Uncertainty Aware Query Execution Time Prediction

Wentao Wu†  Xi Wu†  Hakan Hacıgümüş‡  Jeffrey F. Naughton†
†Department of Computer Sciences, University of Wisconsin-Madison
‡NEC Laboratories America
{wentaowu, xiwu, naughton}@cs.wisc.edu, hakan@nec-labs.com

ABSTRACT

Predicting query execution time is a fundamental issue underlying many database management tasks. Existing predictors rely on information such as cardinality estimates and system performance constants that are difficult to know exactly. As a result, accurate prediction still remains elusive for many queries. However, existing predictors provide a single, point estimate of the true execution time, but fail to characterize the uncertainty in the prediction. In this paper, we take a first step towards providing uncertainty information along with query execution time predictions. We use the query optimizer’s cost model to represent the query execution time as a function of the selectivities of operators in the query plan as well as the constants that describe the cost of CPU and I/O operations in the system. By treating these quantities as random variables rather than constants, we show that with low overhead we can infer the distribution of likely prediction errors. We further show that the estimated prediction errors by our proposed techniques are strongly correlated with the actual prediction errors.

1. INTRODUCTION

The problem of predicting query execution time has received a great deal of recent research attention (e.g., [6, 7, 19, 20, 47, 48]). Knowledge about query execution time is essential to many important database management issues, including query optimization, admission control [43, 49], query scheduling [14], and system sizing [45]. Existing predictors rely on information such as cardinality estimates and system performance constants that are difficult to know exactly. As a result, accurate prediction remains elusive for many queries. However, existing predictors provide a single, point estimate of the true execution time, but fail to characterize the uncertainty in the prediction.

It is a general principle that if there is uncertainty in the estimate of a quantity, systems or individuals using the estimate can benefit from information about this uncertainty. (As a simple but ubiquitous example, opinion polls cannot be reliably interpreted without considering the uncertainty bounds on their results.) In view of this, it is somewhat surprising that something as foundational as query running time estimation typically does not provide any information about the uncertainty embedded in the estimates.

There is already some early work indicating that providing this uncertainty information could be useful. For example, in approximate query answering [27, 30], approximate query results are accompanied by error bars to indicate the confidence in the estimates. It stands to reason that other user-facing running time estimation tasks, for example, query progress indicators [13, 35], could also benefit from similar mechanisms regarding uncertainty. Other examples include robust query processing and optimization techniques (e.g., [9, 15, 21, 22, 37, 44]) and distribution-based query schedulers [14]. We suspect that if uncertainty information were widely available many more applications would emerge.

In this paper, we take a first step towards providing uncertainty information along with query execution time predictions. In particular, rather than just reporting a point estimate, we provide a distribution of likely running times. There is a subtlety in semantics involved here — the issue is not “if we run this query 100 times what do we think the distribution of running times will be?” Rather, we are reporting “what are the likelihoods that the actual running time of this query would fall into certain confidence intervals?” As a concrete example, the distribution conveys information such as “I believe, with probability 70%, the running time of this query should be between 10s and 20s.”

Building on top of our previous work [48], we use query optimizers’ cost models to represent the query execution time as a function of selectivities of operators in the query plan as well as basic system performance parameters such as the unit cost of a single CPU or I/O operation. However, our approach here is different from that in [48] — we treat these quantities as random variables rather than fixed constants. We then use sampling based approaches to estimate the distributions of these random variables. Based on that, we further develop analytic techniques to infer the distribution of likely running times.

In more detail, for specificity consider the cost model used by the query optimizer of PostgreSQL:

**Example 1 (PostgreSQL’s Cost Model).** PostgreSQL estimates the execution runtime overhead $t_O$ of an operator $O$ (e.g., scan, sort, join, etc.) as follows:

$$t_O = n_a \cdot c_a + n_r \cdot c_r + n_t \cdot c_t + n_i \cdot c_i + n_o \cdot c_o.$$  \hspace{1cm} (1)

Here the $c_i$’s are cost units described in Table 1. Accordingly, the $n_i$’s are then the number of pages sequentially scanned, the number of pages randomly accessed, and so on, during the execution of $O$. The total estimated overhead $t_q$ of a query $q$ is simply the sum of the costs of the individual operators in its query plan. Moreover, as illustrated in [48], the $n_i$’s are actually functions of the input/output cardinalities (or equivalently, selectivities) of the operators. As a
The CPU cost to process a tuple via
The I/O cost to
The I/O cost to
The CPU cost to process a tuple
The CPU cost to process a tuple via index access
The CPU cost to perform an operation (e.g., hash)

<table>
<thead>
<tr>
<th>c</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_0</td>
<td>The I/O cost to sequentially access a page</td>
</tr>
<tr>
<td>c_r</td>
<td>The I/O cost to randomly access a page</td>
</tr>
<tr>
<td>c_t</td>
<td>The CPU cost to process a tuple</td>
</tr>
<tr>
<td>c_i</td>
<td>The CPU cost to access a page</td>
</tr>
<tr>
<td>c_o</td>
<td>The CPU cost to perform an operation (e.g., hash)</td>
</tr>
</tbody>
</table>

result, we can further represent \( t_q \) as a function of the cost units \( c \) and the selectivities \( X \), namely,

\[
t_q = \sum_{O \in \text{Plan}(q)} c_O = g(c, X).
\]  

Perfect predictions therefore rely on three assumptions: (i) the \( c \)'s are accurate; (ii) the \( X \)'s are accurate; and (iii) \( g \) is itself accurate. Unfortunately, none of these holds in practice. First, the \( c \)'s are inherently random. For example, the value of \( c_r \) may vary for different disk pages accessed by a query, depending on where the pages are located on disk. Second, accurate selectivity estimation is often challenging, though significant progress has been made. Third, the equations and functions modeling query execution make approximations and simplifications so they could make errors. For instance, Equation (1) does not consider the possible interleaving of CPU and I/O operations during runtime.

To quantify the uncertainty in the prediction, we therefore need to consider potential errors in all three parts of the running time estimation formula. It turns out that the errors in the \( c \)'s, the \( X \)'s, and \( g \) are inherently different. The errors in the \( c \)'s result from fluctuations in the system state and/or variances in the way the system performs for different parts of different queries. (That is, for example, the cost of a random I/O may differ substantially from operator to operator and from query to query.) We therefore model the \( c \)'s as random variables and extend our previous calibration framework [48] to obtain their distributions. The errors in the \( X \)'s arise from selectivity estimation errors. We therefore also model these as random variables and consider sampling-based approaches to estimate their variance. The errors in \( g \), however, result from simplifications or errors made by the designer of the cost model and are out of the scope of this work. We show in our experiments that even imperfect cost model functions are useful for estimating uncertainty in predictions.

Based on the idea of treating the \( c \)'s and the \( X \)'s as random variables rather than constants, the predicted execution time \( t_q \) is then also a random variable so that we can estimate its distribution. A couple of challenges arise immediately. First, unlike the case of providing a point estimate of \( t_q \), knowing that \( t_q \) is "some" function of the \( c \)'s and the \( X \)'s is insufficient if we want to infer the distribution of \( t_q \) — we need to know the explicit form of \( g \). By Equation (2), \( g \) relies on cost functions that map the \( X \)'s to the \( n \)'s. As a result, for concreteness we have to choose some specific cost model. Here, for simplicity and generality, we leverage the notion of logical cost functions [18] rather than the cost functions of any specific optimizer. The observation is that the costs of an operator can be specified according to its logical execution. For instance, the number of CPU operations of the in-memory sort operator could be specified as \( n_o = \alpha N \log N \), where \( N \) is the input cardinality. Second, while we can show that the distribution of \( t_q \) is asymptotically normal based on our current ways of modeling the \( c \)'s and the \( X \)'s, determining the parameters of the normal distribution (i.e., the mean and variance) is difficult for non-trivial queries with deep query trees. The challenge arises from correlations between selectivity estimates derived by using shared samples. We present a detailed analysis of the correlations and develop techniques to either directly compute or provide upper bounds for the covariances with respect to the presence of correlations. Finally, providing estimates to distributions of likely running times is desirable only if it can be achieved with low overhead. We show that it is the case for our proposed techniques — the overhead is almost the same as that of the predictor in [48] which only provides point estimates.

Since our approach makes a number of approximations when computing the distribution of running time estimates, an important question is how accurate the estimated distribution is. An intuitively appealing experiment is the following: run the query multiple times, measure the distribution of its running times, and see if this matches the estimated distribution. But this is not a reasonable approach due to the subtlety we mentioned earlier. The estimated distribution we calculate is not the expected distribution of the actual query running time, it is the distribution of running times our estimator expects due to uncertainties in its estimation process. To see this another way, note that cardinality estimation error is a major source of running time estimation error. But when the query is actually run, it does not appear at all — the query execution of course observes the true cardinalities, which are identical every time it is run.

Speaking informally, what our predicted running time distribution captures is the “self-awareness” of our estimator. Suppose that embedded in the estimate is a dependence on what our estimator knows is a very inaccurate estimate. Then the estimator knows that while it gives a specific point estimate for the running time (the mean of a distribution), it is likely that the true running time will be far away from the estimate, and it captures this by indicating a distribution with a large variance.

So our task in evaluating our approach is to answer the following question: how closely does the variance of our estimated distribution of running times correspond to the observed errors in our estimates (when compared with true running times)? To answer this question, we estimate the running times for and run a large number of different queries and test the agreement between the observed errors and the predicted distribution of running times, where “agreement” means that larger variations correspond to more inaccurate estimates.

In more detail, we report two metrics over a large number of queries: (M1) the correlation between the standard deviations of the estimated distributions and the actual prediction errors; and (M2) the proximity between the inferred and observed distributions of prediction errors. We show that (R1) the correlation is strong; and (R2) the two distributions are close. Intuitively, (R1) is qualitative; it suggests that one can judge if the prediction errors will be small or large based on the standard deviations of the estimated distributions. (R2) is more quantitative; it further suggests that the likelihoods of prediction errors are specified by the distributions as well. We therefore conclude that the estimated distributions do a reasonable job as indicators of prediction errors.

We start by presenting terminology and notation used throughout the paper in Section 2. We then present the details of how to estimate the distributions of the \( c \)'s and the \( X \)'s (Section 3), the explicit form of \( g \) (Section 4), and the distribution of \( t_q \) (Section 5). We further present experimental evaluation results in Section 6, discuss related work in Section 7, and conclude the paper in Section 8.

### 2. PRELIMINARIES

In most current DBMS implementations, the operators are either unary or binary. Therefore, we can model a query plan with a rooted binary tree. Consider an operator \( O \) in the query plan. We use \( O_l \) and \( O_r \) to represent its left and right child operator, and use
R
The left (right) child operator of
The descendant operators of
An operator in the query plan
The leaf tables of
The left (right) input cardinality of
The subtree rooted at


1L
O
O
O
N
1
(=
R
1
O
Scan
our discussion does not depend on the
functions of the selectivities (i.e., the


X
3. INPUT DISTRIBUTIONS

To learn the distribution of \( t_q \), we first need to know the distributions of the \( c \)'s and the \( X \)'s. We do this by extending the framework in our previous work [48].

3.1 Distributions of the \( c \)'s

In [48], we designed dedicated calibration queries for each \( c \). Consider the following example:

**Example 3** (CALIBRATION QUERY). Suppose that we want to know the value of \( c_t \), namely, the CPU cost of processing one tuple. We can use the calibration query \( \text{SELECT * FROM R} \), where \( R \) is some table whose size is known and is loaded into memory. Since this query only involves \( c_t \), its execution time \( \tau \) can be expressed as \( \tau = |R| \cdot c_t \). We can then run the query, record \( \tau \), and compute \( c_t \) from this equation.

Note that we can use different \( R \)'s here, and different \( R \)'s may give us different \( c_t \)'s. We can think of these observed values as i.i.d. samples from the distribution of \( c_t \), and in [48] we used the sample mean as our estimate of \( c_t \). To quantify the uncertainty in \( c_t \), it would make more sense to treat \( c_t \) as a random variable rather than a constant. We assume that the distribution of \( c_t \) is normal (i.e., Gaussian), for intuitively the CPU speed is likely to be stable and centered around its mean value. Now let \( c_t \sim N(\mu_t, \sigma_t^2) \). It is then a common practice to use the mean and variance of the observed \( c_t \)'s as estimates for \( \mu_t \) and \( \sigma_t^2 \).

In general, we can apply similar arguments to all the five cost units. Due to space limitations, readers are referred to [48] for more details on the calibration procedure. In [48] we only calculated the mean, not the variance, but the extension to the variance is straightforward.

3.2 Distributions of the \( X \)'s

The uncertainties in the \( X \)'s are quite different from those in the \( c \)'s. The uncertainties in the \( c \)'s are due to unavoidable fluctuations in hardware execution speeds. In other words, the \( c \)'s are inherently random. However, the \( X \)'s are actually fixed numbers — if we run the query we should always obtain the same ground truths for the \( X \)'s. The uncertainties in the \( X \)'s really come from the fact that so far we do not have a perfect selectivity estimator. How to quantify the uncertainties in the \( X \)'s therefore depends on the nature of the selectivity estimator used. Here we extend the sampling-based approach used in [48], which was first proposed by Haas et al. [25]. It provides a mathematically rigorous way to quantify potential errors in selectivity estimates. It remains interesting future work to investigate the possibility of extending other alternative estimators such as those based on histograms.

3.2.1 A Sampling-Based Selectivity Estimator

Suppose that we have a database consisting of \( K \) relations \( R_1, ..., R_K \), where \( R_k \) is partitioned into \( m_k \) blocks each with size \( N_k \), namely, \( |R_k| = m_k N_k \). Without loss of generality, let \( q \) be a selection-join query over \( R_{1}, ..., R_{K} \), and let \( B(k, j) \) be the \( j \)-th block of relation \( k \) (\( 1 \leq j \leq m_k \), and \( 1 \leq k \leq K \)). Define

\[
B(L_{1,i_1}, ..., L_{K,i_K}) = B(1, L_{1,i_1}) \times \cdots \times B(K, L_{K,i_K}),
\]

where \( B(k, L_{k,i_k}) \) is the block (with index \( L_{k,i_k} \)) randomly picked from the relation \( R_k \) in the \( i_k \)-th sampling step. After \( n \) steps, we can obtain \( n^K \) such samples (notice that these samples are not independent), and the estimator is defined as

\[
\rho_n = \frac{1}{n^K} \sum_{i_1=1}^{n} \cdots \sum_{i_K=1}^{n} \rho_B(L_{1,i_1}, ..., L_{K,i_K}).
\]

Here \( \rho_n \) is the estimated selectivity of \( q \) (after \( n \) sampling steps), and \( \rho_B \) is the observed selectivity of \( q \) over the sample \( B \). This

We use "relation" and "table" interchangeably in this paper since our discussion does not depend on the set/bag semantics.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O )</td>
<td>An operator in the query plan</td>
</tr>
<tr>
<td>( O_l ) (( O_r ))</td>
<td>The left (right) child operator of ( O )</td>
</tr>
<tr>
<td>( N_l ) (( N_r ))</td>
<td>The left (right) input cardinality of ( O )</td>
</tr>
<tr>
<td>( M )</td>
<td>The output cardinality of ( O )</td>
</tr>
<tr>
<td>( R )</td>
<td>The leaf tables of ( O )</td>
</tr>
<tr>
<td>( X )</td>
<td>The selectivity of ( O )</td>
</tr>
<tr>
<td>( T )</td>
<td>The subtree rooted at ( O )</td>
</tr>
<tr>
<td>( \text{Desc}(O) )</td>
<td>The descendant operators of ( O ) in ( T )</td>
</tr>
</tbody>
</table>

\( N_l \) and \( N_r \) to denote its left and right input cardinality. If \( O \) is unary, then \( O_l \) does not exist and thus \( N_r = 0 \). We further use \( M \) to denote \( O \)'s output cardinality.

