advantages over using separate samples to evaluate \( \text{SUM} \) and \( \text{COUNT} \): (1) we can use a larger sample to estimate both queries, and (2) since \( \text{SUM} \) and \( \text{COUNT} \) will be positively correlated, the variance of their ratio will be lower. Due to the lack of a close form expression for the variance of the ratio of two random variables, next we present a first order bivariate Taylor expansion to approximate the ratio.

**Theorem 22.** Let \( S \) and \( C \) be random variables denoting the sum and cardinality of the join of two samples produced by applying UBS sampling parameters \((p_1, q_1)\) for \( T_1 \) and \((p_2, q_2)\) for \( T_2 \). Let \( p_{\min} = \min\{p_1, p_2\} \). We have:

\[
\text{Var}[S/C] \approx \left( \frac{E[S]^2}{E[C]^2} \right) \left( \frac{\text{Var}[S]}{E[S]^2} - 2 \frac{\text{Cov}[S, C]}{E[S]E[C]} + \frac{\text{Var}[C]}{E[C]^2} \right)
\]

The denominator, i.e., the size of the sampled join, can even be zero. Furthermore, the expectation of a random variable’s reciprocal is not equal to the reciprocal of its expectation.
where

\[ E[S] = p_{\text{min}} q_1 q_2 \sum_v \mu_v a_v b_v \]

\[ E[C] = p_{\text{min}} q_1 q_2 \sum_v a_v b_v \]

\[ \text{Var}[S] = p_{\text{min}} q_1 q_2 [(1 - q_1) q_1 \sum_v \mu_v^2 b_v + q_2 \sum_v \mu_v (\mu_v^2 + \sigma_v^2) b_v^2 + (1 - q_1) \sum_v \mu_v (\mu_v^2 + \sigma_v^2) b_v] + p_{\text{min}} (1 - p_{\text{min}}) q_1^2 q_2^2 \mu_v^2 b_v^2 \]

\[ \text{Var}[C] = p_{\text{min}} q_1 q_2 [(1 - q_2) q_2 \sum_v a_v^2 b_v + (1 - q_1) q_2 \sum_v a_v b_v^2 + (1 - q_1) (1 - q_2) \sum_v a_v b_v + (1 - p_{\text{min}}) q_1 q_2 \sum_v a_v^2 b_v^2] \]

\[ \text{Cov}[S, C] = p_{\text{min}} q_1 q_2 [(1 - q_1) q_1 \sum_v \mu_v a_v b_v + (1 - q_1) q_2 \sum_v \mu_v a_v b_v^2 + (1 - q_1) (1 - q_2) \sum_v \mu_v a_v b_v + (1 - p_{\text{min}}) q_1 q_2 \sum_v a_v^2 b_v b_v^2] \]

5.3.1 Centralized Sampling for Average

In the centralized setting, where \( a_v, b_v, \mu_v \) and \( \sigma_v \) values are given for all \( v \), every term in the expression \( \frac{\text{Var}[S]}{E[S]^2} \frac{1}{E[C]} \) that depends on \( p \) is proportional to either \( p \) or \( 1/p \). The terms proportional to \( 1/p \) are \( 1/(A - 2B + C) \) where

\[ A = \sum_v a_v (\mu_v^2 + \sigma_v^2) b_v + \sum_v a_v^2 b_v b_v \]

\[ B = \sum_v a_v b_v + \sum_v \mu_v b_v + \sum_v \mu_v a_v b_v + \sum_v \mu_v^2 b_v \]

\[ C = \sum_v a_v b_v^2 + \sum_v a_v^2 b_v^2 \]

The terms proportional to \( p \) are \( pD \) where

\[ D = \frac{1}{\epsilon_2 \epsilon_2} \sum_v a_v (\mu_v^2 + \sigma_v^2) b_v + \frac{1}{\epsilon_2} \sum_v a_v b_v b_v - \sum_v a_v b_v - \sum_v a_v b_v b_v \]

We can find a \( p \) that minimizes \( \frac{1}{p} (A - 2B + C) + pD \) as follows.

Theorem 23. In the centralized setting, set \( p^- = \max\{\epsilon_1, \epsilon_2\} \), \( p^+ = 1 \) and \( p^* = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{A - 2B + C}/B\}\} \). Then the optimal sampling parameter is given by:

\[ p = \begin{cases} 
  p^- & \text{if } A - 2B + C \leq 0 \text{ and } D > 0 \\
  p^+ & \text{if } A - 2B + C > 0 \text{ and } D \leq 0 \\
  \arg\min_{p \in (p^-, p^+)} \frac{1}{p} (A - 2B + C) + pD & \text{otherwise}
\end{cases} \]

5.3.2 Decentralized Sampling for Average

Minimizing the worst case variance for AVG (for the decentralized setting) is much more involved than the average case. In most cases, the objective function (variance) is neither convex nor concave in \( T_q \)'s frequencies. However, note that every term in Theorem 23 is an inner product \( (x, y) \), where \( x \) and \( y \) are two vectors stored on part 1 and party 2, respectively. Fortunately, inner products can be approximated with transferring a very small amount of information using the AMS sketch [11, 24]. With such a sketch, we can derive an approximate sampling rate without communicating the full frequency statistics.

6. MULTIPLE QUERIES AND FILTERS

Creating a separate sample for each combination of aggregation function, aggregation column, and WHERE clause is clearly impractical. In this section, we show how to create a single sample per join pattern that supports multiple queries at the cost of some possible loss of approximation quality. First, we ignore the WHERE clauses and then show how they can be handled too.

In essence, one can take the union of multiple universe samples to form a single sample that is simultaneously useful for various join patterns. However, the resulting sample size depends heavily on inter-column correlations and, in the worse case, can be as large as creating a separate universe sample for each join pattern. To remove the need for capturing correlations, here we present a formulation that creates a separate sample for every pair of table and join key (a join key can itself be one or more columns).

Multiple Tables and Queries—We formulate our input as a graph \( G= (V, E) \). The vertex set \( V \) is the set of all table and join key pairs, and the edge set \( E \) corresponds to all join queries of interest. Specifically, for every join query between tables \( T_1 \) and \( T_2 \) on \( J_1 = J_2 \), we have a corresponding edge \( e \) between vertices \( (T_1, J_1) \in V \) and \( (T_2, J_2) \in V \) (henceforth, we will use a query and its corresponding edge interchangeably). This means \( G \) is a multigraph, where potentially parallel edges or self-loops. For each vertex \( v = (T, J) \in V \), we must output a sampling budget \( \epsilon_v \) as well as the corresponding universe sampling rate \( p_v \), which will be used to create a sample \( S = UBS_{p_v, \epsilon_v}(T, J) \). This sample will be used for any query that involves a join with \( T \) on column(s) \( J \).

