Distributed error estimation of functional dependency

Cheqing Jin\textsuperscript{a,*}, Ashwin Lall\textsuperscript{b}, Jun (Jim) Xu\textsuperscript{c}, Zhigang Zhang\textsuperscript{a}, Aoying Zhou\textsuperscript{a}

\textsuperscript{a}Institute for Data Science and Engineering, East China Normal University, Shanghai, China
\textsuperscript{b}Department of Math and Computer Science, Denison University, Granville, OH, USA
\textsuperscript{c}School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

Abstract

Measuring or estimating the number of errors in (i.e., violations to) a functional dependency (FD) offers valuable information about data semantics and quality. Most existing work focuses on FD error estimation in a centralized environment, where data are stored only in one site and the goal is to optimize the time and space complexities of the estimation algorithms. The distributed FD error estimation problem, in which the data can reside in multiple physically distributed sites, has never been studied in depth and is the subject of this work. In this work, we study a version of the distributed FD error estimation problem where a coordinator site communicates with multiple remote sites for arriving at such estimations, and the goal is to minimize this communication cost. We study two types of queries—that are dual to each other in semantics—for such estimations: one tries to maximize the accuracies of FD error estimations under fixed communication costs, and the other to minimize the communication costs needed to meet certain accuracy requirements. In our framework, each remote site maintains a concise synopsis data structure obtained by scanning its local data once, and the coordinator site receives and processes all such data structures to arrive at an estimate of the FD error. Our solution extends from the case of two remote sites to that of multiple remote sites. We demonstrate the efficacy of our proposed techniques via rigorous analysis and extensive experiments.

1. Introduction

As a core concept in relational database theory, functional dependency (FD) that describes a constraint between two sets of attributes has been studied extensively for decades. Let \( r(R) \) denote a relation \( r \) on a schema \( R \) that contains multiple attributes and let \( X \) and \( Y \) denote two non-empty attribute sets, \( X, Y \subseteq R \). We say an FD \( F: X \rightarrow Y \) holds over \( r(R) \), if \( \forall t_1, t_2 \in r(R), t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y] \). Functional dependencies can represent meaningful constraints for a relation.

Since functional dependencies may not always hold in some scenarios due to the existence of outliers, the functional dependency that allows few violations, which is called approximate functional dependency [17] or soft functional dependency [15], plays an important role in such applications. For example, the approximate functional dependency from a set of structural attributes to carcinogenicity may provide valuable hints to biochemists about the potential cause of cancer, indicating when further analysis by domain specialists is necessary [13]. Other applications include predicting missing values of attributes in relational tables [33], query optimization [15], query rewriting [23,33] and so on.

\* Corresponding author. Tel.: +86 21 62235578.
E-mail addresses: cqjin@sei.ecnu.edu.cn (C. Jin), lalla@denison.edu (A. Lall), jx@cc.gatech.edu (J. (Jim) Xu), zzg22936@sina.com (Z. Zhang), ayzhou@sei.ecnu.edu.cn (A. Zhou).

http://dx.doi.org/10.1016/j.ins.2016.01.051
0020-0255/© 2016 Elsevier Inc. All rights reserved.
One direction of study, which this work follows, is to estimate the errors of dependencies of a relation [7,11,15,17] due to two needs. First, such accurate error estimation helps us avoid missing important rules that have few FD violations, as mentioned above. Second, it helps us monitor the “quality” of data in a database. Consider an application where data continue to arrive from different sources, and as a consequence the number of FD violations also changes over time: A health-monitoring system can track the quality of the database using the proposed techniques, and take action to improve the data quality once the estimated error exceeds a pre-defined alert threshold.

There are multiple metrics for measuring the errors of a functional dependency in a relation. Kivinen and Mannila define three such metrics, namely the number of violating pairs (named $G_1$), the number of violating tuples (named $G_2$), and the number of tuples one has to delete to obtain a relation that satisfies the dependency (named $G_3$), along with sampling-based methods to detect approximate functional dependencies [17]. A series of work uses $G_3$ to measure the errors of a functional dependency in a relation [7,15]. As all such metrics are natural, and none of them is indisputably the best, we focus on $G_1$ in this paper. The differences between such measures will be discussed in Section 3.

1.1. Distributed environment

Though most of the existing work on FD, approximate FD, and CFD (conditional functional dependency) focuses on centralized databases, in practice a relation is often fragmented and distributed across more than one sites. For example, in sensor network applications, data are generated by physical equipments deployed at different locations. Moreover, distributing a relation to more than one site—a capability supported by multiple commercial systems, such as MySQL [22], Oracle [24,25] and SQL Server [29]—can significantly improve the efficiency of query processing. As illustrated in Fig. 1, a typical distributed environment has one coordinator site and multiple remote sites. Each remote site maintains its local data and communicates with the coordinator site. A user can submit a query to the coordinator site to get the result. Lowering the communication cost across all sites for answering the query is critical in this model, as described earlier.

We estimate the errors (violations) of a functional dependency in a distributed environment in this paper, aiming to minimize the communication costs. For example, in a credit card record-keeping application, each customer may have several credit cards issued by different banks, and some banks may allow a customer to apply for more than one card. Each bank may have a relation C(Card, SSN, Name) to store customers’ information, where Card is the credit card number, and SSN and Name are the customer’s social security number and name, respectively. Since each customer has a unique SSN, the functional dependency $F$: SSN $\rightarrow$ Name holds in the customer database at each bank (i.e., each remote site). However, some customers’ information may be recorded slightly differently at different banks, due to typos or use of nicknames or abbreviations (e.g., “Mike Jordan” and “M. Jordan”, “Dickie Dole” and “Dick Dole”). Table 1 illustrates a small example where 8 customers applied for 18 credit cards, and some of their names were recorded slightly differently across different sides, namely the customers with SSN 111, 222 and 444. For example, one customer (SSN = 222) applied for 2 and 3 cards in bank 1 and 2, respectively. His name was recorded as “Dickie Dole” at Bank 1, but as “Dick Dole” at Bank 2. The name of another customer (SSN=111) was recorded as “Mike Jordan”, “M. Jordan” and “Michael Jordan” at different banks.

Example 1. In Table 1, there exists no violating pair at each individual site. Between $C_1$ and $C_2$, there exist $8 (= 2 \times 1 + 2 \times 3)$ violating pairs. For example, card 101 in $C_1$ and card 203 in $C_2$ form a violating pair. Between $C_1$ and $C_3$, there exist...
Table 1
Relation C(Card, SSN, Name), stored in three remote sites.

<table>
<thead>
<tr>
<th>(a) $C_1$ from Bank 1</th>
<th>(b) $C_2$ from Bank 2</th>
<th>(c) $C_3$ from Bank 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card</td>
<td>SSN</td>
<td>Name</td>
</tr>
<tr>
<td>101</td>
<td>111</td>
<td>Mike Jordan</td>
</tr>
<tr>
<td>102</td>
<td>222</td>
<td>Dickie Dole</td>
</tr>
<tr>
<td>103</td>
<td>333</td>
<td>Harry Port</td>
</tr>
<tr>
<td>104</td>
<td>444</td>
<td>James Bond</td>
</tr>
<tr>
<td>105</td>
<td>111</td>
<td>Mike Jordan</td>
</tr>
<tr>
<td>106</td>
<td>222</td>
<td>Dickie Dole</td>
</tr>
</tbody>
</table>

4 ($= 2 \times 1 + 1 \times 2$) violating pairs, and between $C_2$ and $C_3$, there exists only 1 violating pair. Hence, there are 13 violating pairs in the global relation $C$.

1.2. The semantics of queries

Note that it is straightforward and computationally inexpensive to strictly enforce FD at each site, using simple techniques such as building a hash or tree-based index on $X$. Empowered by such a technique, a system can deny the insertion of a new tuple $t(X, Y)$ if it would violate the consistency of a given FD. Hence, we assume that violations of FD occur only across different sites.

The most straightforward—and also the naivest—way for detecting data inconsistency in a distributed environment is to transmit all data sets at all remote sites to the coordinator site. However, despite its simplicity, this approach is impractical in modern applications due to prohibitively high communication cost when the data sets at all remote sites are massive. The goal of this work is to design a solution that is much more efficient in the communication cost.

The error level, in terms of the number of violating pairs (NVP), measures the quality of a functional dependency in a relation. In a typical scenario, a user may set up an alert threshold, and expect to know whether or not the error exceeds this threshold. Hence, our first research question is how to decide whether the error level is above or below the alert threshold with high confidence, given a fixed amount of communication cost among all sites. Since the error level cannot be estimated precisely, typically this decision can be made if the error is close to the alert threshold.

One question that arises here is how to set a suitable communication cost bound in advance for estimating this error level. A standard approach is to define some estimation accuracy requirements, based on which the minimal communication cost that meets these requirements is computed. Taking this approach, we define a threshold and a confidence parameters and require that the event that the error is above the given threshold is reported with probability no smaller than the confidence parameter.

1.3. Contributions

We study the problem of error estimation of functional dependency in a distributed environment. Different from traditional centralized techniques, the major challenge is how to reduce communication cost, not the space- or time-complexities. Most of the existing work cannot be applied to our scenario, since (i) such centralized solutions are not aimed at reducing or minimizing communication costs, and (ii) they do not use NVP to measure errors. The major contributions are summarized below:

1. We study two types of queries. The first one decides whether the violations exceed a pre-defined threshold or not provided that the communication cost is fixed. The second one has two additional quality-aware requirements to control the result’s quality. They complement each other in semantics. 
2. We propose novel solutions to handle these queries. We begin with a simple case that only contains two remote sites, and the sub-relation in each site has a uniqueness property. Subsequently, we release the unique constraint. Our final solution is capable of dealing with the general case with more than two remote sites.
3. We study the lower bound of this problem and show that it can be challenging in both the approximation-only and randomization-only regimes, requiring us to use a randomized approximation solution.
4. We conduct a series of experiments upon synthetic and real data sets to evaluate the performance of the proposed methods.

The rest of the paper is organized below. Section 2 reviews the related work. Section 3 defines the problem formally. Sections 4–6 describe our solutions to deal with these queries. Section 7 analyzes the lower bounds of queries. Section 8 reports a series of experimental results. Finally, we conclude the paper briefly in the last section.

2. Related work

Functional dependency that describes a dependency constraint between two sets of attributes has been studied for several decades [21]. The work on functional dependency and its variants has many directions, inclusive of discovering and
maintaining FDs [5,28], repairing inconsistencies [6,18], reasoning FDs [4], error estimation [7,15,17], etc. In [28], Savnik and Flach discover FD rules in bottom-up manner. In [5], Chiang and Miller discover CFDs and dirty values in a data instance. To repair functional dependency violations, Kolahi and Lakshmanan propose an approximation algorithm to produce a repair close to the optimum repair [18]. Cong et al. propose solutions to improve data quality, where consistency is expressed by CFD. Bravo et al. define eCFD, an extension of CFDs, to specify patterns of semantically related values in terms of disjunction and inequality [4].