Let \( T \) be the subtree rooted at the operator \( O \), and let \( R \) be the (multi)set of relations accessed by the leaf nodes of \( T \). Note that the leaf nodes in a query plan must be \( \text{scan} \) operators that access the underlying tables. We call \( R \) the leaf tables of \( O \). Let \( |R| = \prod_{R \in R} |R| \). We define the selectivity \( X \) of \( O \) to be:

\[
X = \frac{M}{|R|} = \frac{M}{\prod_{R \in R} |R|}.
\]
estimator is shown to be both unbiased and strongly consistent for the actual selectivity \( q \) of \( q \) [25, 48].

By applying the Central Limit Theorem, we can show that

\[
\frac{\sqrt{n}}{\sigma} (\rho_n - \rho) \xrightarrow{d} \mathcal{N}(0, 1).
\]

That is, the distribution of \( \rho_n \) is approximately normal after a large number of sampling steps [25]: \( \rho_n \sim \mathcal{N}(\rho, \sigma_n^2) \), where \( \sigma_n^2 = \sigma^2/n \) and \( \sigma^2 = \lim_{n \to \infty} n \Var[\rho_n] \). We present a more detailed study of \( \Var[\rho_n] \) in Appendix A.1.

However, here \( \sigma_n^2 \) is unknown since \( \sigma^2 \) is unknown. In [25], the authors further proposed the following estimator for \( \sigma^2 \):

\[
S_n^2 = \frac{K}{n-1} \sum_{j=1}^{n} (Q_{k,j,n}/n^{K-1} - \rho_n)^2,
\]

for \( n \geq 2 \) (we set \( S_1^2 = 0 \)). Here

\[
Q_{k,j,n} = \sum_{(i_1, \ldots, i_K) \in \Omega_{k}^{n}(j)} \rho_B(L_{i_1, i_2, \ldots, L_{i_K, i_k}}),
\]

where \( \Omega_{k}^{n}(j) = \{(i_1, \ldots, i_K) \in \{1, \ldots, n\}^K : i_k = j \} \). It can be shown that \( \lim_{n \to \infty} S_n^2 = \sigma^2 \) a.s. As a result, it is reasonable to approximate \( \sigma^2 \) with \( S_n^2 \) when \( n \) is large. So \( \rho_n \approx S_n^2/n \).

### 3.2.2 Efficient Computation of \( S_n^2 \)

Efficiency is crucial for a predictor to be practically useful. We have discussed efficient implementation of \( \rho_n \) in [48]. Taking samples at runtime might not be acceptable since it will result in too many random disk I/O’s. Therefore, we instead take samples offline and store them as materialized views (i.e., sample tables). In the following presentation, we use \( R^* \) to denote the sample table of a relation \( R \). In [48], we further showed that, given a selection-join query, we can estimate the selectivities of all the selections and joins by running the original query plan over the sample tables once. The trick is that, since the block size is not specified when partitioning the relations, it could be arbitrary. We can then let a block be a single tuple so that the cross-product of sample blocks is reduced to the cross-product of sample tuples.

**Example 4** (Implementation of \( \rho_n \)). Let us consider the query plan in Figure 1 again. Based on the tuple-level partitioning scheme, by Equation (4) we can simply estimate \( X_4 \) and \( X_5 \) as

\[
\tilde{X}_4 = \frac{R_1^* \bowtie R_2^*}{|R_1^*| \cdot |R_2^*|} \quad \text{and} \quad \tilde{X}_5 = \frac{R_1^* \bowtie R_2^* \bowtie R_3^*}{|R_1^*| \cdot |R_2^*| \cdot |R_3^*|}.
\]

Also note that we can compute the two numerators by running the query plan over the sample relations \( R_1^* \), \( R_2^* \) and \( R_3^* \) once. That is, to compute \( R_1^* \bowtie R_2^* \bowtie R_3^* \), we reuse the join results from \( R_1^* \bowtie R_2^* \) that has been computed when estimating \( X_4 \).

We now extend the above framework to further compute \( S_n^2 \). For this sake we need to know how to compute the \( Q_{k,j,n} \)’s in Equation (5). Let us consider the cases when an operator represents a selection (i.e., a scan), a two-way join, or a multi-way join query.

---

2Strong consistency is also called *almost sure convergence* in probability theory (denoted as “a.s.”). It means that the more samples we take, the closer \( \rho_n \) is to \( \rho \).

### Selection

In this case, \( K = 1 \) and by Equation (6) \( Q_{k,j,n} \) is reduced to \( Q_{1,j,n} = \rho_B(L_{1,j}) \). Therefore, \( S_n^2 \) can be simplified as

\[
S_n^2 = \frac{1}{n-1} \sum_{j=1}^{n} (\rho_B(L_{1,j}) - \rho_n)^2.
\]

Since a block here is just a tuple, \( \rho_B(L_{1,j}) = 0 \) or \( \rho_B(L_{1,j}) = 1 \). We thus have

\[
S_n^2 = \frac{1}{n-1} \left( \sum_{\rho_B(L_{1,j})=0} \rho_n^2 + \sum_{\rho_B(L_{1,j})=1} (1 - \rho_n)^2 \right)
\]

\[
= \frac{1}{n-1} \left( (n-M)\rho_n^2 + M(1-\rho_n)^2 \right),
\]

where \( M \) is the number of output tuples from the selection. When \( n \) is large, \( n \approx n - 1 \), so we have

\[
S_n^2 \approx \left( 1 - \frac{M}{n} \right)\rho_n^2 + \frac{M}{n}(1-\rho_n)^2 = \rho_0(1 - \rho_n),
\]

by noticing that \( \rho_n = \frac{M}{n} \). Hence \( S_n^2 \) is directly computable for a scan operator once we know its estimated selectivity \( \rho_n \).

### Two-way Join

Consider a join \( R_1 \bowtie R_2 \). In this case, \( Q_{k,j,n} \) \((k = 1, 2)\) can be reduced to

\[
Q_{1,j,n} = \sum_{i_2=1}^{n} \rho_B(L_{1,j, i_2}) \quad \text{and} \quad Q_{2,j,n} = \sum_{i_1=1}^{n} \rho_B(L_{1,i_1, j}).
\]

Again, since a block here is just a tuple, \( \rho_B \) is either 0 or 1. It is then equivalent to computing the following two quantities:

- \( Q_{1,j,n} = \{|t_{1,j} : t_{1,j} \bowtie R_2^* \} \), where \( t_{1,j} \) is the \( j \)th tuple of \( R_1^* \);
- \( Q_{2,j,n} = |R_1^* \bowtie \{t_{2,j}\}| \), where \( t_{2,j} \) is the \( j \)th tuple of \( R_2^* \).

That is, to compute \( Q_{k,j,n} \) \((k = 1, 2)\), conceptually we need to join each sample tuple of one relation with all the sample tuples of the other relation. However, directly performing this is quite expensive, for we need to do \( 2n \) joins here.

We seek a more efficient solution. Recall that we need to join \( R_1^* \) and \( R_2^* \) to compute \( \rho_n \). Let \( R^* = \bigcirc_{i=1}^{r} R_i^* \). Consider any \( t \in R^* \), \( t \) must satisfy \( t = t_{1,i} \bowtie R_{2,i} \), where \( t_{1,i} \in R_1^* \) and \( t_{2,i} \in R_2^* \). Then \( t \) contributes 1 to \( Q_{1,i,n} \) and 1 to \( Q_{2,j,n} \). On the other hand, any \( t \in R_1^* \bowtie R_2^* \) but not in \( R^* \) will contribute nothing to the \( Q \)’s. Based on this observation, we only need to scan the tuples in \( R^* \) and increment the corresponding \( Q \)’s. The remaining problem is how to know the indexes \( i \) and \( j \) as in \( t = t_{1,i} \bowtie R_{2,i} \). For this purpose, we assign an *identifier* to each tuple in the sample tables when taking the samples. This is akin to the idea in data provenance research where tuples are annotated to help tracking the lineages of the query results [23].

### Multi-way Joins

The approach of processing two-way joins can be easily generalized to handle multi-way joins. Now we have

\[
Q_{k,j,n} = |R_1^* \bowtie \cdots \bowtie \{t_{k,j}\} \bowtie \cdots \bowtie R_K^*|.
\]

As a result, if we let \( R^* = \bigcirc_{i=1}^{r} R_i^* \) then any \( t \in R^* \) satisfies \( t = t_{1,i_1} \bowtie \cdots \bowtie t_{k,i_k} \bowtie \cdots \bowtie t_{r,i_r} \). \( t \in R_1^* \bowtie \cdots \bowtie R_K^* \) will contribute 1 to each \( Q_{k,i,n} \) \((1 \leq k \leq K)\) if and only if \( t \in R^* \). Therefore, as before, we can just simply scan \( R^* \) and increment the corresponding \( Q \)’s when processing each tuple.
Putting It Together. Algorithm 1 summarizes the procedure of computing \( \rho_s \) and \( S_n^2 \) for a single operator \( O \). It is straightforward to incorporate it into the previous framework where the selectivities of the operators are refined in a bottom-up fashion (Appendix B). We discuss some implementation details in the following.

Algorithm 1: Computation of \( \rho_s \) and \( S_n^2 \)

Input: \( O \), an operator; \( R^* = \{ R_1^*, \ldots, R_K^* \} \), the sample tables; \( \text{Agg} \), if some \( O' \in \text{Desc}(O) \) is an aggregate

Output: \( \rho_s \), estimated selectivity of \( O \); \( S_n^2 \), sample variance

1. \( R^* \leftarrow \text{RunOperator}(O, R^*) \);
2. if \( \text{Agg} \) then
   3. \( \rho_s \leftarrow \text{CardinalityByOptimizer}(O); \)
   4. \( \rho_s \leftarrow \frac{|R|^k}{|R^*|}; \)
   5. \( S_n^2 \leftarrow 0; \)
3. else if \( O \) is a scan then
   4. \( \rho_s \leftarrow \frac{|R|^k}{|R^*|}; \)
   5. \( S_n^2 \leftarrow \rho_s(1 - \rho_s); \)
4. else if \( O \) is a join then
   5. \( \rho_s \leftarrow \frac{|R|^k}{|R^*|}; \)
   6. foreach \( i = 1, \ldots, \max(K) \) do
      7. \( Q_{k,i}, n \leftarrow \text{Join}(Q_{k,i}, n+1), \text{for } 1 \leq k \leq K; \)
   8. end
   9. \( S_n^2 \leftarrow \frac{\sum_{k=1}^K (\frac{1}{n+1} \sum_{j=1}^n (Q_{k,j,n}/n^{K-1} - \rho_s)^2)}{n}; \)
10. end
11. else
   12. \( \rho_s \leftarrow \hat{\mu}_i, S_n^2 \leftarrow \hat{\sigma}_i^2; // \text{Let } X_i \sim N(\hat{\mu}_i, \hat{\sigma}_i^2). \)
13. end
14. return \( \rho_s \) and \( S_n^2 \).

First, the selectivity estimator cannot work for operators such as aggregates. Our current strategy is to use the original cardinality estimates from the optimizer to compute \( \rho_s \), and we simply set \( S_n^2 \) to be 0 for these operators (lines 3 to 5). This may cause inaccuracy in the prediction as well as our estimate of its uncertainty, if the optimizer does a poor job in estimating the cardinalities. However, we find that it works reasonably well in our experiments. Nonetheless, we are working to incorporate sampling-based estimators for aggregates (e.g., the GEE estimator [11]) into our current framework.

Second, to compute the \( Q_{k,i}, n \)'s, we maintain a hash map \( H_k \) for each \( k \) with \( i_k \)'s the keys and \( Q_{k,i_k}, n \)'s the values. The size of \( H_k \) is upper bounded by \( |R^*_k| \) and usually is much smaller.

Third, for simplicity of exposition, in Algorithm 1 we first compute the whole \( R^* \) and then scan it. In practice we actually do not need to do this. Typical join operators, such as \( \text{merge join} \), \( \text{hash join} \), and \( \text{nested-loop join} \), usually compute join results on the fly. Once a join tuple is computed, we can immediately postprocess it by increasing the corresponding \( Q_{k,i_k}, n \)'s. Therefore, we can avoid the additional memory overhead of caching intermediate join results, which might be large even if the sample tables are small.

4. COST FUNCTIONS

By Equation (2), to infer the distribution of \( t_q \) for a query \( q \), we also need to know the explicit form of \( g \). According to Equation (1), \( g \) relies on the cost functions of operators that map the selectivities to the \( n \)'s. As mentioned in the introduction, we use logical cost functions in our work. While different DBMS may differ in their implementations of a particular operator, e.g., nested-loop join, they follow the same execution logic and therefore have the same logical cost function. In the following, we first present a detailed study of representative cost functions. We then formulate the computation of cost functions as an optimization problem that seeks the best fit for the unknown coefficients, and we use standard quadratic programming techniques to solve this problem.

4.1 Types of Functions

We consider the following types of cost functions in this paper:

- \( f = a_0 \): The cost function is a constant. For instance, since a sequential scan has no random disk reads, \( n_r = 0 \).
- \( f = a_0 M + a_1 \): The cost function is linear with respect to the output cardinality. For example, the number of random reads of an index-based table scan falls into this category, which is proportional to the number of qualified tuples that pass the selection predicate.
- \( f = a_0 N_i + a_1 \): The cost function is linear with respect to the input cardinality. This happens for unary operators that process each input tuple once. For example, \( \text{materialization} \) is such an operator that creates a buffer to cache the intermediate results.
- \( f = a_0 N_i^2 + a_1 N_i + a_2 \): The cost function is nonlinear with respect to the input cardinality. For instance, the number of CPU operations (i.e., \( c_i \)) performed by a \( \text{sort} \) operator is proportional to \( N_i \log(N_i) \). While different nonlinear unary operators may have specific cost functions, we choose to only use quadratic polynomials based on the following observations:
  - It is quite general to approximate the nonlinear cost functions used by current relational operators. First, as long as a function is smooth (i.e., it has continuous derivatives up to some desired order), it can be approximated by using the well-known Taylor series, which is basically a polynomial of the input variable. Second, for efficiency reasons, the overhead of an operator usually does not go beyond quadratic of its input cardinality — we are not aware of any operator implementation whose time complexity is \( \omega(N^2) \). Similar observations have been made in [16].
  - Compared with functions such as logarithmic ones, polynomials are mathematically much easier to manipulate. Since we need to further infer the distribution of the predicted query execution time based on the cost functions, this greatly simplifies the derivations.
- \( f = a_0 N_i + a_1 N_r + a_2 \): This cost function is linear with respect to the input cardinalities when the operator is binary. An interesting observation here is that the cost functions in the case of binary operators are not necessarily nonlinear. For example, the number of I/O’s involved in a hash join is only proportional to the number of input tuples.
- \( f = a_0 N_i N_r + a_1 N_i + a_2 N_r + a_3 \): This cost function here also involves the product of the left and right input cardinalities of a binary operator. This happens typically in a nested-loop join, which iterates over the inner (i.e., the right) input table multiple times with respect to the number of rows in the outer (i.e., the left) input table.