According to Lemma 9 Lemma 16, and Theorem 22, for each edge \( e = (v_1, v_2) \in E \), we can express the estimator variance of its corresponding query as a function of \( \epsilon_1, \epsilon_2, p_{v_1}, p_{v_2}, p_v \), and \( p_v \) is an auxiliary variable denoting the minimum of \( p_{v_1} \) and \( p_{v_2} \):

\[ f(e, p_{v_1}, p_{v_2}, p_{v_1}, p_{v_2}) = \frac{1}{p_v} (A + B + C p_{v_1} + D p_{v_2}) \]

(7)

where \( A, B, C, D \) are constants that depend on the distributional information of the tables in \( v_1 \) and \( v_2 \). To cast this as an optimization problem, we also take in a user specified weight \( \omega_e \) for each edge \( e \) and express our objective as:

\[ F = \sum_{e = (v_1, v_2) \in E} \omega_e f(e, p_{v_1}, p_{v_2}, p_{v_1}, p_{v_2}) \]

(8)

The choice of \( \omega_e \) values is up to the user. For example, they can be all set to 1, or to the relative frequency, importance, or probability of appearance (e.g., based on past workloads) of the query corresponding to \( e \). Then, to find the optimal sampling parameters we solve the following optimization:

\[ \min_{e \in (v_1, v_2) \in E} F \text{ subject to } \sum_{v = (T, J) \in V} \epsilon_v \cdot \text{size}(T) \leq B \]

(9)

where \( \text{size}(T) \) is the storage footprint of table \( T \), and \( B \) is the overall storage budget for creating samples. Note that by replacing the non-linear \( p_e = \min(p_{v_1}, p_{v_2}) \) constraints with \( p_e \leq p_v \), and
\(p_e \leq p_{x \leq C} \) is reduced to a smooth optimization problem, which can be solved numerically with off-the-shelf solvers [14].

**Known Filters**— To incorporate \textit{WHERE} clauses, we simply regard a query with a filter \( c \) on \( T_1 \bowtie T_2 \) as a query without a filter but on a sub-table that satisfies \( c \), namely \( T' = \sigma_c(T_1 \bowtie T_2) \).

**Unknown Filters with Distributional Information**— When the set of columns appearing in the \textit{WHERE} clause can be predicted but the exact constants are unknown, a similar technique can be applied. For example, if an equality constraint \( C > x \) is anticipated but \( x \) may take on 100 different values, we can conceptually treat it as 100 separate queries each with a different value of \( x \) in its \textit{WHERE} clause. This reduces our problem to that of sampling for multiple queries without a \textit{WHERE} clause, which we know how to handle using equation (3).\(^{10}\) Here, the weight \( \omega_i \) can be used to exploit any distributional information that might be available. In general, \( \omega_i \) should be set to reflect the probability of each possible \textit{WHERE} clause appearing in the future. For example, if there \( R \) possible \textit{WHERE} clauses and are equally likely, we can set \( \omega_i = 1/R \), but if popular values in a column are more likely to appear in the filters, we can use the column’s histogram to assign \( \omega_i \).

**Unknown Filters**— When there is no information about the columns (or their values) in future filters, we can take a different approach. Since the estimator variance is a monotone function in the frequencies of each join key (see Theorem [11], Theorems [18] and [22]), the larger the frequencies, the larger the variance. This means the worst case variance always happens when the \textit{WHERE} clause selects all tuples from the original table. Hence, in the absence of any distributional information regarding future \textit{WHERE} clauses, we can simply focus on the original query without any filters to minimize our worst case variance.

### 7. EXPERIMENTS

Our experiments aim to answer the following questions:

(i) How does our optimal sampling compare to other baselines in centralized and decentralized settings? (§7.2, §7.3)

(ii) How well does our optimal UBS sampling handle join queries with filters? (§7.4)

(iii) How does our optimal UBS sampling perform when using a single sample for multiple queries? (§7.5)

(iv) How does our optimal UBS sampling compare to existing stratified sampling strategies? (§7.6)

(v) How much does a decentralized setting reduce the resource consumption and sample creation overhead? (§7.7)

#### 7.1 Experiment Setup

**Hardware and Software**— We borrowed a cluster of 18 c220g5 nodes from CloudLab [3]. Each node was equipped with an Intel Xeon Silver 4114 processor with 10 cores (2.2Ghz each) and 192GB of RAM. We used Impala 2.12.0 as our backend database to store data and execute queries.

**Datasets**— We used several real-life and synthetic datasets:

1. **Instacart** [1]. This is a real-world dataset from an online grocery. We used their orders and order_products tables (3M and 32M tuples, resp.), joined on order_id.

   \(^{10}\)Note that, even though each query in this case is on a different table, they are all sub-tables of the same original table, and hence their sampling rate \( p \) is the same.

2. **Movielens** [32]. This is a real-world movie rating dataset. We used their ratings and movies tables (27M and 58K tuples, resp.), joined on movieid.

3. **TPC-H** [4]. We used a scale factor of 1000, and joined the fact and largest dimension tables. Specifically, we joined \( l\text{orderkey} \) of the lineitem table with \( o\text{orderkey} \) of the orders table (6B and 1.5B tuples, resp.).

4. **Synthetic**. To better control the join key distribution, we also generated several synthetic datasets, where tables \( T_1 \) and \( T_2 \) each had 1M tuples and a join column \( J \). \( T_1 \) had an additional column \( W \) for aggregation, drawn from a power law distribution with range [1, 1000]. We varied the distribution of the join key in each table to be one of uniform, normal, or power law, creating 9 different datasets. The values of column \( J \) were integers randomly drawn from [1, 1000] according to the chosen distribution. For normal distribution, we used a truncated distribution with \( \sigma=1000/5 \). For power law, we used \( \alpha=3.5 \) for both \( J \) and \( W \). We refer to these datasets by their tables’ distributions as \( S\{\text{distribution of } T_1,\text{distribution of } T_2\} \), e.g., \( S\{\text{uniform,uniform}\} \).

**Baselines**— We compared our optimal UBS parameters (referred to as OPT) against six baselines. The UBS parameters of these baselines, \( B_1, \ldots, B_6 \), are listed in Table 2. \( B_1 \) and \( B_6 \) are simply universe and uniform sampling, respectively. \( B_2, \ldots, B_5 \) represent different hybrid variants of these sampling schemes. Sampling budgets were \( \epsilon_1 = \epsilon_2 = 0.01 \), except for TPC-H and Movielens where we used 0.001 and 0.1 due to their large and small number of tuples, respectively.

**Implementation**— We implemented our optimal parameter calculations in Python application. Our sample generation logic read required information, such as table size and join key frequencies, from the database, and then constructed SQL statements to build appropriate samples in the target database. We used Python to compute approximate answers from sample-based queries.

**Variance Calculations**— We generated \( \beta=500 \) pairs of samples for each experiment, and re-ran the queries on each pair, to calculate the variance of our approximations.

#### 7.2 Join Approximation: Centralized Setting

Table 3 shows the sampling rates used by OPT for different datasets and aggregate functions in the centralized setting. For Synthetic, the optimal parameters were a full uniform sampling for COUNT and SUM, and a full universe sampling for AVG in most cases.
The only exceptions were COUNT and SUM for $S\{\text{power law}, \text{power law}\}$ where the optimal parameters were some mixture of uniform and universe sampling. This is due to the higher probability of missing extremely popular join keys with universe sampling. To the contrary, for AVG, OPT reduced to a simple universe sampling. This is because maximizing the output size in this case was the best way to reduce variance. For the other datasets (Instacart, Movielens, and TPC-H), the optimal parameters led to universe sampling, regardless of aggregate type, and their joins were PK-FK, hence making uniform sampling less useful for the table with primary keys.