Because of the existence of some outliers, it is necessary to allow approximations to make the functional dependencies robust against outliers [27]. The first issue is how to measure the errors of a relation. Kivinen and Mannila propose three measures, G1, G2 and G3, to measure the error of a functional dependency [17]. The authors also propose some solutions to detect all dependencies with error at least \( \varepsilon \) in relation \( r \) with high probability based on a sample set from \( r \) [17]. In [15], Ilyas et al. propose a sampling-based method to detect AFD. However, their method does not have an error bound. In [7], Cormode et al. try to estimate the confidence of CFD with G3 as the error measurement. Their sampling-based method uses space complexity of \( O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}) \). Our work is close to [7,15,17] but with the following difference. First, our result has a bound in theorem, while [15] has no theorem bound. Second, We use G1 to measure the errors, while [7,15] uses G3. Detailed comparison with [17] is provided in Section 6.4 (([17] also studies G1). Other common difference includes (i) we consider a distributed environment, while the rest consider a centralized environment; (ii) our solutions are hashing-based, supporting delete operator by nature, while the rest are sampling-based.

The work on functional dependency in the distributed environment has also been studied recently. Fan et al. study how to detect data inconsistency (conditional functional dependency) in the distributed environment [9]. They show that the CFD detection problem with minimum data shipment is NP-complete either in the horizontal setting or in the vertical setting. Therefore, they propose a method to detect the violation of CFD in horizontally partitioned data. In their later work, they also study an incremental method for this issue [18]. In their data model, remote sites can communicate with each other while in our model, remote sites only communicate with the coordinator site. Furthermore, in their methods, data are shipped among remote sites, making all tuples with the same \( X \) value be evaluated at a single site, so that the total data communication cost is still large. But we only send to the coordinator a succinct summarization of each sub-relation at remote site, thus the communication cost is significantly saved.

Another kind of related work is on constraint enforcement in distributed databases [1,12,14]. Since constraint checking is hard in distributed settings, a common solution is to check constraints locally at individual sites to avoid data shipment. The global evaluation is triggered only when local constraint is detected violated. Different from theirs, our goal is to estimate error of an FD. Nowadays, many Map-Reduce based platforms help to manage and shuffle data by maintaining multiple copies [31,32]. Under this framework, the querying task is typically divided into several sub-tasks to be processed in different computing nodes so as to improve time-efficiency. However, these techniques are inefficient in communication cost since all data at remote sites must be transmitted to the platform in advance. Distributed hash table (DHT) provides a lookup service similar to a hash table in the distributed environment to [30]. However, it works for the Peer-to-Peer model, not the model in Fig. 1.

3. Problem definition

3.1. Data model and violations

Consider a relation \( r(R) \) with data distributed across \( \psi \) remote sites, denoted as \( r_1(R), \ldots, r_\psi(R) \). Here, \( R \) represents the schema of \( r \). We consider an arbitrary functional dependency \( F: X \rightarrow Y \) over \( r(R) \), where \( X, Y \subseteq R \). Assume \( F \) holds exactly at each remote site \( k \), i.e., for any \( i, j \), \( t_{ik}[X] = t_{jk}[X] \) implies \( t_{ik}[Y] = t_{jk}[Y] \), where \( t_{ik}[X] \) denotes the \( i \)th tuple in \( r_k(R) \). As mentioned earlier, this condition is reasonable in real applications, since it can be enforced using existing techniques such as tree-like or hashing-based indexes. However, \( F \) may still be violated on \( r(R) \) in a distributed environment.

Generally speaking, more than one record with the same \( t[X] \) value may co-exist at each site \( k \). In other words, there may exist \( t_i \neq t_j \) at one site such that \( t_{ik}[X] = t_{jk}[X] \). This in turn, we pay special attention to a simpler yet typical case that has uniqueness property over \( X \) at each site: for any two tuples \( t_i, t_j \) at any site \( k \), \( t_{ik}[X] \neq t_{jk}[X] \). This special case is called the unique case.

Let \( N_k \) denote the number of records in \( r_k(R) \). Let \( M_k \) denote the number of distinct values of \( t[X] \) in each site. Hence, in general, \( N_k \geq M_k \); but in the unique case, \( N_k = M_k \). We use the number of violating pairs (NVP) to measure the errors upon the global relation, which is also used in [17] (named as G1).

**Definition 1** (NVP). NVP refers to the number of violating pairs in \( r(R) \), computed as below:

\[
NVP(X \rightarrow Y, r) = |\{(u, v) | u, v \in r, u[X] = v[X], u[Y] \neq v[Y]\}|.
\] (1)
Table 2  
An example of error metrics.

<table>
<thead>
<tr>
<th></th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S2</td>
<td>99</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>S3</td>
<td>196</td>
<td>100</td>
<td>2</td>
</tr>
</tbody>
</table>

3.2. Error metrics of functional dependency

There exist several metrics to measure the error of a functional dependency, including G1, G2 and G3 that are proposed in [17]. However, there is no consensus on which one is the best since how to describe the “quality” of a database is subjective, depending on the specific application scenario. Here, we attempt to clarify the differences between them. All three metrics are functions of the inconsistent tuples of the database. For any such inconsistent tuple t, we can find another tuple t’ in the database, such that t[X] = t’[X] ∧ t[Y] ≠ t’[Y]. As illustrated in the following example, the differences between these three metrics lie only in how the number of inconsistent tuples is quantified: Metrics G1 and G2 consider all inconsistent tuples, while G3 considers only a subset of them.

Example 2. Consider three relations S1(X, Y), S2(X, Y), and S3(X, Y), each containing 100 tuples. S1 contains one (a, b), one (a, c) and 98 (b, d). S2 contains one (c, d) and 99 (c, e). S3 contains two (a, c) and 98 (a, d). Let’s consider the functional dependency F: X → Y. Table 2 illustrates the error metrics for these data sets. Although the G3 value is similar for each data set, the values of G1 and G2 differ significantly because all inconsistent tuples are considered.

In general, multiple metrics co-exist in other fields. For example, edit distance and Jaccard distance are commonly used to measure the distance between a pair of strings. Edit distance, similar to some extent, any of G1–G3 is meaningful in concrete scenarios.

3.3. Query definitions

We consider two types of FD error queries that present the tradeoff between the accuracy of an answer to the query and the amount of resource (communication cost) involved to produce the answer, in different ways. The first type is to output answers as accurate as possible under a fixed communication cost bound.

Definition 2 (Cost oriented detection, CoD). Let D be the aforementioned (fixed) communication cost bound, τ ≥ 0 a threshold parameter, and δ a confidence parameter that satisfies 0 < δ < 1. A CoD query will output one of the following claims by incurring a communication cost of at most D.

1. Claim 1: NVP > τ with confidence at least 1 − δ.
2. Claim 2: NVP < τ with confidence at least 1 − δ.
3. Neither of the above two claims can be made.

Eq. (2) gives an alternate but equivalent definition of the CoD query:

\[
\begin{align*}
\Pr[NVP > \tau | \text{“output Claim 1”}] & \geq 1 - \delta \\
\Pr[NVP < \tau | \text{“output Claim 2”}] & \geq 1 - \delta.
\end{align*}
\]  

(2)

The value of D strongly influences the query result. When D is large (i.e., a higher communication cost is allowed), it is more likely that either claim 1 or claim 2 is the answer to the query; When D is very small, it is highly likely that neither claim will be the answer to the query, even if NVP is significantly different from τ.

In most applications, when NVP is significantly different from τ, we really would like to know whether NVP is much smaller than τ, or much larger, and are willing to pay the (ideally minimum) communication cost D that is required to achieve this goal. Toward this end, we define the other type of query, which incorporates two additional sets of parameters. The first set of parameters are τ1 and ρ1, that satisfy τ1 > τ and 0 < ρ1 < 1. If NVP > τ1, then our estimation algorithm must output “NVP > τ” (Claim 1) with confidence at least 1 − ρ1. The second set of parameters are τ2 and ρ2, that satisfy τ2 < τ and 0 < ρ2 < 1. If NVP < τ2, then our estimation algorithm must output “NVP < τ” (Claim 2) with confidence at least 1 − ρ2. The research task is to compute the minimal communication cost D that is sufficient for these requirements.

Definition 3 (Quality oriented detection, QoD). Let τ, τ1, τ2 denote three threshold parameters, and δ, ρ1, ρ2 denote three confidence parameters, 0 < δ, ρ1, ρ2 < 1, 0 < τ2 < τ < τ1. A QoD query satisfies the following three conditions.

1. One of three claims in Definition 2 is output.
2. If NVP > τ1, claim 1 in Definition 2 is output with confidence at least 1 − ρ1.
3. If NVP < τ2, claim 2 in Definition 2 is output with confidence at least 1 − ρ2.
The QoD query is summarized by Eq. (2), above, and Eq. (3), below:

\[
\begin{align*}
\Pr[&\text{"output Claim 1"} \mid NVP > \tau_1] \geq 1 - \rho_1 \\
\Pr[&\text{"output Claim 2"} \mid NVP < \tau_2] \geq 1 - \rho_2.
\end{align*}
\]  
(3)

4. Unique case at two sites

Recall that we have introduced in Section 3 a simple unique case where there exists a uniqueness property over \( X \) in each site. In this section, we begin with the unique case upon two remote sites. For the ease of reading, we name two remote sites as \( A \) and \( B \). Site \( A \) (\( B \)) contains \( N_A \) (\( N_B \)) tuples, denoted as \( t_A^i \) (\( t_B^i \)).

We use two families of hash functions: \( \mathcal{H} \) and \( \mathcal{G} \). Any hash function in \( \mathcal{H} \), denoted as \( h(\cdot) \), maps values in the domain of \( X \) to \([-1, 1] \) uniformly, while any hash function in \( \mathcal{G} \), denoted as \( g(\cdot) \), also maps values in the domain of \( (X \times Y) \) to \([-1, 1] \) uniformly. Let \( Z_A \) and \( W_A \) denote two random variables, computed as: \( Z_A = \sum_{i=1}^{N_A} h(t_A^i[X]) \) and \( W_A = \sum_{i=1}^{N_A} g(t_A^i[X], t_A^i[Y]) \). The other two random variables, \( Z_B \) and \( W_B \), are defined symmetrically, but for the data in site \( B \). Here, \( E[Z_A] = E[W_A] = 0 \) and \( E[Z_A^2] = E[W_A^2] = N_A \). These four random variables can be computed at their respective local sites.