It is straightforward to translate these cost functions in terms of selectivities. Specifically, we have \( N_i = |R_i|/X_i \), \( N_r = |R_r|/X_r \), and \( M = |R|/|X| \). The above six cost functions can be rewritten as
(C1') \( f = b_0, \) where \( b_0 = a_0. \)

(C2') \( f = b_0X + b_1, \) where \( b_0 = a_0|R| \) and \( b_1 = a_1. \)

(C3') \( f = b_0X_1 + b_1, \) where \( b_0 = a_0|R_i| \) and \( b_1 = a_1. \)

(C4') \( f = b_0X^2 + b_1X + b_2, \) where \( b_0 = a_0|R_i|^2 \), \( b_1 = a_1|R_i| \), and \( b_2 = a_2. \)

(C5') \( f = b_0X_1 + b_1X_2 + b_2, \) where \( b_0 = a_0|R_i| \), \( b_1 = a_1|R_i| \), and \( b_2 = a_2. \)

(C6') \( f = b_0X_1X_2 + b_1X_1 + b_2X_2 + b_3, \) where \( b_0 = a_0|R_i|^2 \), \( b_1 = a_1|R_i| \), \( b_2 = a_2|R_i| \), and \( b_3 = a_3. \)

4.2 Computation of Cost Functions

To compute the cost functions, we use an approach that is similar to the one proposed in [16]. Regarding the types of cost functions we considered, the only unknowns given the selectivity estimates are the coefficients in the functions (i.e., the \( b_i \)'s). Moreover, notice that \( f \) is a linear function of the \( b_i \)'s once the selectivities are given. We can then collect a number of \( f \) values by feeding in the cost model with different \( X \)'s and find the best fit for the \( b_i \)'s.

As an example, consider (C4'). Suppose that we invoke the cost model \( m \) times and obtain \( m \) points:

\[
\{(X_{11}, f_1), \ldots, (X_{1m}, f_m)\}.
\]

Let \( y = (f_1, \ldots, f_m) \), \( b = (b_0, b_1, b_2) \), and

\[
A = \begin{pmatrix}
X_{11} & X_{11} & 1 \\
X_{1l} & X_{1l} & 1 \\
\vdots & \vdots & \vdots \\
X_{ln} & X_{ln} & 1 
\end{pmatrix}.
\]

The optimization problem we are concerned with is:

\[
\begin{align*}
\text{minimize} & \quad \|Ab - y\| \\
\text{subject to} & \quad b_i \geq 0, \quad i = 0, 1.
\end{align*}
\]

Note that we require \( b_0 \) and \( b_1 \) to be nonnegative since they have the natural semantics in the cost functions as the amount of work with respect to the corresponding terms. For example, \( b_1X_1 = a_1N_i \) is the amount of work that is proportional to the input cardinality. To solve this quadratic programming programming, we use the \texttt{qpsolve} function of Scilab [42]. Other equivalent solvers could also be used.

The remaining problem is how to pick these \( (X, f) \)'s. In theory, one could arbitrarily pick the \( X \)'s from \([0, 1]\) to obtain the corresponding \( f \)'s as long as we have more points than unknowns. Although more points usually mean we can have better fittings, in practice we cannot afford too many points due to the efficiency requirements when making the prediction. On the other hand, given that the \( X \)'s here follow normal distributions and the variances are usually small when the sample size is large, the likely selectivity estimates are usually concentrated in a much shorter interval than \([0, 1]\). Intuitively, we should take more points within this interval, for we can then have a more accurate view of the shape of the cost function restricted to this interval. Therefore, in our current implementation, we adopt the following strategy.

Let \( X \sim \mathcal{N}(\mu, \sigma^2). \) Consider the interval \( I = [\mu - 3\sigma, \mu + 3\sigma]. \) It is well known that \( Pr(X \in I) \approx 0.997, \) which means the probability that \( X \) falls out of \( I \) is less than 0.3%. We then proceed by partitioning \( I \) into \( W \) subintervals of equal width, and pick the \( W + 1 \) boundary \( X \)'s to invoke the cost model. Generalizing this idea to binary cost functions is straightforward. Suppose \( X_1 \sim \mathcal{N}(\mu_1, \sigma^2_1) \) and \( X_2 \sim \mathcal{N}(\mu_2, \sigma^2_2). \) Let \( I_1 = [\mu_1 - 3\sigma_1, \mu_1 + 3\sigma_1] \) and \( I_2 = [\mu_2 - 3\sigma_2, \mu_2 + 3\sigma_2]. \) We then partition \( I_1 \times I_2 \) into a \( W \times W \) grid and obtain \((W + 1) \times (W + 1)\) points \((X_1, X_2)\) to invoke the cost model.

5. DISTRIBUTION OF RUNNING TIMES

We have discussed how to estimate the distributions of input parameters (i.e., the \( c_i \)'s and the \( X \)'s) and how to estimate the cost functions of each operator. In this section, we discuss how to combine these two to further infer the distribution of \( t_q \) for a query \( q. \)

Since \( t_q = g(c, X), \) the distribution of \( t_q \) relies on the joint distribution of \((c, X). \)\(^3\) We therefore first present a detailed analysis of the correlations between the \( c_i \)'s and the \( X \)'s. Based on that, we then show that the distribution of \( t_q \) is asymptotically normal and thus reduce the problem to estimating the two parameters of normal distributions, i.e., the mean and variance of \( t_q. \) We further address the nontrivial problem of computing \( \text{Var}[t_q] \) due to correlations between selectivity estimates.

5.1 Correlations of Input Variables

In our current setting, it is reasonable to assume that the \( c_i \)'s and the \( X \)'s are independent. In the following, we analyze the correlations within the \( c_i \)'s and the \( X \)'s.

5.1.1 Correlations Between Cost Units

Since the randomness within the \( c_i \)'s comes from the variations in hardware execution speeds, by using our current framework we have no way to observe the true values of the \( c_i \)'s and thus it is impossible to obtain the exact joint distribution of the \( c_i \)'s. Nonetheless, it might be reasonable to assume the independence of the \( c_i \)'s. First, since the CPU and I/O cost units measure the speeds of different hardware devices, their values do not depend on each other. Second, within each group (i.e., CPU or I/O cost units), we used independent calibration queries for each individual cost unit.

Assumption 1. The \( c_i \)'s are independent of each other.

We further note here that the independence of the \( c_i \)'s depends on the cost model as well as the hardware configurations. For instance, if certain devices are connected via the same infrastructure (e.g., a bus), then they might influence each other’s communication patterns. Our current framework for calibrating the \( c_i \)'s cannot capture the correlations of the \( c_i \)'s. However, perhaps low-level tools for monitoring hardware execution status could be used for this purpose. We leave it as interesting future work to investigate such possibilities and study the effectiveness of incorporating correlation information of the \( c_i \)'s into our current framework.

5.1.2 Correlations Between Selectivity Estimates

The \( X \)'s are clearly not independent, because the same samples are used to estimate the selectivities of different operators. We next study the correlations between the \( X \)'s in detail.

Let \( O \) and \( O' \) be two operators, and \( R \) and \( R' \) be the corresponding leaf tables. Consider the two corresponding selectivity estimates \( \rho_n \) and \( \rho'_n \) as defined by Equation (4). Since the samples from each table are drawn independently, we first have:

Lemma 1. If \( R \cap R' = \emptyset, \) then \( \rho_n \perp \rho'_n. \)

For binary operators, it follows from Lemma 1 immediately that:

Lemma 2. Let \( O \) be binary. If \( R \cap R' = \emptyset, \) then \( X_1 \perp X_r. \)

\(^3\)Note that the distributions of the \( c_i \)'s and \( X \)'s that we obtained in Section 3 are marginal rather than joint.

\(^4\)We use \( Y \perp Z \) to denote that \( Y \) and \( Z \) are independent.
That is, $X_t$ and $X_r$ will only be correlated if $R_t$ and $R_r$ share common relations. However, in practice, we can maintain more than one sample table for each relation. When the database is large, this is affordable since the number of samples is very small compared to the database size [48]. Since the samples from each relation are drawn independently, $X_t$ and $X_r$ are still independent if we use a different sample table for each appearance of a shared relation. We thus assume $X_t \perp X_r$ in the rest of the paper.

More generally, $X$ and $X'$ are independent as long as neither $O \in \text{Desc}(O')$ nor $O' \in \text{Desc}(O)$. However, the above discussion cannot be applied if $O \in \text{Desc}(O')$ (or vice versa). This is because we pass the join results from downstream joins to upstream joins when estimating the selectivities (recall Example 4). So $\mathcal{R}$ and $\mathcal{R}'$ are not naturally disjoint. In fact, $\mathcal{R} \subseteq \mathcal{R}'$. To make $\rho_n$ and $\rho_n'$ independent, we need to replace each of the sample tables used in computing $\rho_n'$ with another sample table from the same relation, which basically is the same as run the query plan again on a different sample table for each appearance of a shared relation. We therefore summarize this observation as follows:

**Lemma 3.** Given that multiple sample tables of the same relation can be used, $\rho_n$ and $\rho_n'$ are correlated if and only if either $O \in \text{Desc}(O')$ or vice versa.

### 5.2 Asymptotic Distributions

Now for specificity suppose that the query plan of $q$ contains $m$ operators $O_1, \ldots, O_m$. Since $\mathcal{t}_q$ is the sum of the predicted execution time spent on each operator, it can be expressed as $\mathcal{t}_q = \sum_{k=1}^{m} \mathcal{t}_k$, where $\mathcal{t}_k$ is the predicted execution time of $O_k$ and is itself a random variable.

We next show that $\mathcal{t}_k$ is asymptotically normal, and then by using very similar arguments, we can show that $\mathcal{t}_q$ is asymptotically normal as well. Since $\mathcal{t}_k$ can be further expressed in terms of Equation (1), to learn its distribution we need to know the distributions of cost functions that map the selectivities to the $n$’s. We therefore start by discussing the distributions of the typical cost functions as presented in Section 4.1.

#### 5.2.1 Asymptotic Distributions of Cost Functions

In the following discussion, we assume that $X \sim \mathcal{N}(\mu, \sigma^2)$, $X_t \sim \mathcal{N}(\mu_t, \sigma^2_t)$, and $X_r \sim \mathcal{N}(\mu_r, \sigma^2_r)$. The distributions of the six types of cost functions previously discussed are as follows:

- (C1’) $f = b_0; f \sim \mathcal{N}(b_0, 0)$.
- (C2’) $f = b_0X + b_1; f \sim \mathcal{N}(b_0\mu + b_1, b_0^2\sigma^2)$.
- (C3’) $f = b_0X_t + b_1; f \sim \mathcal{N}(b_0\mu_t + b_1, b_0^2\sigma^2_t)$.
- (C4’) $f = b_0X_t^2 + b_1X_t + b_2; \text{ In this case } \Pr(f) \text{ is not normal.}$

Although it is possible to derive the exact distribution of $f$ based on the distribution of $X_t$, the derivation would be very messy. Instead, we consider $f^N \sim \mathcal{N}(\text{E}[f], \text{Var}[f])$ and use this to approximate $\Pr(f)$. We present the formula of $\text{Var}[f]$ in Lemma 4 (proof in Appendix A.2). Obviously, $f^N$ and $f$ have the same expected value and variance. Moreover, we can actually show that $f^N$ and $f$ (and therefore their corresponding distributions) are very close to each other when the number of samples is large (see Theorem 1 below; the proof is in Appendix A.3).

- (C5’) $f = b_0X_t + b_1X_r + b_2; \text{ Since } X_t \perp X_r$, by Lemma 2, $f \sim \mathcal{N}(b_0\mu_t + b_1\mu_r + b_2, b_0^2\sigma^2_t + b_1^2\sigma^2_r)$.

\begin{align*}
(C6') f = b_0X_tX_r + b_1X_t + b_2X_r + b_3; \text{ Again, } \Pr(f) \text{ is not normal. Since } X_t \perp X_r, X_tX_r \text{ follows the so called normal product distribution [8], whose exact form is again complicated. We thus use the same strategy as in } (C4') \text{ (see Appendix A.4).}
\end{align*}

**Lemma 4.** If $X_t \sim \mathcal{N}(\mu_t, \sigma^2_t)$ and $f = b_0X_t^2 + b_1X_t + b_2$, then

\[
\text{Var}[f] = \sigma^2\left((b_1 + 2b_0\mu_1)^2 + 2b_0^2\sigma^2_t\right).
\]

**Theorem 1.** Suppose that $X_t \sim \mathcal{N}(\mu_t, \sigma^2_t)$ and $f = b_0X_t^2 + b_1X_t + b_2$. Let $f^N \sim \mathcal{N}(\text{E}[f], \text{Var}[f])$, where $\text{Var}[f]$ is shown in Lemma 4. Then $f^N \xrightarrow{p} f$.\(^5\)

#### 5.2.2 Asymptotic Distribution of $t_k$

Based on the previous analysis, the cost functions (or equivalently, the $n$’s in Equation (1)) are asymptotically normal. Since the $c$’s are normal and independent of the $X$’s (and hence the $n$’s as well), by Equation (1) again $t_k$ is asymptotically the sum of products of two independent normal random variables. Specifically, let $C = \{c_0, c_1, \ldots, c_n\}$, and for $c \in C$, let $f_c$ be the cost function indexed by $c$. Defining $t_{kc} = f^N_c$, we have

\[
t_k \approx \sum_{c \in C} f^N_{kc} = \sum_{c \in C} f^N_{kc}c,
\]

Again, each $t_{kc}$ is not normal. But we can apply techniques similar to that in Theorem 1 here by using the normal random variable $t^N_{kc} \sim \mathcal{N}(\text{E}[f^N_{kc}], \text{Var}[f^N_{kc}])$ as an approximation of $t_{kc}$. Defining $Z = \text{E}[f^N_{kc}]$, we have

**Theorem 2.** $t_{kc} \xrightarrow{d} Z$, and $t^N_{kc} \xrightarrow{d} Z$.\(^6\)

Theorem 2 (proof in Appendix A.5) implies that $t_{kc}$ and $t^N_{kc}$ tend to follow the same distribution as the sample size grows. Since $c$ is normal, $Z$ is normal as well. Furthermore, the independence of the $c$’s also implies the independence of the $Z$’s. So $t_k$ is asymptotically the sum of the independent normal random variables $t^N_{kc}$. Hence $t_k$ is itself asymptotically normal with large sample size.

#### 5.2.3 Asymptotic Distribution of $t_q$

Finally, let us consider the distribution of $t_q$. Since $t_q$ is merely the sum of the $t_k$’s, we have exactly the same situation as when we analyze each $t_k$. Specifically, we can express $t_q$ as

\[
t_q = \sum_{k=1}^{m} t_k \approx \sum_{c \in C} g_c c,
\]

where $g_c = \sum_{k=1}^{m} f^N_{kc}$ is the sum of the cost functions of the operators with respect to the particular $c$. However, since the $f^N_{kc}$’s are not independent, $g_c$ is not normal. We can again use the normal random variable

\[
g^N_c \sim \mathcal{N}(\text{E}[g_c], \text{Var}[g_c])
\]

as an approximation of $g_c$. We show $g^N_c \xrightarrow{p} g_c$ in Appendix A.6. With exactly the same argument used in Section 5.2.2 we can then see that $t_q$ is approximately normal when the sample size is large.

\(^5\) $f^N \xrightarrow{p} f$ means $f^N$ converges in probability to $f$.\(^6\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.\(^7\) $f^N \xrightarrow{d} Z$ means $f^N$ converges in distribution to $Z$.\(^8\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.\(^9\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.\(^10\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.\(^11\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.\(^12\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.\(^13\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.\(^14\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.\(^15\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.\(^16\) $t^N \xrightarrow{d} Z$ means $t^N$ converges in distribution to $Z$.
5.2.4 Discussion

The analysis that \( t_q \) is asymptotically normal relies on three facts: (1) the selectivity estimates are unbiased and strongly consistent; (2) the cost model is additive; and (3) the cost units are independently normally distributed. While the first fact is a property of the sampling-based selectivity estimator and thus always holds, the latter two are specific merits of the cost model of PostgreSQL, though we believe that cost models of other database systems share more or less similar features. (As far as we know, MySQL [50], IBM DB2 [1], Oracle [2], and Microsoft SQL Server [3] use similar cost models.) Therefore, we need new techniques when either (2) or (3) does not hold. For instance, if the cost model is still additive and the \( c \)'s are independent but cannot be modeled as normal variables, then by the analysis in Section 5.2.3 we can still see that \( t_q \) is asymptotically a linear combination of the \( c \)'s and thus the distribution of \( t_q \) can be expressed in terms of the convolution of the distributions of the \( c \)'s. We may then find this distribution by using generating functions or characteristic functions [41]. We leave the investigation of other types of cost models as future work.