Figure 2 shows OPT's improvement over the baselines in terms of variance for COUNT queries. Each bar is also annotated with the relative percentage error of the corresponding baseline. OPT outperformed all baselines in most cases, achieving over 10x lower variance than the worst baseline in several scenarios. Figures 2 and 3 show the same experiment for SUM and AVG. In both cases, OPT achieved the minimum variance across all sampling strategies, except for AVG when $T_1$ or $T_2$ was a power law distribution. This is because OPT for AVG was calculated from an approximation, rather than a closed-form solution, unlike COUNT or SUM, as discussed in Section 7.1. Furthermore, we deliberately randomized both the join key and aggregate columns in Synthetic to create a challenging setting. This, combined with the power law distribution, made it difficult to correctly estimate variances from generated samples. However, UBS with OPT still achieved lowest variance for all aggregates on the real-world datasets Instacart and Movielens, as shown in Figure 5. For the selected join key, OPT determined that a full universe sampling was the best sampling scheme, achieving the minimum variance among the baselines.

In summary, this experiment highlights OPT's ability in outperforming simple uniform or universe sampling—or choosing one of them, when optimal—for aggregates on joins.

### 7.3 Join Approximation: Decentralized

We evaluated both OPT and other baselines under a decentralized setting using Instacart and Synthetic datasets. Here, we constructed a possible worst case distribution for $T_2$ that was still somewhat realistic, given the distribution of $T_1$ and minimal information about $T_2$ (i.e., $T_2$'s cardinality). To do this, we used the following steps: 1) let $J_{MAX(T_1)}$ be the most frequent join key value in $T_1$; 2) assign 75% of join key values of $T_2$ to have the value of $J_{MAX(T_1)}$ and draw the rest of join key values from a uniform distribution.

Figure 6 shows the results. For Synthetic, the OPT was the same under both settings whenever there was a power law distribution or the aggregate was AVG. This is because our assumption of the worst case distribution for $T_2$ was close to a power law distribution. For COUNT and SUM with Synthetic dataset, OPT in the decentralized setting had a much higher variance than OPT in the centralized setting when there was no power law distribution. With Instacart, OPT in the decentralized setting was same as OPT in the centralized setting, which had the minimum variance among the baselines. This illustrates that OPT in the decentralized setting can perform well with real-world data where the joins are mostly PK-FK. This also shows that if a reasonable assumption is possible on the distribution of $T_2$, OPT can be as effective in the decentralized setting as it is in a centralized one, while requiring significantly less communication.

### 7.4 Join Approximation with Filters

To study OPT's effectiveness in the presence of filters, we used $S\{\text{uniform}, \text{uniform}\}$ and Instacart datasets. We added an extra column $C$ to $T_2$ in $S\{\text{uniform}, \text{uniform}\}$, with integers values in $[1, 100]$, and tried three distributions (uniform, normal, power law).

For Instacart, we used the $\text{order}_\text{hour}_\text{of}_\text{day}$ column for filtering, which had an almost normal distribution. We used an equality operator and chose the comparison value $x$ uniformly at random. We calculated the average variance over all possible values of $x$.

Table 4 shows the optimal sampling parameters for $S\{\text{uniform}, \text{uniform}\}$ for different distributions of the filtered column $C$.

<table>
<thead>
<tr>
<th>Dist. of $C$</th>
<th>COUNT</th>
<th>SUM</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>$p$</td>
<td>$q$</td>
<td>$p$</td>
</tr>
<tr>
<td>Normal</td>
<td>$0.010$</td>
<td>$1.000$</td>
<td>$0.001$</td>
</tr>
<tr>
<td>Power law</td>
<td>$0.051$</td>
<td>$0.195$</td>
<td>$0.050$</td>
</tr>
</tbody>
</table>

Table 4: Optimal sampling parameters for $S\{\text{uniform}, \text{uniform}\}$ for different distributions of the filtered column $C$.

Figure 7 shows OPT's improvement over baselines in terms of average variance. Again, OPT successfully achieved the lowest average vari-
Figure 4: OPT’s improvement in terms of variance for AVG over six baselines with synthetic dataset (percentages are relative error).

Figure 5: OPT’s improvement in terms of variance over six baselines with Instacart, Movielens and TPC-H (percentages are relative error).

Figure 6: Variances of the query estimators for OPT in the centralized and decentralized settings.

Figure 7: OPT’s improvement in terms of the estimator’s variance over six baselines in the presence of filters.

Figure 8: Variance of the query estimators for OPT (individual) and OPT (combined) for the \texttt{S\{normal,normal\}} dataset.

Table 5: Sampling parameters \((p, q)\) of OPT using individual samples for different aggregates versus a combined sample \((\texttt{S\{normal,normal\}})\) dataset.

7.5 Combining Samples

We evaluated the idea of using a single sample for multiple queries instead of generating individual samples for each query, as discussed in Section 6. Here, we use OPT (individual) and OPT (combined) to denote the use of one-sample-per-query and one-sample-for-multiple-queries, respectively. For OPT (combined), we considered a scenario where each of COUNT, SUM, and AVG is equally likely to appear. Table 5 reports the sampling rates chosen in each case. As shown in Figure 8, without having to generate an individual sample for each query, the variances of OPT (combined) were only slightly higher than those of OPT (individual). This experiment shows that it is possible to create a single sample for multiple queries without sacrificing too much optimality.

7.6 Stratified Sampling

We also evaluated SUBS for join queries with group-by. Here, we used the \texttt{S\{normal,normal\}} dataset, and added an extra group column \texttt{C} to \texttt{T}_1 with integers from 0 to 9 drawn from a power law distribution with \(\alpha = 1.5\). This time we did not randomize the
7.7 Overhead: Centralized vs. Decentralized

We compared the overhead of OPT in centralized versus decentralized settings, in terms of the sample creation time and resources, such as network and disk. OPT should have a much higher overhead in the centralized setting, as it requires full frequency information of every join key value in both tables. To quantify their overhead difference, we used Instacart and TPC-H, and created a pair of samples for SUM in each case. Here, the aggregation type did not matter as the time spent calculating \( p \) and \( q \) was negligible compared to the time taken by transmitting the frequency vectors.

As shown in Figure 10, we measured the time for statistics acquisition, sampling rate calculation, and sample table creation. Here, the time taken by collecting the frequencies was the dominant factor. For Instacart, it took 65.16 secs from start to finish in the centralized setting, compared to 99.98 secs in the centralized setting, showing a 1.53x improvement in time. For TPC-H, it took 59.5 min in decentralized setting, compared to 97.7 mins of the centralized setting, showing a speedup of 1.54x.

We also measured the total network and disk I/O usage across the entire cluster, as shown in Figure 11. For Instacart, compared to the decentralized setting, the centralized one used 3.66x (0.9 MB → 3.29 MB) more network and 2.22x (7.59 MB → 16.9 MB) more disk bandwidth. Overall, the overhead was less for TPC-H. The centralized in this case used 1.38x (243.39 MB → 337.04 MB) more network and 1.49x (519.03 MB → 776.58 MB) more disk bandwidth than the decentralized setting.

This experiment shows the graceful tradeoff between the optimality of sampling and its overhead, making the decentralized variant an attractive choice for large datasets and distributed systems.