Let \( S_A = \{t_A^i[X], i = 1, 2, \ldots, N_A\} \), \( S_B = \{t_B^i[X], i = 1, 2, \ldots, N_B\} \), and \( |S_A \triangle S_B| = |S_A \setminus S_B| + |S_B \setminus S_A| \). Since \( S_A \) is the union of two nonoverlapping sets \( S_A \setminus S_B \) and \( S_A \cap S_B \), we have,

\[
Z_A = \sum_{z \in S_A \setminus S_B} h(z) + \sum_{z \in S_A \cap S_B} h(z).
\]  
(4)

\( Z_B \) is defined similarly. Then,

\[
Z_A - Z_B = \sum_{z \in S_A \setminus S_B} h(z) - \sum_{z \in S_A \cap S_B} h(z)
\]  
(5)

So,

\[
\]  
(6)

Eq. (6) holds since \( E[Z_A \cdot Z_B] = |S_A \cap S_B| \). Moreover, since all \( h(z)'s \) in \( Z_A - Z_B \) are independent Bernoulli random variables, \( Z_A - Z_B \) can be viewed as a centered (zero mean) binomial random variable.

Similarly, \( W_A - W_B \) is also a centered binomial random variable with variance computed in Eq. (7). The additional item \((2 \cdot NVP)\) is due to NVP pairs of tuples in two sites sharing the same \( X \) values, but different \( Y \) values:

\[
\text{Var}[W_A - W_B] = |S_A \triangle S_B| + 2 \cdot NVP.
\]  
(7)

Thus, if \( |S_A \triangle S_B| > 0 \), we have:

\[
\frac{\text{Var}[W_A - W_B]}{\text{Var}[Z_A - Z_B]} = 1 + \frac{2 \cdot NVP}{|S_A \triangle S_B|}.
\]  
(8)

**Example 3.** Consider two relations, \( R_A(X, Y) \) and \( R_B(X, Y) \) in two remote sites, each containing 100 tuples. The ith tuple of \( R_A \) is \((i, i)\), for \( 1 \leq i \leq 100 \). The ith tuple of \( R_B \) is \((i + 1, i + 1)\) for \( 1 \leq i \leq 90 \), or \((i + 1, i)\) for \( 90 < i \leq 100 \). In this case, there are nine pairs of violating tuples, i.e., \( \text{NVP} = 9 \). Then, \( S_A \triangle S_B = \{1, 101\} \). Let \( h \) and \( g \) denote two hash functions mentioned above. We have: \( Z_A - Z_B = h(1) - h(101) \), \( W_A - W_B = g(1, 1) + \sum_{i=2}^{100} g(i, i) - g(i, i - 1) - g(101, 100) \). So, \( \text{Var}[Z_A - Z_B] = 2 \), \( \text{Var}[W_A - W_B] = 20 \). Hence, Eq. (8) holds.

Since the violations only happen in \( S_A \cap S_B \), we know that \( \text{NVP} \leq |S_A \cap S_B| \). Under any given hash functions \( h(\cdot) \) and \( g(\cdot, \cdot) \), if the functional dependency holds upon the entire data, \( \text{NVP} = 0 \) and \( \text{Var}[Z_A - Z_B] = \text{Var}[W_A - W_B] \); otherwise, \( \text{NVP} > 0 \) and \( \text{Var}[W_A - W_B] > \text{Var}[Z_A - Z_B] \).

We use two steps to estimate NVP. First, we estimate \( |S_A \triangle S_B| \). Second, we test whether \( \text{Var}[W_A - W_B] \) is significantly greater than \( \text{Var}[Z_A - Z_B] \) or not.

4.1. Estimating \( |S_A \triangle S_B| \)

Recall that \( Z_A - Z_B \) can be viewed as a centered (zero mean) binomial random variable. By Eq. (6), \( |S_A \triangle S_B| \) is equal to the variance of \( Z_A - Z_B \). We can estimate \( |S_A \triangle S_B| \) by generating a number of samples of \( Z_A - Z_B \), after receiving two sample sets for \( Z_A \) (from site \( A \)) and \( Z_B \) (from site \( B \)) respectively (introduced later in lines 1–2, Algorithm 1).

**Lemma 1.** Given \( n \) samples of \( Z_A - Z_B \), the upper and lower bounds of \( |S_A \triangle S_B| \) can be estimated by Eqs. (13) and (14).

**Proof.** Recall that \( \text{Var}[Z_A - Z_B] = |S_A \triangle S_B| \). Let \( v_1, \ldots, v_n \) denote \( n \) samples from \( Z_A - Z_B \), and \( \bar{v} = \frac{\sum_{i=1}^{n} v_i}{n} \). Let \( v = \sum_{i=1}^{n} (v_i - \bar{v})^2 \). Then \( \frac{v}{\text{Var}[Z_A - Z_B]} \sim \chi^2(n - 1) \). Note that \( E[\chi^2(n-1)] = n - 1 \).
Algorithm 1: FtestCoD ($\tau$, $\delta$, $D$).

1. $n \leftarrow$ the size of each sample set, typically $\frac{D}{10}$;
2. Receive four sets of samples, $\{z_A^n\}$, $\{w_A^n\}$, $\{z_B^n\}$ and $\{w_B^n\}$, each with a length of $n$;
3. $S_X^2 \leftarrow \sum_{i=1}^n (z_A^i - \bar{z}_A)^2$; $S_Y^2 \leftarrow \sum_{i=1}^n (w_A^i - \bar{w}_A)^2$;
4. if $S_X^2 = 0$ then
5. \hspace{1em} if $\frac{S_Y^2}{2(n-1)\tau X(n, n-1)} > \tau$ then
6. \hspace{2em} return claim 1 holds;
7. \hspace{1em} else if $\frac{S_Y^2}{2(n-1)\tau X(n, n-1)} < \tau$ then
8. \hspace{2em} return claim 2 holds;
9. \hspace{1em} else
10. \hspace{2em} return Neither claim holds;
11. else
12. \hspace{1em} $f_{\text{val}} \leftarrow \frac{S_X^2}{S_Y^2}$;
13. \hspace{1em} $\gamma_1 \leftarrow \frac{2\tau (n-1) U(\frac{1}{2}, n-1)}{S_X^2}$, $\gamma_2 \leftarrow \frac{2\tau (n-1) L(\frac{1}{2}, n-1)}{S_Y^2}$;
14. \hspace{1em} if $f_{\text{val}} > (1 + \gamma_1) \cdot F_{\frac{1}{2}, n-1, n-1}$ then
15. \hspace{2em} return claim 1 holds;
16. \hspace{1em} else if $f_{\text{val}} < (1 + \gamma_2) \cdot F_{\frac{1}{2}, n-1, n-1}$ then
17. \hspace{2em} return claim 2 holds;
18. \hspace{1em} else
19. \hspace{2em} return Neither claim holds;

In [20], the upper and lower bounds of a random variable $V$ that follows a Chi-squared distribution of order $n - 1$, $\chi^2(n - 1)$, are estimated below (Eqs. (9) and (10)).

$$
\Pr\left[ V - (n - 1) \geq 2\sqrt{(n - 1) \cdot x + 2x} \right] \leq e^{-x}
$$
\hspace{1em} (9)

$$
\Pr\left[ (n - 1) - V \geq 2\sqrt{(n - 1) \cdot x} \right] \leq e^{-x}
$$
\hspace{1em} (10)

where $x$ is a positive value. Let $\delta = e^{-x}$. We define two functions below:

$$
U(\delta, n - 1) = 1 + 2\sqrt{\ln \delta - 1 \over n - 1} + 2\ln \delta - 1 \over n - 1
$$
\hspace{1em} (11)

$$
L(\delta, n - 1) = 1 - 2\sqrt{\ln \delta - 1 \over n - 1}
$$
\hspace{1em} (12)

We use $\frac{U}{n - 1}$ to estimate $|S_A \triangle S_B|$, as shown below:

$$
\Pr\left[ \frac{U}{n - 1} \leq |S_A \triangle S_B| \cdot L(\delta, n - 1) \right] \leq \delta
$$
\hspace{1em} (13)

$$
\Pr\left[ \frac{U}{n - 1} \geq |S_A \triangle S_B| \cdot U(\delta, n - 1) \right] \leq \delta
$$
\hspace{1em} (14)

4.2. Comparing $\text{Var}[Z_A - Z_B]$ and $\text{Var}[W_A - W_B]$

Both $Z_A - Z_B$ and $W_A - W_B$ are normally distributed when the data volume is huge. Note that $E[Z_A - Z_B] = E[W_A - W_B] = 0$. We use $F$-test, a common way to detect the difference of variance for two normal distributions: $V_1$ and $V_2$ [8]. Let $\{v_1^{(1)}\}$ and $\{v_1^{(2)}\}$ denote two sample sets generated from $V_1$ and $V_2$ respectively, each with a size of $N$. Let $\bar{v}_1$ and $\bar{v}_2$ denote the average of the two sample sets. The variances of these two sets are defined as: $S_{V_1}^2 = \frac{1}{n-1} \sum_{i=1}^N (v_1^{(1)} - \bar{v}_1)^2$, and $S_{V_2}^2 = \frac{1}{n-1} \sum_{i=1}^N (v_2^{(2)} - \bar{v}_2)^2$. We define $f_{\text{val}}$ as $f_{\text{val}} = \frac{S_{V_1}^2}{S_{V_2}^2}$.
Lemma 2 [8]. Let $H_0$ and $H_1$ denote two hypothetical events, $H_0$: $\text{Var}[V_1] \geq \text{Var}[V_2]$, and $H_1$: $\text{Var}[V_1] < \text{Var}[V_2]$. Let $\delta$ denote the probability parameter. The null hypothesis will be rejected if $f\text{val} > F_{1-\delta,N-1,N-1}$, where $F_{1-\delta,N-1,N-1}$ is computed from an $F$-distribution with degrees $N-1$ and $N-1$.

Lemma 3. Let $H_0$ and $H_1$ denote two hypothetical events, and $\gamma$ is a small positive parameter. $H_0 : (1 + \gamma) \cdot \text{Var}[V_1] \geq \text{Var}[V_2]$, and $H_1 : (1 + \gamma) \cdot \text{Var}[V_1] < \text{Var}[V_2]$. Let $\delta$ denote the probability parameter. If we have $f\text{val} > (1 + \gamma)F_{1-\delta,N-1,N-1}$, the null hypothesis will be rejected.

Proof. If $(1 + \gamma')\text{Var}[V_1] = \text{Var}[V_2]$ and $\gamma' \geq \gamma$, then $(\frac{S^2_2}{1 + \gamma'})/S^2_{V_1}$ follows $F$-distribution with degrees $N-1$ and $N-1$. Thus,

$$\text{Pr}\left[ f\text{val} > (1 + \gamma') \cdot F_{1-\delta,N-1,N-1} \right] \leq \text{Pr}\left[ \left( \frac{S^2_2}{1 + \gamma'} \right)/S^2_{V_1} \geq F_{1-\delta,N-1,N-1} \right] < \delta$$

(15)

\[ \square \]

Lemma 4. Let $H_0$ and $H_1$ denote two hypothetical events, and $\gamma$ is a small positive parameter. $H_0 : (1 + \gamma) \cdot \text{Var}[V_1] \leq \text{Var}[V_2]$, and $H_1 : (1 + \gamma) \cdot \text{Var}[V_1] > \text{Var}[V_2]$. Let $\delta$ denote the probability parameter. The null hypothesis will be rejected if $f\text{val} < (1 + \gamma)F_{\delta,N-1,N-1}$ holds.