5.3 Computing Distribution Parameters

As discussed, we can approximate the distribution of \( t_q \) with a normal distribution \( N(E[t_q], \text{Var}[t_q]) \). We are then left with the problem of estimating the two parameters \( E[t_q] \) and \( \text{Var}[t_q] \). While \( E[t_q] \) is trivial to compute — it is merely the original prediction from our predictor, estimating \( \text{Var}[t_q] \) is a challenging problem due to the correlations presented in selectivity estimates.

In more detail, so far we have observed the additive nature of \( t_q \), and \( \text{Var}[t_q] \) is normal; the non-central moments of \( t_q \) are of type \((C5')\). Based on Lemma 2, \( \text{Cov}(X_1, X_2) = 0 \) and \( \text{Cov}(X_1, X_3) = 0 \). Also, based on Lemma 3, \( \text{Cov}(X_1, X_3) = 0 \) and \( \text{Cov}(X_2, X_3) = 0 \). However, we are not able to compute \( \text{Cov}(X_1, X_4) \) and \( \text{Cov}(X_2, X_4) \). Instead, we provide upper bounds for them.

5.3.2 Upper Bounds of Covariances

Based on the fact that the covariance between two random variables is bounded by the geometric mean of their variances [41], we can establish an upper bound for \( Z \) and \( Z' \) in the previous section:

\[
|\text{Cov}(Z, Z')| \leq \sqrt{\text{Var}[Z] \text{Var}[Z']}
\]

Note that the variances are directly computable based on the independence assumptions (Lemma 2 and 3).

By analyzing the correlation of the samples used in selectivity estimation, we can develop tighter bounds (details in Appendix A.7). The key observation here is that the correlations are caused by the samples from the shared relations. Consider two operators \( O \) and \( O' \) such that \( O \in \text{Desc}(O') \).

5. EXPERIMENTAL EVALUATION

We present experimental evaluation results in this section. There are two key respects that could impact the utility of a predictor: its prediction accuracy and runtime overhead. However, for the particular purpose of this paper, we do not care much about the
absolute accuracy of the prediction. Rather, we care if the distribution of likely running times reflects the uncertainty in the prediction. Specifically, we measure if the estimated prediction errors are correlated with the actual errors. To measure the accuracy of the predicted distribution, we also compare the estimated likelihoods that the actual running times will fall into certain confidence intervals with the actual likelihoods. On the other hand, we measure the runtime overhead of the sampling-based approach in terms of its relative overhead with respect to the original query running time without sampling. We start by presenting the experimental settings and the benchmark queries we used.

6.1 Experimental Settings
We implemented our proposed framework in PostgreSQL 9.0.4. We ran PostgreSQL under Linux 3.2.0-26, and we evaluated our approaches with both the TPC-H 1GB and 10 GB databases. Since the original TPC-H database generator uses uniform distributions, to test the effectiveness of the approach under different data distributions, we used a skewed TPC-H database generator [4]. It produces TPC-H databases with a Zipf distribution and uses a parameter $z$ to control the degree of skewness. $z = 0$ generates a uniform distribution, and the data becomes more skewed as $z$ increases. We created skewed databases using $z = 1$. All experiments were conducted on two machines with the following configurations:

- PC1: Dual Intel 1.86 GHz CPU and 4GB of memory;
- PC2: 8-core 2.40GHz Intel CPU and 16GB of memory.

6.2 Benchmark Queries
We created three benchmarks MICRO, SELJOIN, and TPCH:

- MICRO consists of pure selection queries (i.e., scans) and two-way join queries. It is a micro-benchmark with the purpose of exploring the strength and weakness of our proposed approach at different points in the selectivity space. We generated the queries with the similar ideas used in the Picasso database query optimizer visualizer [40]. Since the queries have either one (for scans) or two predicates (for joins), the selectivity space is either one or two dimensional. We generated SQL queries that were evenly across the selectivity space, by using the statistics information (e.g., histograms) stored in the database catalogs to compute the selectivities.

- SELJOIN consists of selection-join queries with multi-way joins. We generated the queries in the following way. We analyzed each TPC-H query template, and identified the “maximal” sub-query without aggregates. We then randomly generated instance queries from these reduced templates. The purpose is to test the particular type of queries to which our proposed approach is tailored — the selection-join queries.

- TPCH consists of instance queries from the TPC-H templates. These queries also contain aggregates, and our current strategy is simply ignoring the uncertainty there (recall Section 3.2.2). The purpose of this benchmark is to see how this simple work-around works in practice. We used 14 TPC-H templates: 1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 18, and 19. We did not use the other templates since their query plans contain structures that cannot be handled by our current framework (e.g., sub-query plans or views).

We ran each query 5 times and took the average as the actual running time of a query. We cleared both the filesystem cache and the database buffer pool between each run of each query.

6.3 Usefulness of Predicted Distributions
Since our goal is to quantify the uncertainty in the prediction and our output is a distribution of likely running times, the question is then how we can know that we have something useful. A reasonable metric here could be the correlation between the standard deviation of the predicted (normal) distribution and the actual prediction error. Intuitively, the standard deviation indicates the confidence of the prediction. A larger standard deviation indicates lower confidence and hence larger potential prediction error. With this in mind, if our approach is effective, we would expect to see positive correlations between the standard deviations and the real prediction errors when a large number of queries are tested.

A common metric used to measure the correlation between two random variables is the Pearson correlation coefficient $r_p$. Suppose that we have $n$ queries $q_1, ..., q_n$. Let $\sigma_i$ be the standard deviation of the distribution predicted for $q_i$, $\mu_i$ and $t_i$ be the predicted (mean) and actual running time of $q_i$, and $e_i = |\mu_i - t_i|$ be the prediction error. $r_p$ is then defined as

$$r_p = \frac{\sum_{i=1}^{n}(\sigma_i - \bar{\sigma})(e_i - \bar{e})}{\sqrt{\sum_{i=1}^{n}(\sigma_i - \bar{\sigma})^2} \sqrt{\sum_{i=1}^{n}(e_i - \bar{e})^2}},$$

where $\bar{\sigma} = \frac{1}{n} \sum_{i=1}^{n} \sigma_i$ and $\bar{e} = \frac{1}{n} \sum_{i=1}^{n} e_i$.

Basically, $r_p$ measures the linear correlation between the $\sigma$’s and the $e$’s. The closer $r_p$ is to 1, the better the correlation is. However, there are two issues here. First, even if the $\sigma$’s and the $e$’s are positively correlated, the correlation may not be linear. Second, $r_p$ is not robust and its value can be misleading if outliers are present [17]. Therefore, we also measure the correlations by using another well known metric called the Spearman’s rank correlation coefficient $r_s$ [38]. The formula of $r_s$ is the same as Equation (7) except that for the $\sigma$’s and the $e$’s are replaced with their ranks in the ascending order of the values. For instance, given three $\sigma$’s $\sigma_1 = 4$, $\sigma_2 = 7$, and $\sigma_3 = 5$, their ranks are 1, 3, and 2 respectively. Intuitively, $r_s$ indicates the linear correlation between the ranks of the values, which is more robust than $r_p$ since the mapping from the values to their ranks can be thought of as some normalization procedure that reduces the impact of outliers. In fact, $r_s$ assesses how well the correlation can be characterized by using a monotonic function and $r_s = 1$ means the correlation is perfect.

In Figure 2, we report the $r_s$’s (and the corresponding $r_p$’s) for the benchmark queries over different hardware and database settings (see Table 4 of Appendix C.1 for the complete results). Here, sampling ratio (SR) stands for the fraction of the sample size with respect to the database size. For instance, SR = 0.01 means that 1% of the data is taken as samples. We have several observations.

First, for most of the cases we tested, both $r_s$ and $r_p$ are above 0.7 (in fact above 0.9), which implies strong positive (linear) correlation between the standard deviations of the predicted distributions and the actual prediction errors. Second, in [48] we showed that as expected, prediction errors can be reduced by using larger number of samples. Interestingly, it is not necessarily the case that more samples improves the correlation between the predicted and actual errors. This is because taking more samples simultaneously reduces the errors in selectivity estimates and the uncertainty in the predicted running times. So it might improve the estimate but not the correlation with the true errors. Third, reporting both $r_s$ and $r_p$ is necessary since they sometimes disagree with each other. For instance, consider the following two cases in Figure 2(a) and 2(b):

1. On PC2, the MICRO queries over the uniform TPC-H 1GB database give $r_s = 0.9400$ but $r_p = 0.5691$ when SR = 0.01;

2. It is generally believed that two variables are strongly correlated if their correlation coefficient is above 0.7.
(2) On PC1, the SELJOIN queries over the uniform TPC-H 1GB database give $r_s = 0.6958$ but $r_p = 0.8414$ when $SR = 0.05$.

In Figure 3(a) and 3(c), we present the scatter plots of these two cases. Figure 3(b) further shows the scatter plot after the rightmost point is removed from Figure 3(a). We find that now $r_s = 0.9386$ but $r_p = 0.8868$. So $r_p$ is much more sensitive to outliers in the population. Since in our context there is no good criterion to remove outliers, $r_s$ is thus more trustworthy. On the other hand, although the $r_p$ of (2) is better than that of (1), by comparing Figure 3(b) with Figure 3(c) we would instead conclude that the correlation of (2) seems to be worse. This is again implied by the worse $r_s$ of (2). More results and analysis can be found in Appendix C.5.

Nonetheless, the strong positive correlations between the estimated standard deviations and the actual prediction errors may not be sufficient to conclude that the distributions of likely running times are useful. For our purpose of informing the consumer of the running time estimates of the potential prediction errors, it might be worth to further consider what information regarding the errors the predicted distributions really carry. Formally, consider the $n$ queries $q_1, ..., q_n$ as before. Since the estimated distributions are normal, with the previous notation the distribution for the likely queries $E_i$ is $T_i \sim N(\mu_i, \sigma_i^2)$. As a result, assuming $\alpha > 0$, without loss of generality the estimated prediction error $E_i = |T_i - \mu_i|$. We then calculate

$$Pr(E_i \leq \alpha \sigma_i) = Pr(-\alpha \leq \frac{T_i - \mu_i}{\sigma_i} \leq \alpha) = 2\Phi(\alpha) - 1,$$

where $\Phi$ is the cumulative distribution function of the standard normal distribution $N(0, 1)$. Therefore, if we define the statistic $E'_i = \frac{E_i}{\sigma_i} = \frac{|T_i - \mu_i|}{\sigma_i}$, then $Pr(E'_i \leq \alpha) = Pr(E_i \leq \alpha \sigma_i)$. Note that $Pr(E'_i \leq \alpha)$ is determined by $\alpha$ but not $i$. We thus simply use $Pr(\alpha)$ to denote $Pr(E'_i \leq \alpha)$. On the other hand, we can estimate the actual likelihood of $E'_i \leq \alpha$ by using

$$Pr_n(\alpha) = \frac{1}{n} \sum_{i=1}^{n} I(e'_i \leq \alpha), \text{ where } e'_i = \frac{e_i}{\sigma_i} = \frac{|T_i - \mu_i|}{\sigma_i}.$$

Here $I$ is the indicator function. To measure the proximity of $Pr_n(\alpha)$ and $Pr(\alpha)$, we define

$$D_n(\alpha) = |Pr_n(\alpha) - Pr(\alpha)|.$$

Clearly, a smaller $D_n(\alpha)$ means $Pr(\alpha)$ is closer to $Pr_n(\alpha)$, which implies better quality of the distributions. We further generated $\alpha$’s from the interval $(0, 6)$ which is sufficiently wide for normal distributions and computed the average of the $D_n(\alpha)$’s (denoted as $\bar{D}_n$). Figure 4 reports the results for the benchmark queries and PC1 over the uniform TPC-H 1GB databases (see Table 5 of Appendix C.2 for the complete results).

We observe that in most cases the $\bar{D}_n$’s are below 0.3 with the majority below 0.2, which suggests that the estimated $Pr(\alpha)$’s are reasonably close to the observed $Pr_n(\alpha)$’s. To shed some light on what is going on here, in Figure 5 we further plot the $Pr(\alpha)$ and $Pr_n(\alpha)$ for the (1) MICRO, (2) SELJOIN, and (3) TPCH queries over the uniform TPC-H 1GB database on PC1 when $SR = 0.05$, which give $\bar{D}_n = 0.2532, 0.1098$, and 0.0535 respectively. We can see that we overestimated the $Pr(\alpha)$’s for small $\alpha$’s. In other words, we underestimated the prediction errors by presenting smaller than actual variances in the distributions. Moreover, we find that our estimate is more significant for the MICRO queries (Figure 5(a)). One possible reason is that since these queries are really simple the predictor tends to be over-confident by underestimating the variances even more. When handling SELJOIN and TPCH queries, the confidence of the predictor drops and underestimate tends to be alleviated (Figure 5(b) and 5(c)).

6.3.1 More Discussion on Correlation

While using ordinal ranks instead of values can help in smoothing the data reducing the impact of outliers, it is still imperfect. The
best way of presenting correlations between two quantities might be a scatter plot (as shown in Figure 3). There are four possible cases: (1) $r_s$ is better than $r_p$; (2) $r_p$ is better than $r_s$; (3) $r_s$ and $r_p$ are both good; and (4) $r_p$ and $r_s$ are both not so good. We have presented scatter plots for (1) and (2) in Figure 3(a) and 3(c), respectively. To gain more insight, in Figure 6(a) and 6(b) we present two typical scatter plots for (3) and (4):

(3) On PC1, the TPCH queries over the skewed TPC-H 10GB database give $r_s = 0.9439$ and $r_p = 0.9887$ when SR = 0.05;

(4) On PC1, the TPCH queries over the uniform TPC-H 1GB database give $r_s = 0.7209$ and $r_p = 0.7571$ when SR = 0.01.

As we can see, when both $r_s$ and $r_p$ are good, the correlation is close to positive linear. On the other hand, the correlation is not so good when both $r_s$ and $r_p$ are not so good.

6.3.2 A Note on a Baseline Experiment

Shrewd readers might have wondered a different, intuitively simpler experiment: fix one query, generate many different samples, and make a prediction based on each sample; then test if and how well the distribution of the predicted estimates matches the distribution computed by using our proposed framework.

The question raised here is if the distribution of running times predicted by using different samples would match the one computed by our model. But note that “the distribution in the model” actually depends on samples, that is, the model will output a different distribution if it uses a different sample.

To put things in context, let us consider a query $q$ where we used two samples $S_1$ and $S_2$ to make predictions for its running time. Suppose that the two point estimates by using $S_1$ and $S_2$ are $\mu_1$ and $\mu_2$, respectively. We would then expect to see a picture as shown in Figure 7(a), where the likelihoods of $\mu_1$ and $\mu_2$ match “the” distribution $D$ computed by our model. However, our model would actually compute a distribution $D_1$ describing its uncertainty about $\mu_1$, and compute a different distribution $D_2$ describing its uncertainty about $\mu_2$. We illustrate this in Figure 7(b). Therefore, the expected $D$ is not unique. Rather, using different samples will result in different $D$’s.

Why should we derive different distributions if different samples are used? This is somehow not surprising. Different samples derived from the base tables may vary tremendously. As a result, the
uncertainties in the selectivity estimates based on different samples may differ and hence the uncertainties in the running time estimates may differ as well.