7.8 UBS vs. Two-Level Sampling

Two-level sampling (2LV) is similar to our UBS scheme in that it also applies stratified sampling before Bernoulli sampling. However, unlike UBS which applies the same sampling rate to all tuples, 2LV uses a different universe sampling rate for each join key, i.e., 2LV is strictly more expressive than UBS. Thus, by using significantly more parameters (i.e., number of distinct join keys), 2LV should be able to achieve a lower variance than any UBS scheme (which uses only two parameters \( p \) and \( q \)). To empirically measure this gap, we compared the relative error of 2LV versus OPT. Since 2LV is originally designed for cardinality estimation, we compared their COUNT error on our synthetic datasets (since COUNT is symmetric, we only have 6 combinations), with 100K tuples, 100 keys, and \( \epsilon=1\% \) budget. Here, we used As shown in Figure 12, 2LV’s error was slightly lower than OPT, and both errors increased with the skew in the join keys, e.g., when a power law was involved. This was as expected: having a separate parameter for each join key means more complexity, but also allows 2LV to better adapt to the distribution of the data.

8. RELATED WORK

Online Sample-based Join Approximation—Ripple Join is an online join algorithm that operates under the assumption that the tuples of the original tables are processed in a random order. Each time, it retrieve a random tuple (or a set of random tuples) from the tables and then joins the new tuples with the previously read tuples and with each other. SMS speeds up the hashed version of Ripple Join when hash tables exceed memory. Wander Join tackles the problem of \( k \)-way chain join and eliminates the random order requirement of Ripple Join. However, it requires an index on every join column in each of the tables. Using indexes, Wander Join performs a set of random walks and obtains a non-uniform but independent sample of the join. Maintaining an
approximation of the size of all partial joins can help overcome the non-uniformity problem [41, 58].

**Offline Sample-based Join Approximation**— AQUA [6] acknowledges the quadratic reduction and the non-uniformity of the output when joining two uniform random samples. The same authors propose Join Synopses [7], which computes a sample of one of the tables and joins it with the other tables as a sample of the actual join. Chaudhuri et al. [17] also point out that a join of independent samples from two relations does not yield an independent sample of their join, and propose use precomputed statistics to overcome this problem. However, their solution requires collecting full frequency information of the relation, which can be quite expensive. Zhao et al. [58] provide a better trade-off between sampling efficiency and the join size upper bound. Hashed sampling (a.k.a. universe) [31] is proposed in the context of selectivity estimation for set similarity queries. Block-level uniform sampling [16] is less accurate but more efficient than tuple-level sampling. Bi-level sampling [19, 50] performs Bernoulli sampling at both the block- and tuple-levels, as a trade-off between accuracy and I/O cost of sample generation.

**AQP Systems on Join**— Most AQP systems rely on sampling and support certain types of joins [6, 9, 15, 27, 36, 41, 47, 49]. STRAT [15] discusses the use of uniform and stratified sampling, and how these can support certain types of join queries. More specifically, STRAT only supports PK-FK joins between a fact table and one or more dimension table(s). BlinkDB [9] extends STRAT and considers multiple stratified samples instead of a single one. As previously mentioned, AQUA [6] supports foreign key joins using join synopses. Icicles [27] is a new class of samples that includes tuples that are more likely to be required by future queries based on past workloads. These samples, similar to AQUA, only support foreign key joins. PF-OLA [49] is a framework for parallel online aggregation. It studies parallel joins with group-bys, when partitions of the two tables fit in memory. XDB [41] integrates Wander Join in PostgreSQL. Quicr [56] does not create offline samples. Instead, it uses universe sampling to support equi-joins, where the group-by columns and the value of aggregates are not correlated with the join keys. VerdictDB utilizes a technique called variational subsampling, which creates subsamples of the sample such that it only requires a single join—instead of repeatedly joining the subsamples multiple times—to produce accurate aggregate approximations.

**Join Cardinality Estimation**— There is extensive work on join cardinality estimation (i.e., `COUNT(*)`) in the database community [10, 26, 38, 39, 48, 53, 54, 56] as an important step of the query optimization process for joins. Two-level sampling [18] first applies universe sampling to the join values, and then, for each join value sampled, it performs Bernoulli sampling. However, unlike our UBS scheme which applies the same rate to all keys, two-level sampling uses a different rate during its universe sampling for each join key. In other words, two-level sampling is a more complex scheme with significantly more parameters than UBS (which requires only two parameters, `p` and `q`), and is thus less amenable to efficient and decentralized implementation. Two-level sampling also differs from bi-level sampling [30] in that it applies two different sampling methods, whereas bi-level sampling uses only Bernoulli sampling but at different granularity levels. End-biased sampling [26] samples each tuple with a probability proportional to the frequency of its join key value. Index-based sampling is shown to improve cardinality estimation for main-memory databases [39]. Recently, deep learning is utilized to learn inter-table correlations and improve cardinality estimates [38].

**Theoretical Studies**— The question about the limitation of sample-based approximation of joins, to the best of our knowledge, has not been asked in the theory community. However, the past work in communication complexity on set intersection and inner product estimation has implications for join approximation. In this problem, two parties possess two dimensional vectors `x` and `y` and they wish to compute their inner product `t = ⟨x, y⟩` with as little information exchange as possible. and sends it to Bob, who will in turn estimates `⟨x, y⟩` using `y` and `β(x)`. For this problem, [45] shows that any estimator produced by `s` bits of communication has variance at least `Ω(dt/s)`. Estimating inner product for 0, 1 vectors is directly related to estimating `SUM` and `COUNT` for a PK-FK join. A natural question is whether join is still hard even if frequencies are all larger than 1. Further, the question of whether estimating `AVG` is also hard is not answered by prior work.

9. CONCLUSION

The goal of this paper is to improve our understanding of join approximation using offline samples, and formally address some of the key open questions faced by practitioners using and building AQP engines. We defined generic sampling schemes that cover the most commonly used sampling strategies, as well as their combinations. Within these schemes, we (1) provided an informational-theoretical lower bound on the lowest error achievable by any offline sampling scheme, (2) derived optimal strategies that match this lower bound within a constant factor, and (3) offered a decentralized variant that requires minimal communication of statistics across the network. Finally, we empirically validated our findings—and the optimality of our sampling scheme—through extensive experiments on multiple datasets.

10. REFERENCES


APPENDIX

A. OMITTED PROOFS

Theorem 1. No sampling scheme with sample rate \( \alpha \) can guarantee more than \( \alpha \) fraction of \( T_1 \bowtie T_2 \) in expectation for all possible inputs.

Proof. We can simply consider two identical tables \( T_1, T_2 \) of \( n \) tuples, each having join key 1, 2, ..., \( n \). Their join has size \( n \). Since each tuple of \( T_1 \) joins with exactly one tuple of \( T_2 \), the size of the join of the samples must not be larger than the size of sample of \( T_1 \). Since, by assumption, the expected size of the sample of \( T_1 \) is at most \( \alpha n \), the expected size of the join of the samples must also be at most \( \alpha n \).

Lemma 9. Let \( S_1 = UBS_{p, q_1}(T_1, J) \) and \( S_2 = UBS_{p, q_2}(T_2, J) \). The variance of \( J_{\text{count}} \) is as follows:

\[
\Var(J_{\text{count}}) = \frac{1 - p}{p} \gamma_{2, 2} + \frac{1 - q_2}{pq_2} \gamma_{2, 1}
\]

\[
+ \frac{1 - q_2}{pq_2} \gamma_{1, 1} + \frac{1 - q_1}{pq_1 q_2} \gamma_{1, 1}.
\]

where \( \gamma_{i,j} = \sum_v a_i^v b_j^v \).