Proof. If $(1 + \gamma')\text{Var}[V_1] = \text{Var}[V_2]$ and $\gamma' \leq \gamma$, then $(\frac{S^2_2}{1 + \gamma'})/S^2_{V_1}$ follows $F$-distribution with degrees $N-1$ and $N-1$. Thus,

$$\text{Pr}\left[ f\text{val} < (1 + \gamma') \cdot F_{\delta,N-1,N-1} \right] \leq \text{Pr}\left[ \left( \frac{S^2_2}{1 + \gamma'} \right)/S^2_{V_1} \leq F_{\delta,N-1,N-1} \right] < \delta$$

(16)

\[ \square \]

Let $\gamma = \frac{2NVP}{|S_A\Delta S_B|}$. By Lemmas 3 and 4, we can check whether $\text{Var}[W_A - W_B] > (1 + \gamma)\text{Var}[Z_A - Z_B]$, or $\text{Var}[W_A - W_B] < (1 + \gamma)\text{Var}[Z_A - Z_B]$ with confidence at least $1 - \delta$.

Specially, $\gamma = \infty$ when $|S_A\Delta S_B| = 0$, which makes $F$-test invalid. Fortunately, $NVP = \frac{\text{Var}[W_A-W_B]}{2}$ holds at that time. The query can be handled by Lemma 1.

4.3. Dealing with CoD query

Algorithm FtestCoD (Algorithm 1) illustrates how to deal with CoD query, using three parameters, including threshold $\tau$, confidence $\delta$, and communication cost $D$. Initially, it computes four sets, $\{z^A_n\}$, $\{w^A_n\}$, $\{z^B_n\}$ and $\{w^B_n\}$, from two sites, each following random variables $Z_A$, $W_A$, $Z_B$, and $W_B$. The size of each set is decided by $D$. By default, it is set to $\frac{D}{n}$, since there are four sets, in which each entry is an integer (4 bytes). Then, it computes $S^2_X$ and $S^2_Y$, as so to estimate $|S_A\Delta S_B|$ and compute $f\text{val}$ (lines 1–3). If $S^2_X = 0$, which means $|S_A\Delta S_B|$ is equal to 0, or at least very close to 0, then $NVP = \frac{\text{Var}[W_A-W_B]}{2}$ is estimated as $\frac{S^2_Y}{n(n-1)}$. By Lemma 1, $\text{Pr}\left[ \frac{S^2_Y}{n(n-1)} \leq |S_A\Delta S_B| \right] \leq \delta$, and $\text{Pr}\left[ \frac{S^2_Y}{n(n-1)} \geq |S_A\Delta S_B| \right] \leq \delta$, so we get the result (lines 4–10).

Otherwise, we next compute $f\text{val}$, a key value for $F$-test. Recall that $\frac{\text{Var}[W_A-W_B]}{\text{Var}^{\text{CoD}}_{W_A-W_B}} = (1 + \gamma)$ if $|S_A\Delta S_B| \neq 0$. According to Lemmas 3–4, to obtain Claim 1 ($NVP > \tau$), $f\text{val} > (1 + \gamma)F_{1-\delta,n-1,n-1}$ must hold; to obtain Claim 2 ($NVP < \tau$), $f\text{val} < (1 + \gamma)F_{\delta,n-1,n-1}$ must hold.

However, since it is expensive to compute the exact value of $\gamma$, an alternative way is to get an approximate value of $\gamma$ (denoted as $\gamma_1$ and $\gamma_2$), and the evaluating condition is also altered slightly. According to Lemma 1, $\frac{S^2_X}{n-1}$ acts as an estimator of $|S_A\Delta S_B|$, with $|S_A\Delta S_B| : L(\frac{1}{2}, n - 1)$ and $|S_A\Delta S_B| : U(\frac{1}{2}, n - 1)$ as the lower and upper bounds, each with confidence at least $1 - \frac{\delta}{2}$. Subsequently, we define $\gamma_1$ and $\gamma_2$ below, and have the following inequalities:

$$\text{Pr}\left[ \gamma > \gamma_1 \right] \leq \delta$$

(17)

$$\text{Pr}\left[ \gamma < \gamma_2 \right] \leq \delta$$

(18)

Then, $\text{Pr}\left[ f\text{val} > (1 + \gamma_1)F_{1-\frac{\delta}{2},n-1,n-1} \right] < \delta$ holds for Claim 1, and

$$\text{Pr}\left[ f\text{val} < (1 + \gamma_2)F_{\frac{\delta}{2},n-1,n-1} \right] < \delta$$

holds for Claim 2 (lines 11–19).
Algorithm 2: FtestQoD ($\tau$, $\delta$, $\tau_1$, $\tau_2$, $\rho_1$, $\rho_2$).

1. Receive two sample sets $\{z^A_1\}$, $\{z^B_1\}$ from two sites, each with a size of basic;
2. $S^2_X \leftarrow \sum_{i=1}^{\text{basic}} (z^A_i - z^B_i)^2$;
3. Compute $n$ by using Theorem 1 (if $S^2_X = 0$) or Theorem 2 (otherwise);
4. Invoke Algorithm 1 ($\tau$, $\delta$, 16$n$);

4.4. Dealing with QoD query

A QoD query has two more requirements, in comparison with CoD. By our solution, FtestQoD (Algorithm 2), the coordinator receives two small sample sets of $\{z^A_1\}$ and $\{z^B_1\}$ from two sites initially, each with a size of basic, based on which a new size $n$ is computed according to Theorem 1 (if $S^2_X = 0$) or Theorem 2 (otherwise). Finally, it invokes Algorithm 1 for further processing (line 4). Since basic samples of $\{z^A_1\}$ and $\{z^B_1\}$ have been set to the coordinator, it is only necessary to transmit $(n - \text{basic})$ more samples of $\{z^A_1\}$ and $\{z^B_1\}$ if $n > \text{basic}$.

**Theorem 1.** If $S^2_X = 0$, the minimal size of each sample set $n$ for both requirements is $\max(n_1, n_2)$, where $n_1$ and $n_2$ are the minimal valid $n$'s for Eqs. (19) and (20) respectively.

**Proof.** We first consider event: $\text{NVP} > \tau_1$. Given $n$ samples, $\Pr\left[\frac{S^2_X}{2(n-1)^2} < L(\rho_1, n-1) \cdot \tau_1 \right] < \rho_1$. Algorithm 1 outputs claim 1 if

$$\frac{n - \text{basic}}{2} \geq \frac{\tau_1 \cdot L(\frac{2}{3}, n-1) \cdot (1-F_{\frac{3}{2}, n-1})}{1-F_{\frac{3}{2}, n-1}}$$

Similarly, considering the second requirement, we have:

$$\frac{n - \text{basic}}{2} \geq \frac{\tau_2 \cdot U(\frac{2}{3}, n-1) \cdot (1-F_{\frac{3}{2}, n-1})}{1-F_{\frac{3}{2}, n-1}}$$

Let $n_1$ and $n_2$ denote the minimal $n$ to make Eqs. (19) and (20) hold respectively. Then $\max(n_1, n_2)$ is the minimal value for both requirements. □

**Theorem 2.** If $S^2_X > 0$, the minimal size of each sample set $n$ for a QoD query is the maximal one of the minimal valid $n$’s in Eqs. (21) and (22).

$$\frac{ub_1}{2} \leq \frac{\tau_1 \cdot L(\frac{2}{3}, n-1) \cdot (1-F_{\frac{3}{2}, n-1})}{1-F_{\frac{3}{2}, n-1}}$$

$$\frac{ub_2}{2} \leq \frac{\tau_2 \cdot U(\frac{2}{3}, n-1) \cdot (1-F_{\frac{3}{2}, n-1})}{1-F_{\frac{3}{2}, n-1}}$$

where

$$\begin{cases}
    ub_1 \leftarrow \frac{S^2_X}{2} \\
    ub_2 \leftarrow \frac{S^2_X}{2}
\end{cases}
\begin{cases}
    FR_1 = \frac{F_{\rho_1/3,n-1,n-1}}{1-F_{\frac{3}{2},n-1}} \\
    FR_2 = \frac{F_{\rho_2/3,n-1,n-1}}{1-F_{\frac{3}{2},n-1}}
\end{cases}
$$

**Proof.** We first consider event: $\text{NVP} > \tau_1$. Assume we get $n$ samples so that $\Pr[\text{fval} < (1 + \gamma_1')F_{\frac{3}{2},n-1,n-1}] < \frac{2}{3} \rho_1$, where

$$\gamma_1' = \frac{2\tau_1 (n-1) L(\frac{2}{3}, n-1)}{S^2_X},$$

and $\text{fval} = \frac{S^2_X}{2}$. Since Algorithm 1 that is invoked by Algorithm 2 at last outputs claim 1 when $\text{fval} > (1 + \gamma_1')F_{\frac{3}{2},n-1,n-1}$, this event can be detected with confidence at least $1 - \frac{2}{3} \rho_1$ if the following inequality holds:

$$\frac{(1 + \gamma_1') \cdot F_{\frac{3}{2},n-1,n-1} - (1 + \gamma_1') \cdot F_{\frac{3}{2},n-1,n-1}}{1 - F_{\frac{3}{2},n-1,n-1}} \leq \frac{S^2_X}{2(n-1)}.$$

Subsequently, we have:

$$\frac{S^2_X}{2(n-1)} \leq \frac{\tau_1 \cdot L(\frac{2}{3}, n-1) \cdot (1-F_{\frac{3}{2}, n-1})}{1-F_{\frac{3}{2}, n-1}}.$$

Note that $\frac{S^2_X}{n-1}$ estimates $|S_A \Delta S_B|$. Based on basic samples, we compute $ub_1$ as the upper bound of $|S_A \Delta S_B|$ with confidence at least $1 - \frac{2}{3} \rho_1$. Hence, if Eq. (21) holds, that event can be detected with confidence at least $1 - \rho_1$. Due to the similarity, we omit the analysis for event: $\text{NVP} < \tau_2$. But the result is listed as Eq. (22). □
5. Relaxing the uniqueness constraint

We move to a more general case without the uniqueness property.