6.3.3 Comparison with Simplified Versions

Another interesting question is if we can simplify some steps in our framework. In the following we consider four alternatives (the complete version as well as three simplified versions):

(V1) **All**: the complete version of our proposed framework;
(V2) **No Var[c]**: ignore the uncertainties in the cost units by setting Var[c] = 0 for each c;
(V3) **No Var[X]**: ignore the uncertainties in the selectivity estimates by setting Var[X] = 0 for each X;
(V4) **No Cov**: ignore the covariances in the selectivity estimates.

We compared these four alternatives for the TPCH queries. Figure 8 presents typical results on uniformed databases in terms of the correlation coefficient $r_s$ (see Appendix C.3 for more results on skewed databases). We have several observations. First, ignoring uncertainties in the $c_i$’s is not a good idea. For all the cases we tested, this would lead to a drop of at least 0.25 (typically 0.4 to 0.5) in correlation. Second, the impact of ignoring uncertainties in the $X$’s depends on the sample size. Intuitively, as we increase the sample size, the uncertainties in the $X$’s diminish due to the strong consistency of the estimator. When the uncertainties are small enough, ignoring them is safe. As shown in Appendix C.5, a sampling ratio of 1% is already sufficient for accurate selectivity estimates for most of the queries we tested. To observe the impact of ignoring the uncertainties in the $X$’s, we therefore used even lower sampling ratios. As we can observe from Figure 8, typically the correlation can drop by 0.2 to 0.3 when the sampling ratios are below 1%, while it remains almost unaffected when 1% samples are taken. Third, while the impact of covariances in the $X$’s is often insignificant, sometimes ignoring the covariances causes problems. For instance, as shown in Figure 8(b), the correlations drop by 0.35 and 0.17 when the sampling ratios are 0.05% and 0.1%. Although we cannot directly compute the covariances, our theoretic study in Appendix A suggests that the upper bounds for the covariances become smaller as we increase the sample size. Nevertheless, in general we have no idea how large the sample size needs to be so that we can safely ignore the uncertainties in the selectivity estimates and their covariances. It depends on several factors such as the skewness of the data and the complexity of the queries in the workload. Finally, the complete version is the most robust and effective one among the four alternatives: $r_s$ is consistently above 0.7 (most of the time above 0.8) for all the cases we tested.

6.4 Runtime Overhead of Sampling

We also measured the relative overhead of running the queries over the sample tables compared with that of running them over the original tables. Figure 9 presents the results of the TPCH queries on PC1. Since the other results are very similar, the readers are referred to Appendix C.4 for the complete details. We observe that the relative overhead is comparable to that reported in [48]. For instance, for the TPC-H 10GB database, the relative overhead is around 0.04 to 0.06 when the sampling ratio is 0.05. Note that, here we computed the estimated selectivities as well as their variances by only increasing the relative overhead a little. Also note that, here we measured the relative overhead based on disk-resident samples. The relative overhead can be dramatically reduced by using the common practice of caching the samples in memory [39].

On the other hand, though the Central Limit Theorem guarantees that the selectivity estimates are Gaussian-distributed for sufficiently large samples, it does not say anything about how large the samples should be. In fact, there is no exact answer to this question. As a rule of thumb, statisticians have agreed that the sample size should be larger than or equal to 30 in general, and the larger the better [10].
6.5 Applications

We discuss some potential applications that could take advantage of the distributional information of query running times. The list of applications here is by no means exhaustive, and it is our hope that our study in this paper could stimulate further research in this direction and more applications could emerge in the future.

6.5.1 Query Optimization

Although significant progress has been made in the past several decades, query optimization remains challenging for many queries due to the difficulty in accurately estimating query running times. Rather than betting on the optimality of the plan generated based on (perhaps erroneous) point estimates for parameters such as selectivities and cost units, it makes sense to also consider the uncertainties of these parameters. In fact, there has been some theoretical work investigating optimization based on least expected cost (LEC) based upon distributions of the parameters of the cost model [15]. However, that work did not address the problem of how to obtain the distributions. It would be interesting to see the effectiveness of LEC plans by incorporating our techniques into query optimizers.

6.5.2 Query Progress Monitoring

State-of-the-art query progress indicators [13, 31, 33, 35] provide estimates of the percentage of the work that has been completed by a query at regular intervals during the query’s execution. However, it has been shown that in the worst case no progress indicator can outperform a naive indicator simply saying the progress is between 0% and 100% [12]. Hence, information about uncertainty in the estimate of progress is desirable. Our work provides a natural building block that could be used to develop an uncertainty-aware query progress indicator: the progress indicator could call our predictor to make a prediction for the remaining query running time as well as its uncertainty.

6.5.3 Database as a Service

The problem of predicting query running time is revitalized by the recent move towards providing database as a service (DaaS). Many important decision-making procedures, including admission control [43, 49], query scheduling [14], and system sizing [45], rely on estimation of query running time. Distributional information enables more robust decision procedures in contrast to point estimates. Recent work [14] has shown the benefits in query scheduling by leveraging distributional information. Similar ideas have also been raised in [49] for admission control. Again, these work did not address the fundamental issue of obtaining the distributions without running the queries. It would be interesting to see the effectiveness of our proposed techniques in these DaaS applications.

7. RELATED WORK

The problem of predicting query execution time has been extensively studied quite recently [6, 7, 19, 20, 47, 48]. Ganapathi et al. [20] first raised the question of predicting the actual running time of a query rather than a rough estimate of runtime overhead that is usually provided by most, if not all, query optimizers. They further proposed a predictive approach based on Kernel Canonical Correlation Analysis (KCCA). Follow-up approaches improved the prediction accuracy by using different machine learning models such as Support Vector Machines (SVM) [7] or Multiple Additive Regression-Trees (MART) [32]. Unlike these data-driven machine learning approaches that treated the underlying database system as a black box, we proposed a predictor based on calibrating the query optimizer’s cost models and showed that it could often outperform the machine learning based approaches in terms of prediction accuracy [48]. While most of this line of work focused on the single-query prediction problem, some of them have considered the more general prediction problem when multiple queries are concurrently running. Ahmad et al. addressed the problem by using Gaussian processes [6], while Duggan et al. adopted similar ideas but instead used multivariate linear regression [19]. Both of them, however, assumed static database workloads, in the sense that all queries running in the system should be known beforehand. To overcome this limitation, we proposed a conceptually different approach by extending our single-query predictor [47]. We first used query optimizer’s cost models to estimate the CPU and I/O requirements for each query, and then used a combination queueing model and buffer pool model to merge these quantities from concurrent queries to predict running times. Nonetheless, none of these work ever considered the problem of measuring the degree of uncertainty in the prediction. We have reused some techniques developed in [48] for computing the means of selectivities and cost units when viewed as random variables. Nonetheless, [48] focused on point estimates rather than distributional information, and hence these techniques were insufficient. We have substantially extended [48] by developing new techniques for computing variances (and hence distributions) of selectivity and cost-unit estimates (Section 3), cost functions (Section 4), and, based on that, distributions of likely running times (Section 5).

The idea of using samples to estimate selectivity goes back more than two decades ago (e.g., [9, 11, 24, 25, 28, 29, 34]). While we focused on estimators for selection and join queries [25], some estimators that estimate the number of distinct values might be further used to refine selectivity estimates of aggregate queries [11, 24]. However, not only do we need an estimate of selectivity, we need an estimated distribution as well. So far, we are not aware of any previous study towards this direction for aggregate queries. Regarding the problem of estimating selectivity distributions for selection and join queries, there are options other than the one used in this paper. For example, Babcock and Chaudhuri [9] proposed a framework to learn the posterior distributions of the selectivities based on join synopses [5]. Unfortunately, this solution is restricted to SPI expressions with foreign-key joins, due to the overhead of computing and maintaining join synopses over a large database.

The framework proposed in this paper also relies on accurate approximation of the cost models used by the optimizer. Du et al. [18] first proposed the idea of using logical cost functions in the context of heterogeneous database systems. Similar ideas were later on used in developing generic cost models for main memory based database systems [36] and identifying robust plans in the plan diagram generated by the optimizer [16]. Our idea of further using optimization techniques to find the best coefficients in the logical cost functions is motivated by the approach used in [16].
8. CONCLUSION

In this paper, we take a first step towards the problem of measuring the uncertainty within query execution time prediction. We quantify prediction uncertainty using the distribution of likely running times. Our experimental results show that the standard deviations of the distributions estimated by our proposed approaches are strongly correlated with the actual prediction errors.

The idea of leveraging cost models to quantify prediction uncertainty need not be restricted to single standalone queries. As shown in [47], Equation (2) can also be used to provide point estimates for multiple concurrently-running queries. The key observation is that the selectivities of the operators in a query are independent of whether or not it is running with other queries. Hence it is promising to consider applying the techniques proposed in this paper to multi-query workloads by viewing the interference between queries as changing the distribution of the e’s. We regard this as a compelling area for future work.

9. REFERENCES

APPENDIX

A. THEORETIC RESULTS

This section presents the proofs of the lemmas and theorems as well as other related results mentioned in the paper.

A.1 Variance of The Estimator \( \rho_n \)

The variance of the selectivity estimator \( \rho_n \), unfortunately, is nontrivial when writing it mathematically:

**THEOREM 3.** The variance of \( \rho_n \) is [25]:

\[
\text{Var}[\rho_n] = \sum_{r=1}^{K} \frac{(n-1)^{K-r}}{n^K} \times \sum_{S \in S_r} \left( \frac{1}{|\Lambda(S)|} \sum_{1 \in \Lambda(S)} (\rho_S(1) - \rho)^2 \right).
\]

Here \( S = \{k_1, \ldots, k_r\} \subseteq \{1, 2, \ldots, K\} \) such that \( k_1 < k_2 < \ldots < k_r \), \( S_r \) is the collection of all subsets of \( \{1, 2, \ldots, K\} \) with size \( r \) (for \( 1 \leq r \leq K \)), and \( \Lambda(S) \) is defined to be

\[
\Lambda(S) = \{1, 2, \ldots, m_{k_1}\} \times \cdots \times \{1, 2, \ldots, m_{k_r}\}.
\]

Furthermore, for \( l = (l_1, \ldots, l_r) \in \Lambda(S) \), \( \rho_S(l) \) is the average selectivity over \( B(L_1, \ldots, L_K) \) such that \( L_{k_r} = l_j (1 \leq j \leq r) \).

For example, if \( K = 4 \), \( S = \{2, 3\} \) and \( l = (8, 9) \), then

\[
\rho_S(l) = (m_1 m_4)^{-1} \sum_{l_1 = 1}^{m_1} \sum_{l_4 = 1}^{m_4} \rho_B(L_1, 8, 9, L_4).
\]

We next prove Theorem 3. Roughly speaking, the idea of the proof is to first partition the samples into groups based on how many blocks they share, then compute the variance of each group, and finally sum them up. We start with the following standard result from probability theory:

**LEMMA 5.** Let \( X_1, \ldots, X_n \) be \( n \) random variables, then

\[
\text{Var}\left[ \sum_{i=1}^{n} X_i \right] = \sum_{i=1}^{n} \sum_{j=1}^{n} \text{Cov}(X_i, X_j),
\]

where \( \text{Cov}(X_i, X_j) \) is the covariance of \( X_i \) and \( X_j \):

\[
\text{Cov}(X_i, X_j) = \mathbb{E}[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])].
\]

Now define \( X \) to be the set of all sample blocks, namely,

\[
X = \{\rho_B(L_1, \ldots, L_K) | 1 \leq i_k \leq n, 1 \leq k \leq K\}.
\]

Based on Lemma 5 and Equation (4), we have

\[
\text{Var}[\rho_n] = \frac{1}{(nK)^2} \sum_{X \in X} \sum_{X' \in X} \text{Cov}(X, X').
\]

Consider any \( X = \rho_B(j_1, \ldots, j_K) \) and \( X' = \rho_B(j'_1, \ldots, j'_K) \) in the summands of Equation (9). If \( j_k \neq j'_k \) for \( 1 \leq k \leq K \), then \( X \) and \( X' \) are independent and \( \text{Cov}(X, X') = 0 \). Hence, only \( X \) and \( X' \) that share at least one common coordinate will contribute a non-zero summand to Equation (9). We thereby partition the pairs \((X, X')\) according to the number of coordinates they share. Specifically, for \( S = \{k_1, \ldots, k_r\} \subseteq \{1, 2, \ldots, K\} \), we denote \( X \sim_S X' \) if \( j_k = j'_k \) for \( 1 \leq m \leq r \). This gives us the following equivalent expression for \( \text{Var}[\rho_n] \):

\[
\text{Var}[\rho_n] = \frac{1}{(nK)^2} \sum_{r=1}^{K} \sum_{S \in S_r} \sum_{X \sim_S X'} \text{Cov}(X, X').
\]

**LEMMA 6.** For a fixed \( S \in S_r \), the number of pairs \((X, X')\) such that \( X \sim_S X' \) is \((n-1)^{K-r}n^K \). As a result, \( \text{Var}[\rho_n] \) can be further expressed as:

\[
\text{Var}[\rho_n] = \sum_{r=1}^{K} \frac{(n-1)^{K-r}}{n^K} \times \sum_{S \in S_r} \text{Cov}(X, X').
\]

We next assume that \( \rho \) is a constant such that \( \text{Var}(\rho) = 0 \), which means that any sub-block \( S \) of \( X \) shares the same value. Our next goal is to give an expression for \( \text{Cov}(X, X') \) when \( X \sim_S X' \), as shown in Lemma 7. Equation (8) in Theorem 3 then follows by combining Lemma 6 and 7.

**LEMMA 7.** If \( X \sim_S X' \), then

\[
\text{Cov}(X, X') = \frac{1}{|\Lambda(S)|} \sum_{1 \in \Lambda(S)} (\rho_S(1) - \rho)^2.
\]

**PROOF.** We have

\[
\text{Cov}(X, X') = \mathbb{E}_{X \sim_S X'}[(X - \mathbb{E}[X])(X' - \mathbb{E}[X'])].
\]

Since \( \mathbb{E}[X] = \mathbb{E}[X'] = \rho \), it follows that

\[
\text{Cov}(X, X') = \mathbb{E}_{X \sim_S X'}[(X - \rho)(X' - \rho)].
\]

We further denote \( X \sim_S X' \) for \( l = (l_1, \ldots, l_r) \), if \( X \sim_S X' \) and \( j_k = l_m \) for \( 1 \leq m \leq r \). Then we have

\[
\text{Cov}(X, X') = \frac{1}{|\Lambda(S)|} \sum_{1 \in \Lambda(S)} \mathbb{E}_{X \sim_S X'}[(X - \rho)(X' - \rho)].
\]

Now consider \( \mathbb{E}_{X \sim_S X'}[(X - \rho)(X' - \rho)] \). By definition, it is the average of the following quantities

\[
\gamma = (\rho_B(j_1, \ldots, j_K) - \rho)(\rho_B(j'_1, \ldots, j'_K) - \rho)
\]

by setting \( j_k = j'_k = l_m \) for \( 1 \leq m \leq r \). Let \( S^c = \{1, \ldots, K\} - S \) be the complement of \( S \). We then have

\[
\mathbb{E} = \mathbb{E}_{X \sim_S X'}[(X - \rho)(X' - \rho)] = \left( \frac{1}{|\Lambda(S^c)|} \right)^2 \sum_{\Lambda(S^c)} \gamma.
\]

After some rearrangement of the summands, we can have

\[
\mathbb{E} = \left( \frac{1}{|\Lambda(S)|} \right)^2 \sum_{\Lambda(S^c)} (\rho_{B(j_1, \ldots, j_K)} - \rho)^2
\]

\[
= \left( \frac{1}{|\Lambda(S)|} \right)^2 \sum_{\Lambda(S^c)} (\rho_{B(j'_1, \ldots, j'_K)} - \rho)^2
\]

\[
= \left( \frac{1}{|\Lambda(S)|} \right) \sum_{\Lambda(S^c)} (\rho_B(j_1, \ldots, j_K) - \rho)^2
\]

\[
= (\rho_S(1) - \rho)^2.
\]

This completes the proof of the lemma.