Proof. Let \( X_v \) and \( Y_v \) be the random variable denoting the number of tuple in \( S_1 \) and \( S_2 \) with value \( v \) given that \( h(v) \leq p_{\text{min}} \). Therefore, \( X_v \) and \( Y_v \) are binomial random variables with parameter \( (a_v, q_1) \) and \( (b_v, q_2) \). Let \( Z_v \) be defined as the number of tuples in \( S_1 \bowtie S_2 \) with join value \( v \). By construction, we have:

\[
Z_v = \begin{cases} 
X_v Y_v & \text{with probability } p \\
0 & \text{otherwise}
\end{cases}
\]

And

\[ J = \frac{1}{pq_1 q_2} \sum_v Z_v. \]

To analyze the variance of \( Z_v \), we use the law of total variance:

Let \( W_v = X_v Y_v \), we have

\[
\Var(Z_v) = \E[\Var(Z_v | W_v)] + \Var(E[Z_v | W_v])
\]

We have

\[
E[\Var(Z_v | W_v)] = p(1 - p)E[W_v^2] - E(\Var(W_v))
\]

And

\[
\Var(E[Z_v | W_v]) = p^2 \Var(W_v)
\]

Combining the two terms we have

\[
\Var(Z_v) = p \Var(W_v) + p(1 - p)E^2[\Var(W_v)]
\]

where

\[
\Var(W_v) = E[(X_v^2 - E[X_v]^2)] = (a_v q_1 (a_v + 1) - q_1 a_v^2) - q_1 a_v^2 b_v
\]

\[
= a_v^2 b_v q_1 (1 - q_2) + a_v b_v q_1 (1 - q_1) q_2^2
\]

\[
+ a_v b_v q_1 (1 - q_1) q_2^2 = a_v^2 b_v q_1 (1 - q_2) + a_v b_v q_1 (1 - q_1) q_2^2 + a_v b_v q_1 (1 - q_1) q_2^2
\]

Using our notation for frequency vectors and its moments. The variance of the join size estimator \( J \) can be written as:

\[
\Var(J_{\text{count}}) = \frac{1}{pq_1 q_2} \sum_v \Var(Z_v)
\]

\[
= \frac{1 - p}{p} \sum_v a_v^2 b_v + \frac{1 - q_2}{pq_2} \sum_v a_v b_v
\]

\[
+ \frac{1 - q_1}{pq_1 q_2} \sum_v a_v b_v
\]

\[
+ \frac{1 - q_1}{pq_1 q_2} \sum_v a_v b_v
\]

\[
= \frac{1 - p}{p} \sum_v a_v^2 b_v + \frac{1 - q_1}{pq_2} \sum_v a_v b_v
\]

\[
+ \frac{1 - q_1}{pq_1 q_2} \sum_v a_v b_v
\]

Lemma 10. Given tables \( T_1, T_2 \) joined on column(s) \( J \), a fixed sampling parameter \( (p_1, q_1) \) for \( T_1 \), and a fixed effective sampling rate \( \epsilon_2 \) for \( T_2 \), the variance of \( J_{\text{count}} \) is minimized when \( T_1 \) uses \( p_1 \) as its universe sampling rate and correspondingly \( \epsilon_2 / p_1 \) as its uniform sampling rate.

Proof. Define \( p_2 \) and \( q_2 \) to be the universe and Bernoulli sampling rate for \( T_2 \) and \( p = \min(p_1, p_2) \). For fixed \( p_1, q_1 \) and \( \epsilon_2 \), we write \( \Var(J_{\text{count}}) \) as a function of \( p_2 \).

If \( p_2 \geq p_1 \), we have \( p = p_1 \):

\[
\Var(J_{\text{count}}) = \frac{1 - p}{p} \sum_v a_v^2 b_v + \frac{1 - q_1}{pq_1} \sum_v a_v b_v
\]

\[
+ \frac{1 - q_1}{pq_1 q_2} \sum_v a_v b_v
\]

\[
= \frac{1 - p}{p} \sum_v a_v^2 b_v + \frac{1 - q_1}{pq_1} \sum_v a_v b_v
\]

\[
+ \frac{1 - q_1}{pq_1 q_2} \sum_v a_v b_v
\]

which is nondecreasing in terms of \( p_2 \). Therefore, the minimum is attained when \( p_2 = p_1 \). Intuitively, increasing \( p_2 \) when \( p_2 \geq p_1 \) decreases the Bernoulli sampling rate without increasing the universe sampling rate over join, and hence only decrease the quality of samples.
When $p_2 \leq p_1$, $p = p_2$:

$$\text{Var}(\hat{J}_{\text{count}}) = \frac{1 - p}{p} \sum_v a_v^2 b_v^2 + \frac{1 - q_2}{pq_2} \sum_v a_v^2 b_v$$

$$+ \frac{1}{q_1} \sum_v a_v b_v + \frac{(1 - q_1)(1 - q_2)}{pq_1q_2} \sum_v a_v b_v$$

$$= \left(\frac{1}{p_2} - 1\right) \sum_v a_v^2 b_v^2 + \frac{1}{p_2} \sum_v a_v^2 b_v$$

$$+ \frac{1 - q_1}{q_1} \sum_v a_v b_v + \frac{1 - q_1}{(1 - q_2) p_2} \sum_v a_v b_v$$

$$= \frac{1}{p_2} \sum_v (a_v^2 b_v^2) + \frac{1 - q_1}{q_1} \sum_v (a_v b_v - a_v b_v)$$

$$= \frac{1}{p_2} \sum_v (a_v^2 b_v^2 - a_v^2 b_v - a_v b_v) - \frac{1 - q_1}{q_1} \sum_v a_v b_v$$

Since $0 < q_1 \geq 1$, and $a_v^2 b_v^2 - a_v^2 b_v - a_v b_v \geq 0$ since both $a_v$ and $b_v$ are nonnegative integers. The variance function is nonincreasing in terms of $p_2$, thereby attains its minimum when $p_2 = p_1$.

Theorem 11. When $T_1$ and $T_2$ use sampling parameters ($p, \epsilon_1/p$) and ($p, \epsilon_2/p$), $\hat{J}_{\text{count}}$’s variance is given by:

$$\text{Var}(\hat{J}_{\text{count}}) = \left(\frac{1}{p} - 1\right) \gamma_{2,2} + \frac{1}{\epsilon_2} - \frac{1}{p} \gamma_{2,1}$$

$$+ \left(\frac{1}{\epsilon_1} - \frac{1}{p}\right) \gamma_{1,2} + \left(\frac{p}{\epsilon_1 \epsilon_2} - \frac{1}{\epsilon_1} - \frac{1}{p}\right) \gamma_{1,1}.$$

Proof. This is simply obtained by plugging in $q_1 = \epsilon_1/p$ and $q_2 = \epsilon_2/p$ to Theorem 12.

Theorem 12. Let $T_1$ and $T_2$ be two tables joined on column(s) $J$. Let $a_v$ and $b_v$ be the frequency of value $v$ in column(s) $J$ of tables $T_1$ and $T_2$, respectively. Given their sampling rates $\epsilon_1$ and $\epsilon_2$, the optimal sampling parameters ($p_1, q_1$) and ($p_2, q_2$) are given by:

$$p_1 = p_2 = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\frac{\epsilon_1 \epsilon_2 \gamma_{2,2} - \gamma_{2,1} - \gamma_{1,1}}{\gamma_{1,1}}}\}\}$$

and $q_1 = \epsilon_1/p$, $q_2 = \epsilon_2/p$.