5.1. Analysis

Without loss of generality, to facilitate the analysis, we define the following notation. The data in site \( A \) are represented as \( \{(x_i^A, y_i^A, c_i^A)\} \), where \( x_i \) and \( y_i \) represent the distinct values in \( X \) and \( Y \), and \( c_i^A \) represents the frequency of this pair. Similarly, the data in site \( B \) are represented as \( \{(x_i^B, y_i^B, c_i^B)\} \). Assume the first \( m' \) items have the same \( X \) values, but different \( Y \) values, i.e., \( \forall i \in [1, m'] \), \( x_i^A = x_i^B \land y_i^A \neq y_i^B \). Note that \( m' \) happens to be the size of violation set. The next \( m - m' \) items have the same \( X \) and \( Y \) values, i.e., \( \forall i \in [m' + 1, m] \), \( x_i^A = x_i^B \land y_i^A = y_i^B \). The remaining \( M_A - m \) (or \( M_B - m \)) items in site \( A \) (or \( B \)) do not appear in the other site.

Let \( Z_A, Z_B, W_A \) and \( W_B \) be defined as the same as that in Section 4, i.e., \( Z_A = \sum_{i=1}^{N_A} h(t_i^A[X]) \), \( W_A = \sum_{i=1}^{N_A} g(t_i^A[X], t_i^A[Y]) \). They can also be expressed below:

\[
Z_A = \sum_{i=1}^{M_A} c_i^A \cdot h(x_i^A) \quad W_A = \sum_{i=1}^{M_A} c_i^A \cdot g(x_i^A, y_i^A)
\]

\[
Z_B = \sum_{i=1}^{M_B} c_i^B \cdot h(x_i^B) \quad W_B = \sum_{i=1}^{M_B} c_i^B \cdot g(x_i^B, y_i^B)
\]

Then,

\[
Z_A - Z_B = \sum_{i=1}^{m} (c_i^A - c_i^B)h(x_i^A) + \sum_{i=m+1}^{M_A} c_i^A \cdot h(x_i^A) - \sum_{i=m+1}^{M_B} c_i^B \cdot h(x_i^B)
\]

So, \( \mathbb{E}[Z_A - Z_B] = 0 \), and the variance is computed below:

\[
\text{Var}[Z_A - Z_B] = \sum_{i=1}^{m} (c_i^A - c_i^B)^2 + \sum_{i=m+1}^{M_A} c_i^A \cdot h(x_i^A)^2 - \sum_{i=m+1}^{M_B} c_i^B \cdot h(x_i^B)^2
\]

Symmetrically, \( \mathbb{E}[W_A - W_B] = 0 \), and \( \text{Var}[W_A - W_B] \) is computed below:

\[
\text{Var}[W_A - W_B] = \text{Var}[Z_A - Z_B] + 2 \cdot \text{NVP}
\]  

(25)

The correctness of Eq. (25) is similar to that of Eq. (7). If both \( (W_A - W_B) \) and \( (Z_A - Z_B) \) follow normal distribution, we can still use the approach in Section 4. However, since the values of coefficients (such as \( c_i^A \) and \( c_i^B \)) can be greater than 1, they do not necessarily follow the normal distribution. But fortunately, if the values of such coefficients are not large, they can still be modeled as normal distributions approximately, as discussed next.

5.2. Background: Lyapunov CLT

For all \( 1 \leq i \leq n \), let the independent (but not necessarily identical) random variables \( V_i \) that take on the values \( \nu_i \) or \( -\nu_i \) with 50% probability each \( (\nu_i \geq 1) \). We will show that under certain conditions the sum \( \sum_{i=1}^{\nu_i} V_i \) is close to normally distributed for large \( n \). We do so using the Lyapunov version of the Central Limit Theorem (see, e.g., [3, p. 371]), stated as follows:

**Theorem 3** (Lyapunov). For a sequence of independent random variables \( X_1, X_2, X_3, \ldots \) with finite expected values \( \mu_i \) and variances \( \sigma_i^2 \), if for some \( \delta > 0 \) we have that

\[
\lim_{n \to \infty} \frac{\sum_{i=1}^{\nu_i} E[|X_i - \mu_i|^{2+\delta}]}{(\sum_{i=1}^{\nu_i} \sigma_i^2)^{1+\delta/2}} = 0,
\]

then \( \sum_{i=1}^{\nu_i} X_i \) converges to the normal distribution as \( n \) goes to infinity.

We will use this theorem to show that, as \( n \) gets large, the sum \( \sum_{i=1}^{\nu_i} V_i \) defined above converges to the normal distribution. For each \( V_i \), it is easy to compute that \( E[V_i] = 0 \) and \( \text{Var}[V_i] = \frac{1}{2} (\nu_i^2 + (-\nu_i)^2) = \nu_i^2 \). Hence, for any \( \delta \), we need to show the condition

\[
C = \lim_{n \to \infty} \frac{\sum_{i=1}^{\nu_i} V_i^{2+\delta}}{(\sum_{i=1}^{\nu_i} V_i^2)^{1+\delta/2}} = 0.
\]
converges to zero. Let \( V = \max_{i=1}^n v_i \). Then, since each \( v_i \in [1, V] \), the above value can be bounded as
\[
C \leq \lim_{n \to \infty} n^{2/3} \left( \sum_{i=1}^n V_i \right)^{2/3} = \left( \frac{n^{2/3}}{n^{2/3}} \right)^{2/3}.
\]

If we pick \( \delta \gg 1 \) and assume that \( V^2 \ll n \), then we have that \( C \to 0 \).

Thus we can assume that the sum \( \sum_{i=1}^n V_i \) is normally distributed if each \( v_i \in [1, V] \) where \( V = o(\sqrt{n}) \).

5.3. Algorithm framework

According to Section 5.2, if all the coefficients, \( c_i^A \), \( c_i^B \) and \( |c_i^A - c_i^B| \), are no greater than \( V \), then both \((Z_A - Z_B)\) and \((W_A - W_B)\) are normally distributed and the approaches in Section 4 still work. More precisely, we can use Algorithms 1 and 2 and the proofs are analogous to Theorems 1 and 2.

But in real applications, there may exist a few frequent tuples, violating our assumption about the coefficients. To overcome this issue, we separately measure the frequencies of all high-frequency elements using the following four steps.

1. From remote sites to the coordinator: Site \( A \) (B) transmits a list \( L_A \) (\( L_B \)) to the coordinator with each entry described as \((x, y, c)\), where \((x, y)\) is a tuple whose frequency \( c \) exceeds the threshold \( V \), \( V = o(\sqrt{N_A + N_B}) \).

2. From the coordinator to remote sites: The coordinator then creates two lists, \( L'_A \) and \( L'_B \), each containing all \( x \) values in \( L_B \) (\( L_A \)) but not in \( L_A \) (\( L_B \)). If the list is not empty, it sends \( L'_A \) (\( L'_B \)) to site \( A \) (\( B \)).

3. From remote sites to the coordinator: Site \( A \) (\( B \)) then creates a new list \( L''_A \) (\( L''_B \)) of all elements in \( L'_A \) together with their frequencies at \( A \), \( (x, y, c) \).

4. From the coordinator to remote sites: Finally, the coordinator updates \( L_A \) (\( L_B \)) by merging with \( L''_A \) (\( L''_B \)).

**Example 4.** Assume the sites \( A \) and \( B \) store sub-relations \( C_1 \) and \( C_2 \) in Table 1 respectively. The threshold is set to 2, only surpassed by \((222, Dick Dole)\) in site \( B \). At Step 1, site \( A \) and \( B \) send \( L_A = \emptyset \) and \( L_B = \{(222, Dick Dole, 3)\} \) to the coordinator respectively. At Step 2, the coordinator creates two lists, \( L'_A = \{222\} \) and \( L'_B = \emptyset \), but only sends \( L'_A \) to site \( A \) since it is non-empty. At Step 3, site \( A \) sends a new list \( L''_A = \{(222, Dickie Dole, 2)\} \) to the coordinator. Finally, the coordinator updates \( L_A \) to \((222, Dickie Dole, 2)\) by merging \( L''_A \), and leaves \( L_B \) unchanged.

After preprocessing the data, the coordinator knows all elements with frequency exceeding the threshold in at least one site. Then, the samples can be adjusted accordingly. For example, the coordinator updates each sample in \( \{z_A^2\} \) as:
\[
z_A^2 \leftarrow z_A^2 - 2 \cdot h_l(222).
\]

In this way, all tuples are divided into two groups. The first group contains frequent tuples, and data inconsistency is computed exactly. If in the above example, this group contains all tuples \( t \) such that \( t[\text{SSN}] = 222 \), having \( 6 = (2 \times 3) \) violating pairs. The second group contains all infrequent tuples, and the errors are estimated by using the aforementioned approaches.

6. Multiple sites

We then study how to work for multiple sites, without the uniqueness property. Since the violating pairs only occur between different remote sites, \( \text{NVP} \) for all \( \text{ψ} \) sites can be computed as:
\[
\text{NVP} = \sum_{1 \leq i < j \leq \text{ψ}} \text{NVP}_{i,j},
\]
where \( \text{NVP}_{i,j} \) denotes the number of violating pairs between sites \( i \) and \( j \).

In an arbitrary site \( k \), we maintain two random variables \( Z_k \) and \( W_k \), where \( Z_k = \sum_{i=1}^{N_k} h(t_i^k[X]) \), and \( W_k = \sum_{i=1}^{N_k} g(t_i^k[X], t_i^k[Y]) \). For any two different sites \( k \) and \( l \), we define \( Z_{k,l} \) and \( W_{k,l} \) as:
\[
Z_{k,l} = (Z_k - Z_l)^2, \quad \text{and} \quad W_{k,l} = (W_k - W_l)^2.
\]

As discussed in Section 5.1, we have:
\[
E[Z_{k,l}] = E[Z_k] + 2 \cdot \text{NVP}_{k,l}.
\]

We define \( Z \) and \( W \) as:
\[
Z = \sum_{1 \leq k < \text{ψ}} Z_{k,l}, \quad \text{and} \quad W = \sum_{1 \leq k < \text{ψ}} W_{k,l}.
\]

According to the linearity of expectation,
\[
E[W] = E[Z] + 2 \cdot \text{NVP}.
\]

In this section, we first study the variance of \( W \) and \( Z \), following which two algorithms to deal with CoD and QoD are introduced.