A.2 Proof of Lemma 4

**PROOF.** Table 3 presents the non-central moments of a normal variable \( X \sim N(\mu, \sigma^2) \). By Table 3, \( \text{Var}[X^2] = 2\sigma^2(2\mu^2 + \sigma^2) \), and \( \text{Cov}(X^2, Xi) = 2\mu_i\sigma_i \). Thus

\[
\text{Var}[f] = b_0^2 \text{Var}[X^2] + b_1^2 \text{Var}[X] + 2b_0 b_1 \text{Cov}(X^2, Xi) = \sigma^2 [(b_1 + 2b_0\mu)^2 + 2b_0^2\sigma_i^2].
\]

This completes the proof of the lemma.
Table 3: Non-central moments of $X \sim N(\mu, \sigma^2)$

<table>
<thead>
<tr>
<th>$k$</th>
<th>Non-central moment $E(X^k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\mu$</td>
</tr>
<tr>
<td>2</td>
<td>$\mu^2 + \sigma^2$</td>
</tr>
<tr>
<td>3</td>
<td>$\mu^3 + 3\mu\sigma^2$</td>
</tr>
<tr>
<td>4</td>
<td>$\mu^4 + 6\mu^2\sigma^2 + 3\sigma^4$</td>
</tr>
</tbody>
</table>

A.3 Proof of Theorem 1

To prove the theorem, we need the following result:

**Theorem 4.** $\text{Var}[\rho_n]$ in Equation (8) can be bounded as:

$$\text{Var}[ho_n] \leq (1 - (1 - \frac{1}{n})^K)\rho(1 - \rho).$$

**Proof.** The proof is straightforward since this is a special case of Theorem 8 (see Appendix A.8). Specifically, we have $\text{Var}[ho_n] = \text{Cov}(\rho_n, \rho_n)$. By letting $m = K$ in Theorem 8, we obtain

$$\text{Var}[ho_n] \leq f(n, K)\rho(1 - \rho),$$

where $f(n, K) = 1 - (1 - \frac{1}{n})^K$ and $\rho(1 - \rho)$. This completes the proof. □

According to Theorem 4, $\text{Var}[ho_n] \to 0$ as $n \to \infty$. We are now ready to prove Theorem 1:

**Proof.** (of Theorem 1) Let $\mu_1 = \rho_n$ and $E[\rho_n] = \rho$. Define $g(X) = b_0X^2 + b_1X + b_2$. Since $\rho_n$ is strongly consistent, $\rho_n \xrightarrow{as} \rho$. Moreover, since $g$ is continuous, we have $f = g(\rho_n) \xrightarrow{as} g(\rho)$ by the continuous mapping theorem. Note that $g(\rho)$ is a constant. On the other hand, by Lemma 4 and Theorem 4, $\text{Var}[f] \to 0$ as $n \to \infty$. As a result,

$$f^N \xrightarrow{d} E[f] = g(\rho).$$

Since $f \xrightarrow{as} g(\rho)$ implies $f \xrightarrow{p} g(\rho),

$$f^N - f \xrightarrow{d} (g(\rho) - g(\rho)) = 0$$

by Slutsky’s theorem. Since 0 is a constant, $f^N - f \xrightarrow{p} 0$ as well. As a result, we have $f^N \xrightarrow{p} f$. □

A.4 Similar Results for (C6’)

**Lemma 8.** If $X_1 \sim N(\mu_1, \sigma_1^2)$, $X_2 \sim N(\mu_2, \sigma_2^2)$, and $f = b_0X_1X_2 + b_1X_1 + b_2X_2 + b_3$, then

$$\text{Var}[f] = \sigma_1^2(b_0\mu_2 + b_1)^2 + \sigma_2^2(b_0\mu_1 + b_2)^2 + b_0^2\sigma_1^2\sigma_2^2.$$

**Proof.** Since $X_1 \perp X_2$, $\text{Cov}(X_1, X_2) = 0$. So

$$\text{Var}[f] = b_0^2 \cdot \text{Var}[X_1X_2] + b_1^2\sigma_1^2 + b_2^2\sigma_2^2 + 2b_0b_1 \cdot \text{Cov}(X_1X_2, X_1) + 2b_0b_2 \cdot \text{Cov}(X_1X_2, X_2).$$

Since $\text{Var}[X_1X_2] = \mu_1\mu_2\sigma_1^2 + \mu_1^2\sigma_2^2 + \mu_2^2\sigma_1^2$, $\text{Cov}(X_1X_2, X_1) = \mu_1\mu_2\sigma_1^2$, and similarly $\text{Cov}(X_1X_2, X_2) = \mu_1\mu_2\sigma_2^2$, we can have the desired expression for $\text{Var}[f]$ by substituting these quantities. □

**Theorem 5.** Suppose that $X_1 \sim N(\mu_1, \sigma_1^2)$, $X_2 \sim N(\mu_2, \sigma_2^2)$, and $f = b_0X_1X_2 + b_1X_1 + b_2X_2 + b_3$. Let $f^N \sim N(E[f], \text{Var}[f])$, where $\text{Var}[f]$ is shown in Lemma 8. Then $f^N \xrightarrow{p} f$.

**Proof.** Let $\mu_1 = \rho_n$ and $\mu_2 = \rho'$. Suppose that $E[\rho_n] = \rho$ and $E[\rho'] = \rho'$. Define

$$g(X, Y) = b_0XY + b_1X + b_2Y + b_3.$$

Since $\mu_1$ and $\mu_2$ are both strongly consistent, $\rho_n \xrightarrow{as} \rho$ and $\rho'\xrightarrow{as} \rho'$. Moreover, since $g$ is continuous, by the continuous mapping theorem we have

$$f = g(\rho_n, \rho') \xrightarrow{as} g(\rho, \rho').$$

Note that $g(\rho, \rho')$ is a constant. On the other hand, by Lemma 8 and Theorem 4, $\text{Var}[f] \to 0$ as $n \to \infty$. As a result, since $X_1 \perp X_2$, by Lemma 2, it follows that

$$f^N \xrightarrow{d} E[f] = g(\rho, \rho'),$$

$$f^N - f \xrightarrow{d} (g(\rho, \rho') - g(\rho, \rho')) = 0$$

by Slutsky’s theorem. Since 0 is a constant, $f^N - f \xrightarrow{p} 0$ as well. As a result, we have $f^N \xrightarrow{p} f$. □

A.5 Proof of Theorem 2

**Proof.** Since $f_{kc}$ and $c$ are independent, we have

$$E[t_{kc}] = E[f_{kc}^N] = E[f_{kc}] E[c]$$

and

$$\text{Var}[t_{kc}] = E^2[f_{kc}] \text{Var}[c] + E[f_{kc}] \text{Var}[c] + \text{Var}[c] \text{Var}[f_{kc}].$$

Since $\text{Var}[f_{kc}] \to 0$ as $n \to \infty,

$$\text{Pr}(t_{kc}^N) \to N(E[f_{kc}] E[c], E^2[f_{kc}] \text{Var}[c]).$$

In other words, $t_{kc}^N \xrightarrow{d} E[f_{kc}] c$.

On the other hand, $f_{kc}^N \xrightarrow{p} E[f_{kc}] c$ and $c \xrightarrow{p} c$. As a result, we have $(f_{kc}^N c) \xrightarrow{p} (E[f_{kc}] c) c$. By the continuous mapping theorem, $(f_{kc}^N c) \xrightarrow{p} E[f_{kc}] c$. That is, $t_{kc} \xrightarrow{p} E[f_{kc}] c$, which implies $t_{kc} \xrightarrow{d} E[f_{kc}] c$. This completes the proof of the theorem. □

A.6 Convergence of $g_c^N$

**Theorem 6.** Let $g_c = \sum_{k=1}^m f_{kc}^N$ and

$$g_c^N \sim N(E[g_c], \text{Var}[g_c]).$$

Then $g_c^N \xrightarrow{p} g_c$.

**Proof.** Since by definition $f_{kc}^N \sim N(E[f_{kc}], \text{Var}[f_{kc}])$ and $\text{Var}[f_{kc}] \to 0$, $f_{kc}^N \xrightarrow{d} E[f_{kc}]$. Since $E[f_{kc}]$ is a constant, it implies that $f_{kc}^N \xrightarrow{p} E[f_{kc}]$. By the continuous mapping theorem, $g_c \xrightarrow{p} \sum_{k=1}^m E[f_{kc}]$. On the other hand, since $\text{Var}[g_c] \to 0$, $g_c^N \xrightarrow{d} E[g_c]$. Since $E[g_c]$ is again a constant, it follows that

$$g_c^N \xrightarrow{p} E[g_c] = \sum_{k=1}^m E[f_{kc}].$$

As a result, by applying the continuous mapping theorem again, we have $g_c^N - g_c \xrightarrow{p} 0$ and hence $g_c^N \xrightarrow{p} g_c$. □
A.7 A Tighter Upper Bound for Covariance

Consider two operators $O$ and $O'$ where $O \in \text{Desc}(O')$. Suppose that $|R| = K$, $|R'| = K'$, and $|R \cap R'| = m (m \geq 1)$. Let the estimators for $O$ and $O'$ be $\rho_n$ and $\rho'_n$ where $n$ is the number of sample steps, and define $E = E[\rho_n]$ and $E' = E[\rho'_n]$.

**Theorem 7.** Let $S_\rho$, $\Lambda(S)$, and $\rho_S(1)$ be the same as that defined in Theorem 3. Define

$$
\sigma^2_S = \frac{1}{|\Lambda(S)|} \sum_{l \in \Lambda(S)} \var[\rho_S(l) - \rho]^2,
$$

and

$$
S^2_p(m, n) = \sum_{r=1}^m (1 - \frac{1}{n})^{m-r} \frac{1}{n^r} \sum_{S \in S_r} \sigma^2_S.
$$

We then have

$$
\var\{\text{Cov}(\rho_n, \rho'_n)\} \leq \sqrt{S^2_p(m, n)S^2_p(n, m)} \leq \sqrt{\var[\rho_n] \var[\rho'_n]}.
$$

We next prove Theorem 7. To establish the first inequality in the theorem, namely,

$$
\var\{\text{Cov}(\rho_n, \rho'_n)\} \leq \sqrt{S^2_p(m, n)S^2_p(n, m)} \leq \sqrt{\var[\rho_n] \var[\rho'_n]}.
$$

we need two lemmas. The first one gives an explicit expression of the covariance $\text{Cov}(\rho_n, \rho'_n)$, which is quite similar to the expression of $\var[\rho_n]$ shown in Theorem 3.

**Lemma 9.** Let $S_r$ be the collection of all subsets of $R$ with size $r$ (for $1 \leq r \leq m$). Define

$$
\text{Cov}(\rho, \rho') = \frac{1}{|\Lambda(S)|} \sum_{l \in \Lambda(S)} (\rho_S(l) - \rho)(\rho'_S(l) - \rho').
$$

Then

$$
\text{Cov}(\rho_n, \rho'_n) = \sum_{r=1}^m \frac{(n-1)^{m-r}}{n^m} \sum_{S \in S_r} \text{Cov}(\rho, \rho').
$$

Here $\rho_S(1)$ and $\rho'_S(1)$ are the same as that in Theorem 3, defined over $R$ and $R'$ respectively.

**Proof.** The idea is similar to our proof of Theorem 3. Let $K = |R|$ and $K' = |R'|$. We have

$$
\rho_n = \frac{1}{n^K} \sum_{i_1=1}^K \cdots \sum_{i_K=1}^K \rho_{B(i_{L_{i_1}}, \ldots, L_{i_K})},
$$

and

$$
\rho'_n = \frac{1}{n^{K'}} \sum_{l_{i_1}+1}^{K'} \cdots \sum_{l_{K'}+1}^{K'} \rho'_{B(l_{L_{i_1}}, \ldots, L_{i_K'})}.
$$

Therefore,

$$
E[\rho_n \rho'_n] = \frac{1}{n^{K+K'}} \sum_{k=1}^{n} \sum_{k'=1}^{n'} E[\rho_B \rho'_B],
$$

and

$$
E[\rho_n] E[\rho'_n] = \frac{1}{n^{K+K'}} \sum_{k=1}^{n} \sum_{k'=1}^{n'} E[\rho_B] E[\rho'_B].
$$

Hence, by letting $d_n = \text{Cov}(\rho_n, \rho'_n) = E[\rho_n \rho'_n] - E[\rho_n] E[\rho'_n]$,

$$
d_n = \frac{1}{n^{K+K'}} \sum_{k=1}^{n} \sum_{k'=1}^{n'} (E[\rho_B \rho'_B] - E[\rho_B] E[\rho'_B])
$$

$$
= \frac{1}{n^{K+K'}} \sum_{k=1}^{n} \sum_{k'=1}^{n'} \text{Cov}(\rho_B, \rho'_B).
$$

If $B$ and $B'$ share no blocks, then $\rho_B$ and $\rho'_B$ are independent and thus $\text{Cov}(\rho_B, \rho'_B) = 0$. Thus we only need to consider the case that $B$ and $B'$ share at least one block. Similarly as before, we partition the pairs $(\rho_B, \rho'_B)$ based on the number of blocks $B$ and $B'$ share. According to Lemma 6, for a fixed $S \in \mathcal{S}_r$, the number of pairs $(\rho_B, \rho'_B)$ such that $\rho_B \sim S \rho'_B$ is

$$
n^{K} (n-1)^{m-r} n^{K'-m} = n^{K+K'-m} (n-1)^{m-r}.
$$

We hence have

$$
\text{Cov}(\rho_n, \rho'_n) = \sum_{r=1}^m \frac{(n-1)^{m-r}}{n^m} \sum_{S \in \mathcal{S}_r} \text{Cov}(\rho_B, \rho'_B).
$$

Similarly as in Lemma 7, we have

$$
\text{Cov}(\rho_B, \rho'_B) = E_{\rho_B \sim S \rho'_B} [(\rho_B - \rho)(\rho'_B - \rho')],
$$

and hence

$$
\text{Cov}(\rho_B, \rho'_B) = \frac{1}{|\Lambda(S)|} \sum_{l \in \Lambda(S)} (\rho_B - \rho)(\rho'_B - \rho').
$$

Now let $K$ and $K'$ be the indexes of the relations in $R$ and $R'$ respectively. Denote $S_{K'} = K - S$ and $S_{K'} = K' - S$. Let

$$
E = E_{\rho_B \sim S \rho'_B} [(\rho_B - \rho)(\rho'_B - \rho')].
$$

We have

$$
E \leq \frac{1}{|\Lambda(S)|} \sum_{l \in \Lambda(S)} (\rho_B - \rho)(\rho'_B - \rho')
$$

$$
= \left( \frac{1}{|\Lambda(S)|} \sum_{l \in \Lambda(S)} (\rho_B - \rho) \right) \left( \frac{1}{|\Lambda(S')|} \sum_{l' \in \Lambda(S')} (\rho_B - \rho') \right)
$$

$$
= (\var[\rho_B] - \rho)(\var[\rho_B] - \rho')
$$

$$
= (\rho_S(1) - \rho)(\rho'_S(1) - \rho').
$$

This completes the proof of the lemma. ∎

Our second lemma further provides an upper bound for $\text{Cov}(\rho, \rho')$:

**Lemma 10.** Let $S \in \mathcal{S}_r$. Then we have

$$
|\text{Cov}(\rho, \rho')| \leq \sqrt{\sigma^2_S \cdot (\sigma^2'_S)^2}.
$$

**Proof.** Let $d_S = \text{Cov}(\rho, \rho')$ and $d^2_S = (\rho_S(1) - \rho)^2$. By the Cauchy-Schwarz inequality, we have

$$
d^2_S \leq \frac{1}{|\Lambda(S)|} \left( \sum_{l \in \Lambda(S)} (\rho_B - \rho)(\rho'_B - \rho') \right)^2
$$

$$
\leq \frac{1}{|\Lambda(S)|} \left( \sum_{l \in \Lambda(S)} d^2_S \right) \left( \sum_{l \in \Lambda(S)} d^2_S \right)
$$

$$
= \left( \frac{1}{|\Lambda(S)|} \sum_{l \in \Lambda(S)} d^2_S \right) \left( \sum_{l \in \Lambda(S)} d^2_S \right)
$$

$$
= \sigma^2_S \cdot (\sigma^2'_S)^2.
$$

The lemma then follows immediately. ∎
We can now prove the first inequality in Theorem 7:

**Proof.** Let \( d_n = \text{Cov}(\rho_n, \rho'_n) \). By Lemma 9 and 10 we have

\[
|d_n| = \left| \sum_{r=1}^{m} \left(1 - \frac{1}{n}\right) m-r \left(\frac{1}{n}\right)^r \sum_{S \in S_r} \text{Cov}(\rho, \rho') \right| \\
\leq \sum_{r=1}^{m} \left(1 - \frac{1}{n}\right) m-r \left(\frac{1}{n}\right)^r \sum_{S \in S_r} \sqrt{\sigma_r^2(\sigma')^2}.
\]

By the Cauchy-Schwarz inequality, we have

\[
\sum_{S \in S_r} \sqrt{\sigma_r^2(\sigma')^2} \leq \sqrt{\sum_{S \in S_r} \sigma_r^2 \sum_{S \in S_r} \sigma_r^2}. 
\]

Combining these two inequalities, we obtain

\[
|d_n| \leq \sum_{r=1}^{m} \left(1 - \frac{1}{n}\right) m-r \left(\frac{1}{n}\right)^r \sum_{S \in S_r} \sigma_r^2 \sum_{S \in S_r} \sigma_r^2. 
\]

Now define

\[
A_r = \sqrt{(1 - \frac{1}{n}) m-r \left(\frac{1}{n}\right)^r \sum_{S \in S_r} \sigma_r^2}. 
\]

Then \( |d_n| \leq \sum_{r=1}^{m} A_r A'_r \). Applying the Cauchy-Schwarz inequality again,

\[
|d_n| \leq \left( \sum_{r=1}^{m} A_r^2 \left( \sum_{r=1}^{m} (A')^2 \right) \right) = \sqrt{S^2(m, n)S^2_R(m, n)}, 
\]

which completes the proof of the inequality. \( \square \)

To establish the second inequality in the theorem, namely,

\[
\sqrt{S^2(m, n)S^2_R(m, n)} \leq \sqrt{\text{Var}[\rho_n]} \sqrt{\text{Var}[\rho'_n]}, 
\]

we need two more lemmas. The first one states that the \( \sigma^2_R \) has some nice monotonicity property:

**Lemma 11.** Let \( S \in S_r \) and \( S' \in S_{r+1} \) such that \( S \subset S' \), for \( 1 \leq r \leq K - 1 \). Then \( \sigma^2_R \leq \sigma^2_R \).

**Proof.** Without loss of generality, let \( S = \{1, \ldots, r\} \) and \( S' = \{1, \ldots, r+1\} \). For a given \( I = (j_1, \ldots, j_r) \in \Lambda(S) \), let \( I' = (j_1, \ldots, j_r, j) \), for \( 1 \leq j \leq m_r+1 \). Since \( \Lambda(S') = \Lambda(S) \times \{1, \ldots, m_{r+1}\} \), we have \( \Lambda(S') = \Lambda(S') \times \{1, \ldots, m_{r+1}\} \) and thus \( |\Lambda(S')| = m_{r+1} |\Lambda(S')| \). Therefore, by letting \( d^2_\rho = (\rho_I - \rho')^2 \), it follows that

\[
d^2_\rho = \left( \frac{1}{|\Lambda(S')|} \sum_{\Lambda(S')} \rho_B - \rho \right)^2 = \left( \frac{1}{|\Lambda(S')|} \sum_{\Lambda(S')} (\rho_B - \rho) \right)^2 = \frac{1}{m_r+1} \sum_{j=1}^{m_{r+1}} \left( \frac{1}{|\Lambda((S')^c)|} \sum_{\Lambda((S')^c)} \rho_B - \rho \right)^2.
\]

By the Cauchy-Schwarz inequality, we have

\[
d^2_\rho = \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} \left( \frac{1}{|\Lambda((S')^c)|} \sum_{\Lambda((S')^c)} \rho_B - \rho \right)^2 
\]

\[
= \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} \left( \frac{1}{|\Lambda((S')^c)|} \sum_{\Lambda((S')^c)} \rho_B - \rho \right)^2
\]

\[
= \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} \left( \frac{1}{|\Lambda((S')^c)|} \sum_{\Lambda((S')^c)} \rho_B - \rho \right)^2
\]

\[
\leq \frac{1}{|\Lambda(S)|} \sum_{\Lambda(S)} (\rho_B - \rho)^2
\]

\[
\leq \frac{1}{|\Lambda(S)|} \sum_{\Lambda(S)} (\rho_B - \rho)^2
\]

\[
\leq \sigma^2_R.
\]

This completes the proof of the lemma. \( \square \)

Our next lemma further shows that the \( S^2_R(m, n) \) also has some similar monotonicity property:

**Lemma 12.** For \( m \geq 1 \), we have

\[
S^2_R(m, n) \leq S^2_R(m + 1, n).
\]

**Proof.** We should be careful now since \( S_r \) is actually related to \( m \). Specifically, \( S_r \) is all the \( r \)-subsets of \( \{1, \ldots, m\} \). To make this more explicit, we further use \( S^2_R(m) \) to indicate this relationship. Moreover, to simplify notation, we define

\[
A^2_R(m) = \sum_{S \in S^2_R(m)} \sigma^2_R.
\]

Furthermore, if \( r = m \), then \( S^2_R(m) \) contains only one single element \( \{1, \ldots, m\} \). We thus simply use \( \sigma^2_R \) to represent \( A^2_R(m) \), i.e.,

\[
\sigma^2_R = \sum_{S \in S^2_R} \sigma^2_R.
\]

Now consider \( S_{m+1} = S^2_R(m + 1, n) \). We have

\[
S_{m+1} = \sum_{r=1}^{m+1} \left(1 - \frac{1}{n}\right)^{m+1-r} \left(\frac{1}{n}\right)^r A^2_R(m+1)
\]

\[
= \sum_{r=1}^{m+1} \left(1 - \frac{1}{n}\right)^{m+1-r} \left(\frac{1}{n}\right)^r A^2_R(m+1) + \left(\frac{1}{n}\right)^{m+1}\sigma^2_R.
\]

Define \( \Delta^2_R(m+1) = \sum_{S \in S^2_R(m+1)} \sigma^2_R \). Then

\[
\Delta^2_R(m+1) = A^2_R(m+1) - A^2_R(m).
\]

We therefore have

\[
\Delta^2_R(m+1) = \sum_{r=1}^{m+1} \left(1 - \frac{1}{n}\right)^{m+1-r} \left(\frac{1}{n}\right)^r \Delta^2_R(m+1) + \left(\frac{1}{n}\right)^{m+1}\sigma^2_R.
\]

\[^8\text{More generally, the indexes could be represented as } J_m = \{j_1, \ldots, j_m\} \text{ and } J_{m+1} = \{j_1, \ldots, j_m, j_{m+1}\} \text{ such that } J_m \subset J_{m+1}. \text{ We used } J_m = \{1, \ldots, m\} \text{ and } J_{m+1} = \{1, \ldots, m, m+1\} \text{ in our proof without loss of generality.}\]
Let us further define \( S^m_r = \emptyset \) if \( r > m \). Then \( \Delta_{m+1}^{(m+1)} = \sigma_{m+1}^2 \), and therefore
\[
B_m = \frac{m+1}{n} \sum_{r=1}^{m+1} (1 - \frac{1}{n})^{m+1-r} \left( \frac{1}{n} \right)^r \Delta_{m+1}^{(m+1)}.
\]

Next, consider some \( S \in S_m^{(m+1)} \setminus S_n^{(m)} \) where \( r \geq 2 \). Note that \( S \) must contain \( m+1 \) since otherwise \( S \in S_n^{(m)} \). What’s more, if we remove \( m+1 \) from \( S \), then \( S \) must be now in \( S_n^{(m)} \), that is, \( S \setminus \{m+1\} \in S_n^{(m)} \). On the other hand, for any \( S' \in S_n^{(m)} \), we can obtain an element in \( S_n^{(m+1)} \setminus S_n^{(m)} \) by simply adding \( m+1 \), that is, \( S' \cup \{m+1\} \in S_n^{(m+1)} \setminus S_n^{(m)} \). We therefore have established a 1-1 mapping \( \varphi \) between \( S_n^{(m+1)} \setminus S_n^{(m)} \) and \( S_n^{(m)} \).

Furthermore, note that for any \( S \in S_n^{(m+1)} \setminus S_n^{(m)} \), we have \( \varphi(S) \subset S \). Hence by Lemma 11, \( \sigma^2_{\varphi(S)} \leq \sigma^2_{S} \). Therefore, we have
\[
\Delta_{m+1}^{(m+1)} \geq \sum_{S \in S_n^{(m+1)} \setminus S_n^{(m)}} \sigma^2_{\varphi(S)} = A_{m+1}^{(m)}.
\]

As a result, we have
\[
B_m \geq \left( 1 - \frac{1}{n} \right) m \frac{1}{m} \Delta_{m+1}^{(m+1)} = \frac{1}{n} (1 - \frac{1}{n})^m \sigma_{m+1}^2 \geq 0.
\]

Hence, \( B_m \geq \frac{1}{n} S_m \). Since \( S_{m+1} = (1 - \frac{1}{n}) S_m + B_m \), we conclude that \( S_{m+1} \geq S_m \). This completes the proof of the lemma. \( \square \)

It is now easy to prove the second inequality in Theorem 7:

**Proof.** Based on Lemma 12, by induction, we can easily prove that \( S_n^p(m, n) \leq \text{Var}[\rho_n] \) and \( S_n^p(m, n) \leq \text{Var}[\rho_m] \); since \( m \leq \min\{K, K'\} \). The inequality then follows.

For our special case in this paper where \( \text{Cov}(\rho_n, \rho_m) \neq 0 \), we will always have \( m = \min\{K, K'\} \). Without loss of generality, let \( m = K \). Then \( S_n^p(m, n) = \text{Var}[\rho_n] \), and we only need to approximate \( S_n^p(m, n) \) with \( S_n^p(K, n) \), which by Lemma 12 is guaranteed to be superior to \( \text{Var}[\rho_n] \). Intuitively, the bigger \( K' - K \) is, the bigger the gap is between \( S_n^p(K, n) \) and \( \text{Var}[\rho_n] \). In fact, in the proof of Lemma 12, we have actually showed that \( S_{m+1} \geq S_m + C_m \). So we can roughly estimate that
\[
\text{Var}[\rho_n] - S_n^p(K, n) \geq \frac{1}{n} (1 - \frac{1}{n})^K \sum_{r=K+1}^{K'} \sigma_r^2.
\]

**A.8 More Bounds for Covariances**

We can actually have another upper bound for \( \text{Cov}(\rho_n, \rho_m) \):

**Theorem 8.** We have
\[
|\text{Cov}(\rho_n, \rho_m)| \leq f(n, m) g(\rho) g(\rho'),
\]
where \( f(n, m) = 1 - (1 - \frac{1}{n})^m \) and \( g(\rho) = \sqrt{\rho(1 - \rho)} \).

**Proof.** As in the proof of Lemma 9, let \( K \) and \( K' \) be the indexes of the relations in \( R \) and \( R' \) respectively. By Lemma 11, we have \( \sigma_3^2 \leq \sigma_K^2 \) and \( (\sigma'_3)^2 \leq \sigma_{K'}^2 \). Moreover, consider
\[
\sigma_K^2 = \frac{1}{|A(K)|} \sum_{r \in A(K)} (\rho_K(1 - \rho)).
\]

Since we use the tuple-level partition scheme, we have \( \rho_K(1) = 1 \) or \( \rho_K(1) = 0 \). Therefore,
\[
\sigma_K^2 = (1 - \rho)^2 \frac{1}{|A(K)|} \sum_{r \in A(K)} (\rho_K(1)) \quad \text{or} \quad \sigma_K^2 = (1 - \rho)^2 \cdot \rho + \rho^2 \cdot (1 - \rho) = \rho(1 - \rho).
\]

Similarly, we have \( \sigma_{K'}^2 = \rho'(1 - \rho') \). Hence,
\[
\text{Cov}_S(\rho, \rho') \leq \sqrt{\sigma_3^2 \sigma_{K'}^2} \leq \sqrt{\rho(1 - \rho) \cdot \rho'(1 - \rho')},
\]
and therefore, by letting \( g(\rho) = \sqrt{\rho(1 - \rho)} \),
\[
|d_n| \leq \frac{m}{n} \sum_{r=1}^{m} \left( \frac{n-1}{n} \right)^{m-r} \sum_{S \in S_r} \text{Var}[\rho_n] \sum_{r \in S_r} g(\rho) g(\rho') \leq \frac{m}{n} \sum_{S \in S_r} g(\rho) g(\rho') \leq g(\rho) g(\rho') [1 - (1 - \frac{1}{n})^m].
\]

This completes the proof of the theorem. \( \square \)

When \( n \) is large, \( (1 - \frac{1}{n})^m \approx 1 - \frac{m}{n} \). As a result, \( 1 - \frac{1}{n} = \frac{m}{n} \). Therefore, when \( n \to \infty \), \( \text{Cov}(\rho_n, \rho_m) \to 0 \). This is intuitively true considering the strong consistency of \( \rho_n \). If we keep taking samples, finally the estimated selectivity should converge to the actual selectivity (a constant). On the other hand, a larger \( m \) implies a larger bound since the computations of \( \rho_n \) and \( \rho_m \) share more samples. Another interesting observation is that the bound also depends on the actual selectivities \( \rho \) and \( \rho' \). Note that \( g(\rho) \) is minimized at \( \rho = 0 \) or \( \rho = 1 \) (with \( \text{min}(0) = 0 \) and is maximized at \( \rho = \frac{1}{2} \)). To shed some light on this, observe that whenever \( \rho \) or \( \rho' \) is 0 or 1, \( \rho_n \) or \( \rho_m \) is always 0 or 1 regardless of the number of samples. Hence \( \text{Cov}(\rho_n, \rho_m) \to 0 \) in such cases.

An natural question is how good this bound is compared with the two bounds in Section 5.3.2. Let us name these three bounds as

**(B1)** \( \sqrt{S_n^p(m, n)} S_n^p(m, n) \), the first bound in Theorem 7;

**(B2)** \( \sqrt{\text{Var}[\rho_n]} \text{Var}[\rho_m] \), the second bound in Theorem 7;

**(B3)** \( f(n, m) g(\rho) g(\rho') \), the bound in Theorem 8.

By Theorem 7, we already know that \( B_1 \leq B_2 \). Next, according to the proof of Theorem 8, \( \sigma_3^2 \leq \rho(1 - \rho) \) and \( (\sigma'_3)^2 \leq \rho'(1 - \rho') \). We then immediately have
\[
\sqrt{S_n^p(m, n)} S_n^p(m, n) \leq f(n, m) g(\rho) g(\rho'),
\]
by the definition of \( S_n^p(m, n) \). That is, \( B_1 \leq B_3 \). Moreover, by Theorem 4, we have
\[
|\text{Cov}(\rho_n, \rho_m)| \leq \sqrt{\text{Var}[\rho_n]} \text{Var}[\rho_m] \leq f(n, m) g(\rho) g(\rho'),
\]
where
\[ f(n) = \sqrt{\frac{1 - (1 - \frac{1}{n})^K}{1 - (1 - \frac{1}{n})^{K'}}}. \]

When \( n \) is large, \( 1 - (1 - \frac{1}{n})^K \approx \frac{K}{n} \) and \( 1 - (1 - \frac{1}{n})^{K'} \approx \frac{K'}{n} \).