Proof. By Lemma 10, the table has equal universe sampling rate in the optimal sampling scheme. Thus we assume $p_1 = p_2 = p$ and $p$ is a real number between $\max\{\epsilon_1, \epsilon_2\}$ and 1, and $q_1 = \epsilon_1/p$ and $p_2 = \epsilon_2/p$. Thus we have:

$$\text{Var}(\hat{J}_{\text{count}}) = \frac{1}{p} \sum_v a_v^2 b_v^2 + \frac{1 - q_1}{\epsilon_1} \sum_v a_v^2 b_v$$

$$+ \frac{1}{\epsilon_1} \sum_v a_v b_v + \frac{1 - q_1}{\epsilon_1} \sum_v a_v b_v$$

$$= \frac{1}{p} \sum_v a_v^2 b_v^2 - \frac{1}{\epsilon_1} \sum_v a_v b_v^2 - \sum_v a_v b_v^2 + \sum_v a_v b_v$$

$$+ \frac{1}{\epsilon_1} \sum_v a_v b_v - \frac{1}{\epsilon_1} \sum_v a_v b_v + \frac{1}{\epsilon_1} \sum_v a_v b_v$$

Notice only the first two terms $\frac{1}{p} \sum_v a_v^2 b_v^2 - \sum_v a_v b_v^2 - \sum_v a_v^2 b_v^2$ and $\sum_v a_v^2 b_v^2 + \sum_v a_v^2 b_v^2$ in $\hat{J}_{\text{count}}$ depends on $q$. Moreover, both terms has nonnegative coefficients:

$$\sum_v a_v^2 b_v^2 - \sum_v a_v b_v^2 - \sum_v a_v^2 b_v + \sum_v a_v b_v$$

$$= \sum_v (a_v^2 - a_v)(b_v^2 - b_v)$$

$\geq 0$

($a_v, b_v$ are nonnegative integers.)

Since $p$ takes on value between $\max\{\epsilon_1, \epsilon_2\}$ and 1, by AM-GM inequality and monotonicity of the variance function, the term is minimized when

$$p = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\epsilon_1 \epsilon_2 (F_a F_b - F_a + F_b + 1)}\}\}$$

Lemma 13. Let $a_v$ be the maximum frequency in table $T_1$, $v_2$ be any value that has that frequency, and $n_v$ be the total number of tuples in $T_2$. The optimal value for the problem $\max_{b \in \mathbb{K}} \text{Var}(J_{\text{count}})$ is given by ($\frac{1}{p} - 1) a_v^2 b_v^2 + \left(\frac{1 - 1}{\epsilon_2} - \frac{1}{\epsilon_1}\right) a_v n_v + \left(\frac{1 - 1}{\epsilon_2} - \frac{1}{\epsilon_1}\right) a_v n_v$.

Proof. Since $\text{Var}(J_{\text{count}})$ is strictly convex as a function $b_v$’s in its domain. To maximize this function, it suffices to consider only the extreme points in its feasible polytope. These are the all 0 vector 0, and the vector $b_v$ for each join key $v \in \mathcal{U}$ that has $n_v$ at its $v$-th entries and 0 everywhere else. It is easy to see that since all coefficients for are positive, the maximizing value is achieved by the vector $b_v$.

Theorem 14. Given $\epsilon_1$ and $\epsilon_2$, the optimal UBS parameter $(p, q_1)$ and $(p, q_2)$ for COUNT in the decentralized setting are given by

$$p = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\epsilon_1 \epsilon_2 (F_a F_b - F_a + F_b + 1)}\}\}$$

and $q_1 = \epsilon_1/p$, $q_2 = \epsilon_2/p$.

Proof. Consider the variance of our count estimator given by Theorem 12

$$\text{Var}(\hat{J}_{\text{count}}) = \frac{1}{p} \sum_v a_v^2 b_v^2 + \frac{1}{\epsilon_1} \sum_v a_v^2 b_v$$

$$+ \frac{1}{\epsilon_1} \sum_v a_v b_v + \frac{1}{\epsilon_1} \sum_v a_v b_v$$

Fixed any $p$, under the constraint that for all $v$, $a_v \leq F_a$ and $b_v \leq F_b$, the variance is maximized when $a_v = F_a$ and $b_v = F_b$. This defines the worst case input under such constraint. So to obtain the optimum sampling rate for the worst case input, it suffice to substitute $a_v = F_a$ and $b_v = F_b$ to Theorem 13 and the theorem follows.

Lemma 15. $E[\hat{J}_{\text{sum}}] = E[\text{SUM}_W]$.

Proof. Similar to $\hat{J}_{\text{count}}$, each pair of tuples $(t_1, t_2)$ in the join appears in the join of the sample with probability $p_{\min(q_1, q_2)}$. We have:

$$E[\hat{J}_{\text{sum}}] = \frac{1}{p_{\min(q_1, q_2)}} E[\text{SUM}_W]$$
Lemma 16. The variance of \( \hat{J}_{\text{sum}} \) is given by:

\[
\begin{align*}
\text{Var}[\hat{J}_{\text{sum}}] &= \frac{1 - q_v}{pq_v} \beta_1 + \frac{1 - q_1}{pq_1} \beta_2 \\
&\quad + \frac{(1 - q_v)(1 - q_1)}{pq_v q_2} \beta_3 + \frac{1 - p}{p} \beta_4
\end{align*}
\]

where \( \beta_1 = \sum_v a_v^2 b_v \), \( \beta_2 = a_v(\mu_v^2 + \sigma_v^2) b_v \), \( \beta_3 = a_v(\mu_v^2 + \sigma_v^2)^2 b_v \), and \( \beta_4 = a_v^2 \mu_v^2 b_v \).

Proof. We analyze the variance of \( \hat{J}_{\text{sum}} \) using a similar process as COUNT *( )*. For each join value \( v \), define \( X_v \) to be the sum of the \( c \) values in \( S_1 \), and \( Y_v \) to be the number of tuples in \( S_2 \) whose value for column \( J \) is \( v \). Define \( W_v = X_v Y_v \) and

\[
Z_v = \begin{cases} W_v & \text{with probability } p \\ 0 & \text{otherwise} \end{cases}
\]

We have:

\[
E[X_v] = q_v a_v \mu_v
\]

and

\[
E[X_v^2] = (q_v(1 - q_v)) a_v(\mu_v^2 + \sigma_v^2) + q_v^2 a_v^2 \mu_v^2
\]

Recall that \( E[Y_v^2] = b_v q_2 (b_v q_2 + 1 - q_2) \).