6.1. Variance study of \( Z \) and \( W \)

We consider \( Z_k \) and \( Z_l \). Without loss of generality, assume the first \( m \) elements are common, while the rest are distinct. So, there are in total \( M_{k,l} = (M_k + M_l - m) \) distinct elements \( (x_i)'s\). Then, \( Z_k - Z_l \) is the linear sum of some hash operators, i.e., \( Z_k - Z_l = \sum_{i=1}^{M_{k,l}} d_i \cdot h(x_i) \). Here, the coefficient \( d_i \) is defined as:
(i) if \( 1 \leq i \leq m \), \( d_i = c_i^A - c_i^B \); (ii) if \( m < i \leq M_k \), \( d_i = c_i^B \); (iii) if \( M_k < i \leq M_{k,l} \), \( d_i = -c_i^{l-M_k+m} \). Let \( P_{k,l} \) denote the square sum of all coefficients, \( Q_{k,l} \) denote the quad sum of all coefficients,
and $R_{k,l}$ denote the crossed square sum of all coefficients, i.e.,

$$P_{k,l} = \sum_{i=1}^{M_k} d_i^2$$

$$Q_{k,l} = \sum_{i=1}^{M_k} d_i^4$$

Subsequently, we show how to compute the values of $E[Z_{k,l}]$, $E[Z_{k,l}^2]$ and $E[Z_{k,l} \cdot Z_{i,j}]$.

$$E[Z_{k,l}] = E \left[ \sum_{l=1}^{M_l} (d_l h(x_l)) \right]^2 + 2 \sum_{1 \leq l < j \leq M_l} E(d_l h(x_l) h(x_j)) = R_{k,l}$$

The above equation holds since for any $i,j, i \neq j$, we have: $E[d_i d_j h(x_i) h(x_j)] = 0$. Next, we consider the upper bound of $E[Z_{k,l} \cdot Z_{i,j}]$. Let $d_1, \ldots, d_{M_k}$ denote the coefficients for $Z_k - Z_l$, and $e_1, \ldots, e_{M_l}$ denote the coefficients for $Z_l - Z_j$. In other words, $Z_k - Z_l = \sum_{i=1}^{M_k} d_i h(x_i)$, and $Z_l - Z_j = \sum_{i=1}^{M_l} e_i h(y_i)$. Without loss of generality, assume the first $\zeta$ elements are common, i.e., $\forall u \leq \zeta, x_u = y_u$. Then,

$$E[Z_{k,l} \cdot Z_{i,j}] = E \left[ \sum_{u=1}^{M_k} (d_u h(x_u))^2 + \sum_{1 \leq u < v \leq M_k} 2d_u d_v h(x_u) h(x_v) \right] \cdot \left( \sum_{u=1}^{M_l} (e_u h(y_u))^2 + \sum_{1 \leq u < v \leq M_l} 2e_u e_v h(y_u) h(y_v) \right)$$

$$= P_{k,l} \cdot P_{i,j} + \sum_{1 \leq u < v \leq \zeta} 4d_u d_v e_u e_v \leq 3 \cdot P_{k,l} \cdot P_{i,j}$$

Similarly, $E[Z_{k,l}^2]$ is computed below. We omit the items like $d_i d_j^2 h(x_i) h(x_j)$, $d_i d_j h(x_i) h(x_j)^2$, or $d_i d_j d_u d_v h(x_i) h(x_j) h(x_u) h(x_v)$, where $i \neq j \neq u \neq v$ since the corresponding means are zero.

$$E[Z_{k,l}^2] = E \left[ \sum_{i=1}^{M_k} (d_i h(x_i))^4 \right] + 6 \cdot E \left[ \sum_{1 \leq i < j \leq M_l} (d_i h(x_i) h(x_j))^2 \right] = Q_{k,l} + 6 \cdot R_{k,l}$$

We compute $E[Z]^2$ and $E[Z^2]$ to obtain $Var[Z]$:

$$E[Z^2] = E \left[ \sum_{1 \leq k < l \leq \psi} Z_{k,l}^2 + \sum_{1 \leq k < \psi} Z_{k,l} \cdot Z_{i,j} \right]$$

$$\leq \sum_{1 \leq k < \psi} (3 \cdot P_{k,l}^2 - 2 \cdot Q_{k,l}) + \sum_{1 \leq k < \psi} 3 \cdot P_{k,l} \cdot P_{i,j} \leq 3(E[Z])^2$$

$$\left( E[Z^2] \right)^2 \leq \left( \sum_{1 \leq k < \psi} P_{k,l} \right)^2 = \sum_{1 \leq k < \psi} P_{k,l}^2 + \sum_{1 \leq k < \psi} P_{k,l} P_{i,j}.$$
to guarantee the precision of the final result. Subsequently, it invokes Algorithm 3 after receiving all necessary samples.

The size of each sample set to deal with QoD is

Theorem 4.

The relative error for tug-of-war (Algorithm 4) is no greater than $\sqrt{\frac{32 \log \delta^{-1}}{|S|}}$ with confidence at least $1 - \delta$.

**Proof.** We use an algorithm $A$ that returns the means of $\xi$ independent samples of $Z$, where $\xi = |S|/(2 \log \delta^{-1})$. Since $\text{Var}[Z] \leq 2 \cdot (E[Z])^2$, by using Chebyshev's inequality, $\Pr[|Z - E[Z]| \geq \epsilon \cdot E[Z]] \leq \frac{\text{Var}[Z]}{\epsilon^2 \cdot (E[Z])^2} \leq \frac{1}{8}$ holds, where $\epsilon = \sqrt{\frac{32 \log \delta^{-1}}{|S|}}$. Let $\hat{Z}$ denote the median from $2\log(1/\delta)$ parallel runs of $A$. Then, we have: $\Pr[|\hat{Z} - E[Z]| > \epsilon E[Z]] \leq \delta$. □

**Lemma 6.** To obtain $\epsilon - \delta$ constraint, the number of samples in $S$ will be at least $n = 32 \epsilon^{-2} \log \delta^{-1}$.

The correctness of this lemma directly comes from Lemma 5.

In Algorithm 3, the confidences of $\hat{Z}$ and $\hat{V}$ are both at least $1 - \delta/2$. Then, the upper and lower bounds for $E[\hat{V}] - E[Z]$ are represented as $\frac{\hat{V}}{2\tau} - \frac{\epsilon}{2\tau}$ and $\frac{\hat{V}}{2\tau} - \frac{\epsilon}{2\tau}$ respectively, with confidence at least $1 - \delta$. Then, we output the claims accordingly (lines 9–14).

### 6.3. Dealing with QoD query

Algorithm sketchQoD (Algorithm 5) deals with QoD query for the multiple-site case. At first, we receive two sample sets from each remote site, each with a size of basic. Then, we compute the minimal size of each sample set that is sufficient to guarantee the precision of the final result. Subsequently, it invokes Algorithm 3 after receiving all necessary samples.

**Theorem 4.** The size of each sample set to deal with QoD is $\frac{32 \log(2/\delta)}{\epsilon^2}$, where $\epsilon = \min(\epsilon(1), \epsilon(2))$, and:

$$
\epsilon(1) = \frac{(1 + \eta_1)\hat{V}}{1 - \epsilon'} - \sqrt{\frac{(1 + \eta_1)\hat{V}}{1 - \epsilon'} - 4(\eta_1 \tau_1 + \tau)(\tau_1 - \tau)}
$$

$$
\epsilon(2) = \frac{(1 + \eta_2)\hat{Z}}{1 + \epsilon'''} + \sqrt{\frac{(1 + \eta_2)\hat{Z}}{1 + \epsilon'''} - 4(\eta_2 \tau_2 + \tau)(\tau_2 - \tau)}
$$

where $\eta_1 = \sqrt{\frac{\log(3/\rho_1)}{\log(2/\delta)}}, \eta_2 = \sqrt{\frac{\log(3/\rho_2)}{\log(2/\delta)}}, \epsilon' = \sqrt{\frac{32 \log(2/\delta)}{\text{basic}}}$, and $\epsilon'' = \sqrt{\frac{32 \log(3/\rho_2)}{\text{basic}}}$. 

---

**Algorithm 3:** sketchCoD $(\psi, \tau, \delta, D)$.

```plaintext
1. $n \leftarrow$ the size of each sample set, typically $\frac{D}{8\psi}$;
2. Receive two sets of samples, $\{x_i^k\}$ and $\{w_i^k\}$, each with a length of $n$ from each site $k$;
3. for $i = 1$ to $n$ do
4.   $z_i \leftarrow \sum_{1 \leq k \leq \psi} (x_i^k - z_i^k)^2$;
5.   $w_i \leftarrow \sum_{1 \leq k \leq \psi} (w_i^k - w_i^k)^2$;
6.   $\hat{z} \leftarrow$ tug-of-war($\{z_i\}, \delta/2$);
7.   $\hat{V} \leftarrow$ tug-of-war($\{w_i\}, \delta/2$);
8.   $\epsilon = \sqrt{\frac{32 \log(2/\delta)}{n}}$;
9. if $(\frac{\hat{V}}{2\tau} - \frac{\epsilon}{2\tau})/2 > \tau$ then
10.   return claim 1 holds;
11. else if $(\frac{\hat{V}}{2\tau} - \frac{\epsilon}{2\tau})/2 < \tau$ then
12.   return claim 2 holds;
13. else
14.   return Neither claim holds;
```

**Algorithm 4:** tug-of-war $(S, \delta)$.

```plaintext
1. $\lambda \leftarrow 2 \cdot \log(1/\delta)$;
2. Divide all samples in $S$ into $\lambda$ groups;
3. For each group $i$, compute the mean value as $s_i$;
4. return the median of all elements in $\{s_i\}$;
```
Algorithm 5: sketchQoD $(\tau, \delta, \tau_1, \tau_2, \rho_1, \rho_2)$.

1. Receive two sample sets $\{w_i^k\}$ and $\{z_i^k\}$ from each remote site, each with a size of basic;
2. for $i = 1$ to basic do
   3. $W_i \leftarrow \sum_{1 \leq k \leq \rho} (w_i^k - w_i^k)^2$; $Z_i \leftarrow \sum_{1 \leq k \leq \rho} (z_i^k - z_i^k)^2$;
   4. $W \leftarrow$ tag-of-war($\{W_i\}$, $\rho_1$/3);
   5. $Z \leftarrow$ tag-of-war($\{Z_i\}$, $\rho_2$/3);
   6. $\varepsilon' \leftarrow \sqrt{\frac{32 \log(3/\rho_1)}{\text{basic}}}$;
   7. $n \leftarrow$ the number of samples by using Theorem 4;
8. if $n > \text{basic}$ then
   9. Receive $n$ - basic samples of $z_i^k$, and $n$ - basic samples of $w_i^k$ from each remote site;
10. Execute lines 3–14 in Algorithm 3;

Proof. We first consider the event: $\text{NVP} > \tau_1$. Let $\varepsilon_1$ denote the relative error with confidence at least $1 - \frac{\eta}{2}$ based on $n$ samples. Let $\hat{W}$ and $\hat{Z}$ denote the estimates of $W$ and $Z$. Then, $\Pr[\hat{W} < (1 - \varepsilon_1)E[W]] < \frac{\eta}{2}$, and $\Pr[\hat{Z} > (1 + \varepsilon_1)E[Z]] < \frac{\eta}{2}$. It means:

$$\Pr[\hat{W} > (1 - \varepsilon_1)E[W] \wedge (\hat{Z} < (1 + \varepsilon_1)E[Z])] > 1 - \frac{2}{3}\rho_1$$

(33)

Eq. (33) holds due to the evaluation condition (line 9, Algorithm 3), where $\varepsilon$ is the relative error based on $n$ samples and confidence $\delta$. According to Lemma 5, $\varepsilon_1 = \sqrt{\frac{\log(3/\rho_1)}{\log(2/\delta)}} \cdot \varepsilon$. Let $\eta_1 = \sqrt{\frac{\log(3/\rho_1)}{\log(2/\delta)}}$ since $E[W] - E[Z] = 2 \cdot \text{NVP} \geq 2\tau_1$, we have:

$$\Pr[2(\eta_1 \tau_1 + \tau)\varepsilon^2 - (1 + \eta_1)(E[W] + E[Z])\varepsilon + 2(\tau_1 - \tau) < 0] < \frac{2}{3}\rho_1.$$

However, $E[W] + E[Z]$ cannot be obtained exactly, so we estimate the value based on basic samples. Note that $\hat{W}$ and $\hat{Z}$ are the estimated values of $W$ and $Z$ based on basic samples (lines 4–5). Let $\varepsilon' = \sqrt{\frac{32 \log(3/\rho_1)}{\text{basic}}}$. By Lemma 6, we have

$$\Pr[|\hat{W} - E[W]| > \varepsilon' E[W]] < \rho_1/3.$$

Subsequently, since $E[W] \geq E[Z]$, the above inequality changes to:

$$\Pr[2(\eta_1 \tau_1 + \tau)\varepsilon^2 - (1 + \eta_1)(\frac{2\hat{W}}{1 - \varepsilon'})\varepsilon + 2(\tau_1 - \tau) < 0] < \rho_1.$$

Hence, the upper bound of $\varepsilon$ should be set to:

$$\varepsilon = \frac{(1 + \eta_1)\hat{W} - \sqrt{(1 + \eta_1)\hat{W}^2 - 4(\eta_1 \tau_1 + \tau)(\tau_1 - \tau)}}{2(\eta_1 \tau_1 + \tau)}.$$

(34)

Similarly, the relative error for event: $\text{NVP} < \tau_2$ can also be computed. We omit the details, and only give the result in Eq. (32). □

6.4. Performance analysis

The effectiveness of all aforementioned methods is explicit since the query results (with error bounded) are output reliably. Another topic is about the efficiency of the communication cost. In our work, $\text{NVP} = (E[W] - E[Z])/2$, and both $E[W]$ and $E[Z]$ can be estimated with $\varepsilon$-$\delta$ constraint by using $O(\varepsilon^{-2} \log(\delta^{-1}))$ samples (Lemma 6). So, when $\text{NVP} = O(\varepsilon \cdot E[W])$, the communication cost for estimating $\text{NVP}$ is also $O(\varepsilon^{-2} \log(\delta^{-1}))$. In [17], Kivinen and Mannila also propose a sampling-based method to estimate $\text{NVP}$, with a conclusion introduced below.

Proposition 1. (Proposition 3.3 in [17]) If the function dependency $X \rightarrow Y$ is $\varepsilon$-bad with respect to $g_1$ in $r$, then the probability of drawing from $r$ a sample $s$ of size $m$ such that $X \rightarrow Y$ holds in $s$ is at most $\delta$ from $m \geq 2(\frac{1}{\varepsilon} \ln \frac{1}{\delta})$.

In this proposition, $g_1$ is defined as $\text{NVP}/|r|^2$, and the term “$\varepsilon$-bad” means $g_1 \geq \varepsilon$. This work can be adapted to handle CoD query for distributed databases. We call it KM.

• KM method: Given communication cost $D$, KM will gather a set of $D/8$ tuples (assume $X$ and $Y$ fields are integers) uniformly from all remote sites at first, and then claim “$\text{NVP} \geq \tau$” if at least one violating pair is detected, where $\tau = \frac{16 \ln (3/\delta)}{D} \cdot |r|^2$ ; otherwise, no claim is output.
By Proposition 1, it seems that KM method is more efficient than ours since the communication cost is proportional to $\epsilon^{-1}$, while ours is proportional to $\epsilon^{-2}$. But the major difference is that the $\epsilon$ in KM method is defined for $NVP/|r|^2$, while the $\epsilon$ in our work is defined for $NVP/E|W|$. In real applications, $E|W| \ll |r|^2$. Let's take the unique, two-site case as an example. Assume each site has 1 million tuples. Then, $|r|^2 = 4 \times 10^{12}$, while $E|W|$ is $2 \times 10^6$ in the worst case. Although the gap between them may be shortened when the unique constraint is released, the difference is still significant if the number of identical tuples is not large. We will report later in Figs. 2 and 3 that given 64KB or 128KB communication cost, our methods are sufficient to give some high-quality claims, while KM method needs far more communication cost.

In addition, our methods have two more characteristics than KM method. First, our methods are hashing-based, supporting delete operator by nature, so that the data structure can be maintained incrementally. But as a sampling-based method, KM cannot maintain the sample set incrementally. Second, our methods can deal with two kinds of queries, including CoD and QoD, while KM method can only support CoD query.

7. Lower bound analysis

For proving lower bounds for this problem, we will use the communication complexity model. In this model, two parties (Alice and Bob) have disjoint parts of the input and have to compute the solution to a problem on this input with minimal communication. They can perform arbitrary computation and only the number of bits of communication (perhaps over multiple rounds) matter. By reducing communication complexity problems with known lower bounds to the problem in this paper, we will establish lower bounds for our problem. Note that the lower bounds apply not just for a single round, but also for an arbitrary number of rounds of communication. Moreover, the bounds apply both in the case when data are shipped between the two sites, as well as when both sites ship the data to a central server.

7.1. Deterministic lower bound

We first show a deterministic (non-randomized) lower bound for detecting the presence of violations to illustrate the fact that it is necessary to use a randomized algorithm for this problem. The specific technique we will use for this lower bound is called the fooling set method. Let $T$ be the collection of all sets of precisely $n = N_A = N_B$ tuples and let $f: T \times T \rightarrow \{0, 1\}$ be the function that maps pairs of sets of tuples to whether or not there are violations (1 if there are one or more violations, 0 otherwise). The fooling set method uses the following theorem from the book by Kushilevitz and Nisan:
Theorem 5 ([19]). If there is a set $S \subseteq T \times T$ and a value $b \in \{0, 1\}$ such that for all $(x, y) \in S$, $f(x, y) = b$ and for all distinct $(x_1, y_1), (x_2, y_2) \in S$, either $f(x_1, y_2) \neq b$ or $f(x_2, y_1) \neq b$, then the deterministic communication complexity of $f$ is at least $\log_2|S|$.

We will use this theorem by constructing a fooling set of size $2^n$. This means that any deterministic algorithm for this problem must communicate at least $n = \min(N_A, N_B)$ bits.

Theorem 6. Any deterministic protocol that distinguishes the case when there is a violation between sets with at most $n$ tuples in each must use at least $\Omega(1/n)$ communication.

Proof. The fooling set we will use is the set $S = \{(t, t)| t_i[X] = i, t_i[Y] = z, 1 \leq i \leq n, z \in \{0, 1\}\}$. In other words, the $t_i[X]$'s take on the values $1, \ldots, n$ and the $t_i[Y]$'s are all the possible combinations of $0$'s and $1$'s. Since there are $2^n$ distinct sets of such tuples possible (since each $t_i[X]$ could have $t_i[Y] = 0$ or $t_i[Y] = 1$), this set has size $2^n$.

To see that $S$ is indeed a fooling set, note that for all $(x, x) \in S$, $f(x, x) = 0$ since there can be no violations between identical sets of tuples. On the other hand, for any distinct $(x, y), (y, y) \in S$, the columns must be different at some position (otherwise $x = y$) and hence there is a violation causing $f(x, y)$ (or equivalently $f(y, x)$) to be equal to $1$.

Since $S$ described above is a fooling set of size $2^n$, any deterministic communication scheme for detecting a violation must have at least $n$ bits of communication. Since it would be prohibitive to expect linear communication $\Omega(n)$ for this problem—this would be tantamount to transporting the entire set of tuples—this shows that a non-randomized algorithm will not work for this problem. □

7.2. Randomized lower bound

Next, we show that a randomized algorithm that always gives the exact answer requires a lot of communication. Let $T$ be the collection of all sets of precisely $n = N_A = N_B$ tuples and let $f : T \times T \rightarrow \{0, 1\}$ be the function that maps pairs of sets of tuples to whether or not there are violations (1 if there are one or more violations, 0 otherwise). We demonstrate a randomized lower bound via a reduction from the communication complexity problem of disjointness to this decision problem, $f$. The disjointness problem is simply, for Alice’s input $x \in \{0, 1\}^n$ and Bob’s input $y \in \{0, 1\}^n$, determine whether there is some $i$ such that $x_i = y_i = 1$. This problem has a lower of bound of $\Omega(n)$ even when the protocol is allowed to be randomized with constant failure probability (Razborov [26]; Kalyanasundaram and Schnitger [16]).

Theorem 7. Any randomized protocol that distinguishes the case when there is a violation between sets with at most $n$ tuples in each must use at least $\Omega(n)$ communication.
Proof. The reduction proceeds as follows. Alice converts her vector by creating a tuple with \(t_i[X] = i\) and \(t_i[Y] = 0\) for each non-zero index \(i\) in her vector. Bob does the same except that each of his \(t_i[Y] = 1\). Since the vectors are of length \(n\), both create at most \(n\) tuples.

In the case that Alice and Bob’s vectors are disjoint, there are no tuples that share the same \(t_i[X]\) values and hence there can be no violations. On the other hand, when Alice and Bob have a 1 at some index, this index causes a violation. Hence, there is a violation in the constructed sets if and only if Alice and Bob’s vectors are not disjoint. Since a scheme for detecting violations could easily be used to detect disjointness, it follows that the randomized lower bound for set disjointness of \(\Omega(n)\) applies to our problem as well. □

7.3. Randomized approximation lower bound

The result from the last section indicates that the problem requires linear communication not just when randomization is allowed, but also if there is approximation. Consider the case when there is a randomized approximation algorithm for this problem. Then, so long as the approximation ratio is less than 1 (i.e., less than 100% error), such an algorithm could always be used to solve the disjointness problem since it would have to always give an answer of 0 for the disjoint case and a positive answer for the non-disjoint case.

The above issue is usually endemic in problems in which the answer could be zero (since the algorithm must be perfect, no matter what the approximation ratio, in this case). To avoid this, we consider the modified problem in which we are guaranteed that the number of violations is at least some minimum number \(\beta > 0\).

For the at-least-\(\beta\) case, we have the following lower bound:

**Theorem 8.** Any randomized algorithm that can distinguish the case when there are at most \(\beta\) violations from one in which there are at least \(\alpha + \beta\) violations in pairs of sets with at most \(\alpha \Delta + \beta\) tuples requires at least \(\Omega(\Delta)\) communication.