Hence

\[
\begin{align*}
&= (q_v(1 - q_v)) a_v(\mu_v^2 + \sigma_v^2) + q_v^2 a_v^2 \mu_v^2 (b_v q_2 (b_v q_2 + 1 - q_2)) - q_v^2 a_v^2 \mu_v^2 q_2^2 b_v
\end{align*}
\]

And

\[
\begin{align*}
\text{Var}(Z_v) &= p \text{Var}(W_v) + p(1-p) E^2[W_v] \\
&= p((q_v(1 - q_v)) a_v(\mu_v^2 + \sigma_v^2) - q_v^2 a_v^2 \mu_v^2)(b_v q_2 (b_v q_2 + 1 - q_2)) \\
&\quad - q_v^2 a_v^2 \mu_v^2 q_2^2 b_v + p(1-p) q_v^2 a_v^2 \mu_v^2 q_2^2 b_v
\end{align*}
\]

Hence

\[
\begin{align*}
\text{Var}(\hat{J}_{\text{sum}}) &= \frac{1}{p^2 q_v^2 q_2^2} \sum_v \text{Var}(Z_v) \\
&= \frac{1 - q_v}{pq_v} a_v(\mu_v^2 + \sigma_v^2) b_v + \frac{1 - q_1}{pq_1} a_v(\mu_v^2 + \sigma_v^2) b_v \\
&\quad + \frac{(1 - q_v)(1 - q_1)}{pq_v q_2} a_v(\mu_v^2 + \sigma_v^2) b_v + \frac{1 - p}{p} a_v^2 \mu_v^2 b_v
\end{align*}
\]

Lemma 17. Given tables \( T_1, T_2 \) joined on column(s) \( J \), fixed sampling parameters \( (p_1, q_1) \) for \( T_1 \), and a fixed effective sampling rate \( \epsilon_2 \leq p_1 \) for \( T_2 \), the variance of \( \hat{J}_{\text{sum}} \) is minimized when \( T_2 \) also uses \( p_1 \) as its universe sampling rate and correspondingly, \( \epsilon_2/p_1 \) as its uniform sampling rate.

Proof. Similar to Lemma [10] we show that for any fixed \( p_1 \) between 1 and \( \max \epsilon_1 \), and \( q_1 = \epsilon_1/p_1 \), the variance of the optimal sampling parameters is minimized when the universe sampling rate of \( T_2 \) is the same as \( p_1 \):

**Case 1:** If \( p_2 \geq p_1 \): This case has simple intuition. When \( p_2 \geq p_1 \), the join universe sampling rate for both table \( p = \min \{p_1, p_2\} = p_1 \). Hence increasing \( p_2 \) beyond \( p_1 \) do not increase the join universe sampling rate, and only decreases the Bernoulli sampling rate for table \( T_2 \) and increases the variance of the overall estimator. In particular, we have \( p = p_1 \) and \( pq_1 = \epsilon_1 \), we have:

\[
\begin{align*}
\text{Var}(\hat{J}_{\text{sum}}) &= \sum_v \left( \frac{1 - q_v}{pq_v} a_v^2 b_v + \frac{1 - q_1}{pq_1} a_v(\mu_v^2 + \sigma_v^2)^2 b_v \\
&\quad + \frac{(1 - q_v)(1 - q_1)}{pq_v q_2} a_v(\mu_v^2 + \sigma_v^2) b_v + \frac{1 - p}{p} a_v^2 \mu_v^2 b_v \right)
\end{align*}
\]

which is a non-decreasing function in terms of \( p_2 \) and it is minimized when \( p_2 = p_1 \).

**Case 2:** If \( p_2 \leq p_1 \): Here, a smaller \( p_2 \) can result in smaller join universe sampling rate in exchange of large Bernoulli sampling rate for \( T_2 \). We want to show overall decreasing \( p_2 \) will result in a large variance of \( \hat{J}_{\text{sum}} \). In this case, \( p = p_2 \) and \( pq_2 = \epsilon_2 \), we have:

\[
\begin{align*}
\text{Var}(\hat{J}_{\text{sum}}) &= \sum_v \left( \frac{1 - q_v}{pq_v} a_v^2 b_v + \frac{1 - q_1}{pq_1} a_v(\mu_v^2 + \sigma_v^2)^2 b_v \\
&\quad + \frac{(1 - q_v)(1 - q_1)}{pq_v q_2} a_v(\mu_v^2 + \sigma_v^2) b_v + \frac{1 - p}{p} a_v^2 \mu_v^2 b_v \right)
\end{align*}
\]

Since \( b_v \)'s are nonnegative integers so \( b_v^2 \geq b_v \), \( \text{Var}(\hat{J}_{\text{sum}}) \) is a non-increasing function in terms of \( p_2 \) and \( pq_2 \).
Lemma 20. For any $p \geq \epsilon_1, \epsilon_2$, we have $h(\epsilon_2 \epsilon_1 \epsilon_2) \leq h(\epsilon_2 \epsilon_1 \epsilon_2)$. 

Proof. We focus on the first inequality $1/2h^*(p) \leq h^*(p)$. The second inequality holds since it is $h^*(p)$ obtained by maximizing over a larger subset. 

Observe that we can group the terms in $\mathbf{h}(\mathbf{f}, \mathbf{f})$ into $f_1(\mathbf{f})$ and $f_2(\mathbf{f})$, where 

$$ f_1(\mathbf{f}) = \frac{1}{1 - \epsilon_1} \mathbf{a}^T \mathbf{a} + \mathbf{b}^T \mathbf{b} $$

and 

$$ f_2(\mathbf{f}) = \frac{1}{1 - \epsilon_1} \mathbf{a}^T \mathbf{a} $$

That is, $f_1$ consists of combinations of $a_1^2 a_2^2 b_3 b_4$ and $f_2$ consists of combinations of $a_3 a_4 b_5 b_6$. Define $\mathbf{h}$ to be the vector that has $n_{10}$ on its $v$-th coordinate and 0 everywhere else. We can rewrite $h$ as:

$$ h(v) = f_1(\mathbf{h}, v) + f_2(\mathbf{h}, v) $$

By the choice of $v_1$ and $v_2$, we have for every $v$ that $f_1(\mathbf{h}, v) \leq f_1(\mathbf{b}_1, v)$ and $f_2(\mathbf{h}, v) \leq f_2(\mathbf{b}_2, v)$. Therefore, we have for all $v$ that:

$$ f_1(\mathbf{h}, v) + f_2(\mathbf{h}, v) \leq 2 \max \{ f_1(\mathbf{b}_1, v), f_2(\mathbf{b}_2, v) \} $$

Hence we have:

$$ h^*(p) = \max_v f_1(\mathbf{h}, v) + f_2(\mathbf{h}, v) $$

$$ h^*(p) \leq 2 \max \{ f_1(\mathbf{b}_1, v), f_2(\mathbf{b}_2, v) \} \leq 2h^*(p) $$

Theorem 22. Let $S$ and $C$ be random variables denoting the sum and cardinality of the join of two samples produced by applying UBS sampling parameters $(p_1, q_1)$ to $T_1$ and $(p_2, q_2)$ to $T_2$. Let $p_{\min} = \min \{ p_1, p_2 \}$. We have:

$$ \Var[S] = p_{\min} q_1 q_2 \sum_v \mu_v a_v b_v $$

$$ \Var[S] = \Var[S] \left( \frac{E[S]}{E[C]} \right)^2 + \Var[S] - 2 \Cov[S, C] + \Var[C] $$

where

$$ E[C] = p_{\min} q_1 q_2 \sum_v \mu_v a_v b_v $$

Proof. Following Theorem 19, we can consider a Taylor expansion around $X, Y$ as an approximation of the true value, and then use the fact that $E[f(x, y)]$ is a remainder of smaller order terms. For any function $f(x, y)$, the binomial expansion around any $(x_0, y_0)$ is:

$$ f(x, y) = f(x_0, y_0) + \frac{\partial f}{\partial x}(x_0, y_0)(x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0)(y - y_0) + R $$

where $R$ is a remainder of smaller order terms. Consider $X, Y$ as two random variables with mean $\mu_x$ and $\mu_y$, we can approximate $E[f(x, y)]$ by expanding $E[f(x, y)]$ around $(\mu_x, \mu_y)$:

$$ E[f(x, y)] \approx E[f(\mu_x, \mu_y)] + \frac{\partial f}{\partial x}(\mu_x, \mu_y)(X - \mu_y) $$