**Proof.** This theorem arises from a modification of the reduction in the previous section to always include \(\beta\) violations and to amplify each intersecting element by a factor of \(\alpha\). More specifically, the reduction starts with Alice and Bob’s subsets of \([1, \ldots, \Delta]\) and reduces them to our problem by adding \(\alpha\) tuples for each non-zero index \(i\) in Alice’s vector (e.g., \(t_i[X] = t_1, \ldots, t_n[X] = i_o\)) into Alice’s set with each \(t_i[Y] = 0\) and similarly for Bob, except each of Bob’s \(t_i[Y] = 1\). Both Alice and Bob also enter \(\beta\) violating tuples (with the same \(t_i[X]\) values and different \(t_i[Y]\) values).

In the case that Alice and Bob’s sets are disjoint, there are exactly \(\beta\) violations. If, on the other hand, there are any common elements, then there are at least \(\beta + \alpha\) violations. Since distinguishing these cases solves the disjointness problem, this leads to an \(\Omega(\Delta)\) communication lower bound, even when the algorithm is allowed to be randomized. □

This theorem can be more easily interpreted in the form of this corollary:

**Corollary 1.** Any randomized algorithm that can distinguish the case when of \(\beta\) violations between sets of at most \(n\) tuples must use at least \(\Omega(n/\beta)\) communication.

This follows from substituting \(\alpha = \beta\) and \(\Delta = n/\beta - 1\) into the above theorem. This corollary implies that there is no communication-efficient randomized algorithm for two-approximating the number of errors when the number of violations is much smaller than the size of the set of tuples.

8. Experiments

We report some experimental results in this section. All codes were written in Java, and run on a Windows PC with an Intel(R) Core i7-3770 CPU and 32GB memory. We use KM method (Section 6.4) as the baseline.

8.1. Description of data set

We use four data sets, including synthetic and real-world datasets. The first three sets are for two-site case, while the last one is for four-site case. The last two sets are based on a real-world data set that records all the requests made to the 1998 World Cup Web site from the start of day 49 to the end of day 55 in 1998.¹ All requests are handled by 33 HTTP servers deployed in four regions, including Santa Clara, Plano, Herndon and Paris (they are anonymized as integers from 0 to 3). This real-world data set is also used in [7].

- **Uniq-num-dr-mr-ir.** It contains two data sets. The first one contains num records of two integers. All integers in the first column are distinct. The second one is generated by copying all records in the first one, thereafter dropping \(dr \times num\) records, modifying the second column of \(mr \times num\) records and adding \(ir \times num\) different new records.
- **Dupli-num-dr-mr-ir-\(\mu\).** It contains two data sets, generated in a way similar to that of Uniq-num-dr-mr-ir except that some records may share an identical value in the first column. The frequencies of \(\mu\) records selected in random are assigned as: \([\frac{\mu}{k}], [\frac{\mu}{k}], \ldots, 1\), while the frequencies of the rest records are randomly selected from \([1, 4]\).

• WorldCup-Two. It contains two data sets for the requests handled by HTTP servers at day 49 in regions 0 and 1 respectively. Each record in the data set is in format of (clientID, minute, objectID). To handle QoD query, it needs to compute the minimal necessary communication cost to satisfy all uniqueness requirement. Fig. 4 illustrates the communication cost for QoD query upon four different data sets. In all tests, more than 300KB communication cost for one case since its threshold value is much greater than any parameter. In all situations, FtestCoD outperforms sketchCoD, since FtestCoD takes use of the underlying data distribution, while sketchCoD is based on the usage of Chebyshev’s inequality. KM method cannot output any claim in any case since its threshold value is always has the success rate equal to zero. In comparison, the success rate of FtestCoD method rises to 1.0 when given 150K communication cost. The sketchCoD algorithm behaves a bit lower than FtestCoD, but still significantly better than KM.

Table 3 lists the statistics of some data sets to be used later.

### 8.2. Reports on the two-site case

**Testing CoD query.** FtestCoD and sketchCoD (Algorithms 1 and 3) may not output claim 1 or 2 on some situations. We define *success rate* as the probability that either claim 1 or 2 is output. Figs. 2 and 3 record success rate under different communication cost by executing each method for 10 times, where \( \delta = 0.05 \). The x-axis represents threshold \( \tau \), and the y-axis represents success rate. We use four data sets, including Uniq-1M-0.5–0.3–0.5, Uniq-1M-0.05–0.10–0.05, Dupli-1M-0.5–0.3-0.5–100, and WorldCup-Two. For Uniq-1M-0.5–0.3–0.5, the data in two sites are significantly different, i.e., \( |S_1 \triangle S_2| = 2 \times 0.5 \times 1M = 1.0M \), and \( VS = 0.3 \times 1M = 300K \). For Uniq-1M-0.05–0.10–0.05, the data in two sites are similar, \( |S_1 \triangle S_2| = 100K = (2 \times 0.05 \times 1M) \), and \( VS = 0.1 \times 1M = 100K \). Dupli-1M-0.5–0.3-0.5–100 and WorldCup-Two do not have the uniqueness property. In all situations, FtestCoD outperforms sketchCoD, since FtestCoD takes use of the underlying data distribution, while sketchCoD is based on the usage of Chebyshev’s inequality. KM method cannot output any claim in any case since its threshold value is much greater than any \( \tau \) value in x-axis. For example, when \( D = 128KB \), it will generate 16K samples. However, the threshold value is: \( \frac{16 \times \ln(1/0.05)}{128 \times 10^6} \times (2 \times 10^K)^2 = 1.5 \times 10^2 \), far greater than NVP.

In general, when given more communication cost, both FtestCoD and sketchCoD become more effective to output meaningful claims. For example, given 64KB, success rate for sketchCoD is zero when \( \tau = 480K \). But when the communication cost is doubled, the corresponding success rate rises to 1.0, as shown in Figs. 2(a) and 3(a).

**Testing QoD query.** To handle QoD query, it needs to compute the minimal necessary communication cost to satisfy all requirements. Fig. 4 illustrates the communication cost for QoD query upon four different data sets. In all tests, \( \delta = 0.05 \), \( \rho_1 = \rho_2 = 10\% \), basic=1000. Let tau rate \( (\tau r) \) denote a parameter to control the values of \( \tau_1 \) and \( \tau_2 \), such that \( \tau r = \frac{\tau_1}{\tau_2} \). We set \( \tau r \) to 2 and 1.4 respectively.

FtestQoD consumes less communication cost than sketchQoD. The communication cost will decrement when \( \tau r \) rises, since the gaps between \( \tau \) and \( \tau_1 \), \( \tau_2 \) also increase. For the similar reason, the communication cost also decrements when \( \tau \) increases. In addition, the communication cost requirement differs a lot for different data sets. For a QoD query with \( \tau = 200K \) and \( \tau r = 2.0 \). FtestQoD method only consumes around 10KB communication cost for Uniq-1M-0.05–0.10–0.05, but more than 300KB communication cost for WorldCup-Two.

**Efficiency.** We next evaluate the time-efficiency of the proposed methods. The overall executing cost includes two parts: (i) the time to transfer data from remote sites to the coordinator site, and (ii) the time to compute the final result in the coordinator site. Since the communication cost is highly dependent on the bandwidth of the network, and varies a lot in different environments, we only report the executing time in the coordinator site, as illustrated in Fig. 5. Readers can refer to Figs. 2–4, and 6 to learn the communication cost.

Fig. 5 (a) tests the efficiency of handling CoD query upon the data set Uniq-1M-0.5–0.3-0.5. Upon all situations, all methods run efficiently, completed in 1 ms. When the communication cost \( D \) increases from 64KB to 256KB, the running time of two methods also increases linearly. We optimize the processing of KM method, by sorting all samples in each remote site, so that the coordinator site can process data linearly. Although KM runs faster than our approaches, its efficiency is lower than ours. Fig. 5(b and c) shows the success rate when \( \tau = 120K \) or 140K. We can see that KM method always has the success rate equal to zero. In comparison, the success rate of FtestCoD method rises to 1.0 when given 150K communication cost. The sketchCoD algorithm behaves a bit lower than FtestCoD, but still significantly better than KM.

Fig. 5 (b) tests the efficiency of handling QoD query upon the data set Uniq-1M-0.5–0.3-0.5. The x-axis represents \( \tau \) value, and the y-axis represents the running efficiency. Under all situations, both approaches runs efficiently. Given different \( \tau \)
Fig. 4. The total communication cost for QoD query.

(a) Uniq-1M-0.5-0.3-0.5
(b) Uniq-1M-0.05-0.1-0.05
(c) Dupli-1M-0.05-0.1-0.05
(d) Real data set

Fig. 5. The running efficiency.

(a) For CoD query
(b) For CoD query, τ = 120K
(c) For CoD query, τ = 140K
(d) For QoD query
value, the coordinator site needs to compute $n$ immediately, and get samples from remote sites. Since a smaller $\tau$ value tends to have a larger $n$ value, we can observe that the running time decrements when $\tau$ grows.

Note that the reports in Fig. 5 do not consider the cost on generating the samples. The running cost on per-sample generating is linear to the number of hash functions utilized.

8.3. Results on multiple-site case

Fig. 6 shows the performance for the multiple-site case upon WorldCup-Four data set. Fig. 6(a) and (b) are for 1-day copy, while Fig. 6(c) and (d) are for 1-week copy. For CoD testings, we compare sketchCoD with KM method. We see that even given 128KB communication cost, the success rate rises to 1.0 when $\tau \geq 2.8M$. Contrarily, the threshold for KM method is more than $1.5 \times 10^9$, recall that the number of records is greater than 2 millions. Similar phenomenon happens upon the 1-week copy. For QoD testings, we report the communication cost for different $tr$ values. When $\tau$ increases, the communication cost continues to shrink.

9. Conclusion

This paper studies error estimation of approximate functional dependency in a distributed environments. Error estimation has been studied for 20 years since [17], but the work on distributed databases, aiming at lowering the communication cost, is still rare. We study two kinds of semantics in this field, namely CoD and QoD, along with novel solutions to deal with them. Subsequently, we propose novel solutions to deal with these queries, along with lower bound analysis in theorem.

There still exist some work to be done in future. First, our study focuses on NVP, but how to deal with other measures remains as a piece of future work. Second, we assume an FD holds in each remote site since this condition can be satisfied easily in each remote site. However, there may still exist some situations which allow a few violations in each site. How to deal with this model is also interesting.

Acknowledgments

The work of Jin, Zhang and Zhou is supported in part by the 973 Program of China (no. 2012CB316203), NSFC (61370101, and U1401256), Shanghai Knowledge Service Platform Project (no. ZF1213), and Innovation Program of Shanghai Municipal Education Commission (14ZZ045). The work of Lall and Xu is supported in part by US NSF through a joint grant consisting of awards CNS 1218092 and CNS 1217758; Xu is supported by NSF also through award CNS 1423182.
References