Now consider a second order Taylor expansion around $(x_0, y_0)$:

$$ f(x, y) = f(x_0, y_0) + \frac{\partial f}{\partial x}(x_0, y_0)(x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0)(y - y_0) + \frac{1}{2} \frac{\partial^2 f}{\partial x^2}(x_0, y_0)(X - x_0)^2 + \frac{1}{2} \frac{\partial^2 f}{\partial y^2}(x_0, y_0)(Y - y_0)^2 $$

We can simplify $E[f(x, y)]$ around $(\mu_x, \mu_y)$ and obtain:

$$ E[f(x, y)] = E[f(\mu_x, \mu_y)] + \frac{1}{2} \frac{\partial^2 f}{\partial x^2}(\mu_x, \mu_y)(X - \mu_x)^2 $$

Plugging in $f(S, C) = S/C$, we have:

$$ \Var[S/C] \approx \left( \frac{E[S]}{E[C]} \right)^2 \Var[S] - 2 \Cov[S, C] + \Var[C] $$

Notice that the expression of $E[S], E[C], \Var[S]$ and $\Var[C]$ has already been given in Theorem 11 and Theorem 18. The term $\Cov[S, C] = E[(X - \mu_x)(Y - \mu_y)]$ can be obtained similar to Theorem 11.
Theorem 23. In the centralized setting, set \( p^- = \max\{\epsilon_1, \epsilon_2\} \), 
\( p^+ = 1 \) and \( p^* = \min\{1, \max\{\epsilon_1, \epsilon_2, \sqrt{\frac{A - 2B + C}{D}}\}\} \). Then the optimal sampling parameter is given by:

\[
p = \begin{cases} 
p^- & \text{if } A - 2B + C \leq 0 \text{ and } D > 0 \\
p^+ & \text{if } A - 2B + C > 0 \text{ and } D \leq 0 \\
\arg\min_{p \in (p^-, p^+)} \frac{1}{p}(A - 2B + C) + pD & \text{otherwise}
\end{cases}
\]

Proof. The proof is identical to Theorem 12 and Theorem 19.

B. OMITTED ALGORITHMS

B.1 Omitted algorithm in Section 5.2.2

The algorithm determines the universe sampling rate \( p \) for a decentralized setting for SUM query is as follows:

1. \( v_1 = \arg\max_v a_v \mu^2 \) and \( v_2 = \arg\max_v a_v (\mu^2 + \sigma^2) \)
2. If \( v_1 = v_2 \), return

\[
p = \min\{1, \max\{\epsilon_1, \epsilon_2, 
(\epsilon_1 \epsilon_2 a_v^2 \mu^2, n_b^2 \\
- a_v^2 \mu^2 n_b - a_v(\mu^2 + \sigma^2) n_b^2 \\
+ a_v(\mu^2 + \sigma^2) n_b)/(a_v(\mu^2 + \sigma^2) n_b) \}^{1/2}\}.
\]

3. Otherwise, for \( i = 1, 2 \), let

\[
h_i(p) = \left(\frac{1}{\epsilon_2} - \frac{1}{p}\right)a_v \mu^2 n_b + \left(\frac{1}{\epsilon_1} - \frac{1}{p}\right)a_v(\mu^2 + \sigma^2) n_b \\
- \frac{p}{\epsilon_1} + \frac{1}{\epsilon_2} + \frac{1}{p}a_v(\mu^2 + \sigma^2) n_b \\
+ \left(\frac{1}{p} - 1\right)a_v^2 \mu^2 n_b^2,
\]

4. Find the roots \( p^* \) of \( h_1(p) = h_2(p) \), this can be reduced to solving a quadratic equation since \( h_i(p) \) are in the form \( A_i/p + B_i/p + C_i \). Let \( p_1 \) and \( p_2 \) be the roots.
5. Let \( p_3 \) and \( p_4 \) be the minimizer of \( h_1 \) and \( h_2 \), given by:

\[
p_3 = \min\{1, \max\{\epsilon_1, \epsilon_2, 
(\epsilon_1 \epsilon_2 a_v^2 \mu^2, n_b^2 \\
- a_v^2 \mu^2 n_b - a_v(\mu^2 + \sigma^2) n_b^2 \\
+ a_v(\mu^2 + \sigma^2) n_b)/(a_v(\mu^2 + \sigma^2) n_b) \}^{1/2}\}.
\]

And similarly for \( p_4 \) where we replace \( v_1 \) by \( v_2 \).
6. Let \( p_5 = \max\{\epsilon_1, \epsilon_2\} \).
7. Compute \( j = \arg\min_{i=1, \epsilon_2 \leq p_i \leq 1} \{\max\{h_1(p_i), h_2(p_i)\}\} \).
8. Return \( p = p_j \).

C. OTHER DECENTRALIZED PROTOCOLS

In Section 5 we analyzed decentralized settings using a Dictatorship protocol. In Dictatorship, one of the parties determines the universe sampling rate for everyone based on (i) its local statistics as well as (ii) the table size and the budget received from the other party.

An alternative protocol is a VOTER protocol, where each party proposes (i) a universe sampling rate and (ii) a worst case variance if this rate were to be adopted by everyone. Once this information is exchanged, all parties adopt the rate with the best worst case variance. While offering a better worst case variance by design, VOTER does not guarantee that the actual variance will indeed be lower than that of Dictatorship. This is because the actual variance depends on the local statistics of the other parties as well.

Moreover, for SUM and AVG queries, which unlike COUNT queries involve an aggregate column, one can show that VOTER is unnecessary: it is always better to simply adopt the rate proposed by the party who has the aggregate column. This is because the party without this information can only assume an arbitrary distribution of the aggregate column, leading to overestimation of the worst case variance.

Note that, if one modifies VOTER such that each party also shares their full statistics with others, the protocol then reduces to a centralized setting but with significantly more communication.

There is yet a more complex protocol that one can apply in a decentralized setting: an iterative, multi-round protocol, called EXPLORER. In this protocol, the parties each choose an arbitrary value as their initial sampling parameter, say \( p_1^0 \) and \( p_2^0 \), and produce a sample of their own table using their own parameter, say \( S_1^0 \) and \( S_2^0 \), respectively. Then, they each share their chosen sampling rate and sample with the other party. Then, each party uses its own local table and the sample received from the other party to derive a new sampling parameter, say \( p_1^1 \) and \( p_2^1 \), respectively. This process continues iteratively until the sampling parameters converge, or the amount of communication exceeds a fixed budget. EXPLORER, however, is significantly more expensive than both Dictatorship and VOTER. A full analysis of the EXPLORER protocol is beyond the scope of the current paper, and we leave to future work.