Therefore, the right hand is close to \( \sqrt{\frac{K}{n} n^{K'} / n^{K'}} = \sqrt{K / n} \). Since \( m \leq \min\{K, K'\} < \sqrt{K K'} \), we know that \( B_3 \approx \rho \) better than the upper bound of \( B_2 \). However, in general \( B_2 \) and \( B_3 \) are incomparable.

One more issue of \( B_3 \) is that it includes the true selectivities \( \rho \) and \( \rho' \) that are not known without running the query. As a result, \( B_3 \) is not directly computable. Nonetheless, when \( n \) is large, we can simply use the observed \( \rho_m \) and \( \rho'_m \) as approximations due to the strong consistency of \( \rho_m \).

Finally, the techniques we used in the proof of Theorem 8 can be further generalized to establish similar bounds for other covariances such as \( \text{Cov}(\rho''_1, (\rho')_n) \) and \( \text{Cov}(\rho''_1, \rho'_m) \).

**Theorem 9.**

\[ \text{Cov}(\rho''_1, (\rho')_n) \leq f(n, m) h(\rho)(\rho - \rho' + 1), \]

where
\[ f(n, m) = \left( 1 - \frac{1}{n} \right) K^{K-m}(1 - \frac{2}{n} m) \left( 1 - \frac{3}{n} m \right) \]
and
\[ h(\rho) = \sqrt{\rho(1 - \rho) + \rho^2 - 1}. \]

When \( n \) is large, we can approximate \( f(n, m) \) as:
\[ f(n, m) \approx \left( 1 - \frac{K}{n} - \frac{m}{n} \right) \left( 1 - \frac{2m}{n} \right) \left( 1 - \frac{3m}{n} \right) \]
\[ \approx \left( 1 - \frac{K}{n} \right) \left( 1 - \frac{K'}{n} \right) \frac{\sqrt{K K'}}{n}. \]

**Proof.** For notational convenience, define \( E_{\rho''} = E[\rho''_1(\rho'_1)^2] \), and \( E_{\rho'} = E[\rho''_1(\rho'_1)^2] \).

We have
\[ E_{\rho''} = E[(\frac{1}{n} n_K \rho_2(\rho''_1)^2) \times (\frac{1}{n} n_K \rho''_1)^2] \]
\[ = \frac{1}{n^{2(K+K')} n_{K+K'}} \sum_{k_1, k_2} \sum_{k'_1, k'_2} \text{E} |\rho_{\ell_1, \ell_2}, \rho'_{\ell_1, \ell_2}|^2, \]
and
\[ E_{\rho'} = E[(\frac{1}{n} n_K \rho_2(\rho''_1)^2) \times (\frac{1}{n} n_K \rho''_1)^2] \]
\[ = \frac{1}{n^{2(K+K')} n_{K+K'}} \sum_{k_1, k_2} \sum_{k'_1, k'_2} \text{E} |\rho_{\ell_1, \ell_2}, \rho'_{\ell_1, \ell_2}|^2. \]

Therefore, by letting \( d''_n = \text{Cov}(\rho''_1, (\rho')_n) \), we have
\[ d''_n = E[\rho''_1(\rho'_1)^2] - E[\rho''_1]E[(\rho'_1)^2] \]
\[ = \frac{1}{n^{2(K+K')} n_{K+K'}} \sum_{k_1, k_2} \sum_{k'_1, k'_2} \text{Cov}(\rho_{\ell_1, \ell_2}, \rho'_{\ell_1, \ell_2}). \]

Note that, among the \( n^{2(K+K')} \) summands in \( d''_n, n^{K(n-1)}(n-2)^m n^{K'-m}(n-3)^{m(n-1)}\) of them involve samples that do not share any blocks and are therefore independent. As a result, for these terms,
\[ \text{Cov}(\rho_{B_1, \rho_{B_2}}, \rho'_{B_1, \rho'_{B_2}}) = 0. \]

Since
\[ \text{Cov}^2(\rho_{B_1, \rho_{B_2}}, \rho'_{B_1, \rho'_{B_2}}) \leq \text{Var}[\rho_{B_1, \rho_{B_2}}] \text{Var}[\rho'_{B_1, \rho'_{B_2}}], \]
we have
\[ |d''_n| \leq \left( 1 - \frac{[n(n-1)]^{K-K'-m}[(n-2)(n-3)]^m}{n^{2(K+K')} n_{K+K'}} \right) \text{Var}[\rho_{B_1, \rho_{B_2}}] \text{Var}[\rho'_{B_1, \rho'_{B_2}}] \]
\[ = \left( 1 - \frac{1}{\rho_{B_1, \rho_{B_2}}^2} \right) \left( 1 - \frac{1}{\rho'_{B_1, \rho'_{B_2}}^2} \right) \text{Var}[\rho_{B_1, \rho_{B_2}}] \text{Var}[\rho'_{B_1, \rho'_{B_2}}]. \]

Next, let us consider
\[ V = \text{Var}[\rho_{B_1, \rho_{B_2}}] = E[(\rho_{B_1, \rho_{B_2}} - E[\rho_{B_1, \rho_{B_2}}])^2]. \]

As before, we partition the pairs \((B_1, B_2)\) based on the blocks they share, which gives us
\[ V = \sum_{r=1}^{K} \sum_{S \in \mathcal{S}_r} \left( \frac{1}{|\mathcal{A}(S)|} \sum_{A \in \mathcal{A}(S)} E_{\mathcal{S}_r}(B_1, B_2) \right), \]
where
\[ E_{\mathcal{S}_r}(B_1, B_2) = \text{E}_{\rho_{B_1, \rho_{B_2}}}(\rho_{B_1, \rho_{B_2}} - E[\rho_{B_1, \rho_{B_2}}]). \]

Now define
\[ \sigma^2 = \frac{1}{|\mathcal{A}(S)|} \sum_{A \in \mathcal{A}(S)} E_{\mathcal{S}_r}(B_1, B_2). \]

Similarly as before, we are able to show that, if \( S \in \mathcal{S}_r, S' \in \mathcal{S}_{r+1} \), and \( S \subseteq S' \), then \( \sigma^2 \leq \sigma^2 \). To see this, without loss of generality, let \( S = \{1, \ldots, r\} \) and \( S' = \{1, \ldots, r+1\} \). For a given \( 1 = \{j_1, \ldots, j_r\} \subseteq \mathcal{A}(S) \), let \( I_1 = \{j_1, \ldots, j_r, j_{r+1}\} \) for \( 1 \leq j \leq m_{r+1} \). Since \( \mathcal{A}(S') = \mathcal{A}(S) \times \{1, \ldots, m_{r+1}\} \) and thus \( |\mathcal{A}(S')| = m_{r+1} |\mathcal{A}(S)| \), we have
\[ V = \frac{1}{|\mathcal{A}(S)|} \sum_{A \in \mathcal{A}(S)} (\rho_{B_1, \rho_{B_2}} - E[\rho_{B_1, \rho_{B_2}}])^2. \]

Let \( E[\rho_{B_1, \rho_{B_2}}] = A \). We then have
\[ E = \frac{1}{|\mathcal{A}(S')|^2} \sum_{A \in \mathcal{A}(S')} (\rho_{B_1, \rho_{B_2}} - 2A + A^2). \]

Consider the terms \( \rho^2_{B_1, B_2} \) and \( \rho_{B_1, \rho_{B_2}} \). We have
\[ E_1 = \frac{1}{|\mathcal{A}(S')|^2} \sum_{A \in \mathcal{A}(S')} \rho^2_{B_1, B_2} \]
\[ = \frac{1}{|\mathcal{A}(S')|^2} \sum_{A \in \mathcal{A}(S')} \rho^2_{B_1} \cdot \frac{1}{|\mathcal{A}(S)|} \sum_{A \in \mathcal{A}(S)} \rho^2_{B_2} \]
\[ = \frac{1}{|\mathcal{A}(S')|^2} \sum_{A \in \mathcal{A}(S')} \rho^2_{B_1} \cdot \rho^2_{B_2}. \]

\[ = \frac{1}{|\mathcal{A}(S')|^2} \sum_{A \in \mathcal{A}(S')} \rho^2_{B_1} \cdot \rho^2_{B_2}. \]
Therefore,

\[ E_2 = \frac{1}{|A(S')|^2} \sum_{\Lambda(S')} \sum_{\Lambda(S')} \rho_{B_1} \rho_{B_2} \]
\[ = \left( \frac{1}{|A(S')|^2} \sum_{\Lambda(S')} \rho_{B_1} \right) \left( \frac{1}{|A(S')|^2} \sum_{\Lambda(S')} \rho_{B_2} \right) \]
\[ = \left( \frac{1}{|A(S')|^2} \sum_{\Lambda(S')} \rho \right)^2 \]
\[ = \frac{1}{|A((S')^c)|} \sum_{j=1}^{m_{r+1}} \sum_{j=1}^{m_{r+1}} \rho_{B_j}^2. \]

Since \( \rho_{B} = 0 \) or \( \rho_{B} = 1 \), we have \( \rho_{B}^2 = \rho_{B} \) and thus \( E_1 = E_2 \). Therefore,

\[ E = (1 - 2A)E_2 + A^2. \]

Furthermore, define

\[ \mathcal{p}_{S'} = \frac{1}{|A((S')^c)|} \sum_{\Lambda((S')^c)} \rho_{B}, \quad \mathcal{p}^2_{S'} = \frac{1}{|A((S')^c)|} \sum_{\Lambda((S')^c)} \rho_{B}^2, \]
\[ \mathcal{p}_{1} \mathcal{p}_{2 S'} = \frac{1}{|A((S')^c)|} \sum_{\Lambda((S')^c)} \sum_{\Lambda((S')^c)} \rho_{B_1} \rho_{B_2}, \]
\[ \mathcal{p}^2_{1} \mathcal{p}^2_{2 S'} = \frac{1}{|A((S')^c)|} \sum_{\Lambda((S')^c)} \sum_{\Lambda((S')^c)} \rho_{B_1}^2 \rho_{B_2}^2, \]
\[ (\rho_1 \rho_2 - A)^2 S' = \frac{1}{|A((S')^c)|} \sum_{\Lambda((S')^c)} \sum_{\Lambda((S')^c)} (\rho_{B_1} \rho_{B_2} - A)^2. \]

By the Cauchy-Schwarz inequality, we have

\[ E_2 = \frac{1}{m_{r+1}} \left( \sum_{j=1}^{m_{r+1}} \mathcal{p}_{S'} \right)^2 \leq \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} \mathcal{p}^2_{S'}. \]

Hence,

\[ E \leq (1 - 2A) \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} \mathcal{p}^2_{S'} + A^2 \]
\[ = \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} \left[ (1 - 2A) \mathcal{p}^2_{S'} + A^2 \right] \]
\[ = \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} \left[ (1 - 2A) \mathcal{p}^2_{S'} - 2A \mathcal{p}^2_{S'} + A^2 \right] \]
\[ = \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} \left[ (\rho_1 \rho_2 - A)^2 S' \right] \]
\[= \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} \left[ (\rho_1 \rho_2 - A)^2 S' \right]. \]

Therefore,

\[ \sigma^2_S = \frac{1}{|A(S)|} \sum_{l \in A(S)} E_{S(l)}(B_1, B_2) \]
\[ \leq \frac{1}{|A(S)|} \sum_{l \in A(S)} \frac{1}{m_{r+1}} \sum_{j=1}^{m_{r+1}} (\rho_1 \rho_2 - A)^2 S' \]
\[ = \frac{1}{|A(S)|} \sum_{l \in A(S)} (\rho_1 \rho_2 - A)^2 S' \]
\[ = \sigma^2_{S'}. \]

As a result, we have \( \sigma^2_S \leq \sigma^2_{S'}. \) Since

\[ \sigma^2_K = \frac{1}{|A(K)|} \sum_{l \in A(K)} E_{K(l)}(B_1, B_2) \]
\[ = \frac{1}{|A(K)|} \sum_{l \in A(K)} (\rho_{B_1} \rho_{B_2} - A)^2 \]
\[ = \frac{1}{|A(K)|} \sum_{l \in A(K)} \rho_{B_1}^2 - \frac{1}{|A(K)|} \sum_{l \in A(K)} \rho_{B_1} \rho_{B_2} + \frac{1}{|A(K)|} \sum_{l \in A(K)} \rho_{B_2} \]
\[ = (1 - A)^2 \cdot \frac{1}{|A(K)|} \sum_{l \in A(K)} I(\rho_{B} = 1) \]
\[ + A^2 \cdot \frac{1}{|A(K)|} \sum_{l \in A(K)} I(\rho_{B} = 0) \]
\[ = (1 - A)^2 \cdot (1 - 2A) \rho + A^2 \cdot (1 - \rho) \]
\[ = (A - \rho)^2 + (\rho - \rho^2). \]

Now consider \( A = E[\rho_{B_1} \rho_{B_2}]. \) We have

\( A = \Pr(\rho_{B_1} = 1, \rho_{B_2} = 1) \)
\( \geq \Pr(\rho_{B_1} = 1, \rho_{B_2} = 1, \rho_{B_1} \sim \rho_{B_2}) \)
\( = \Pr(\rho_{B_1} = 1) \cdot \Pr(\rho_{B_2} = 1) \)
\( = \rho^2. \)

On the other hand,

\( A = \Pr(\rho_{B_1} = 1, \rho_{B_2} = 1) \leq \Pr(\rho_{B_1} = 1) \leq \rho. \)

Thus, \( \rho^2 \leq A \leq \rho. \) So we have

\( \rho - \rho^2 \leq \sigma^2_K \leq (\rho^2 - \rho^2) + (\rho - \rho^2). \)

Therefore,

\( V = \text{Var}[\rho_{B_1} \rho_{B_2}] \)
\[ = \sum_{r=1}^{K} \frac{(n-1)^{K-r}}{n^K} \times \sum_{S \in \mathcal{S}} \sigma^2_S \]
\[ \leq \sum_{r=1}^{K} \frac{(n-1)^{K-r}}{n^K} \times \sum_{S \in \mathcal{S}} \sigma^2_K \]
\[ \leq [(\rho^2 - \rho^2) + (\rho - \rho^2)] \cdot (1 - (1 - \frac{1}{n})^K). \]
Similarly, we have
\[
V' = \text{Var}[\rho_{B_1}^{2} \rho_{B_2}^{2}]
\leq [(\rho')^2 - (\rho')^2 + (\rho' - (\rho')^2) \cdot (1 - (1 - \frac{1}{n})^{K'})]
\]
As a result, since
\[
|d_n^2| \leq (1 - (1 - \frac{1}{n})^{K'}) (1 - \frac{2}{n})^m (1 - \frac{3}{n})^m) \cdot \text{Var}[\rho_{B_1} \rho_{B_2}] \rho_{B_1}^{2} \rho_{B_2}^{2}]
\]
we have
\[
|\text{Cov}(\rho_n^2, (\rho')^2)| = |d_n^2| \leq f(n, m) h(\rho) h(\rho'),
\]
where
\[
f(n, m) = (1 - (1 - \frac{1}{n})^{K'}) (1 - \frac{2}{n})^m (1 - \frac{3}{n})^m)
\cdot (1 - (1 - \frac{1}{n})^{K'})^2 (1 - (1 - \frac{1}{n})^{K'})^2,
\]
and \(h(\rho) = \sqrt{\rho(1 - \rho)(\rho - \rho^2 + 1)}\). This completes the proof of
the theorem. \(\square\)

With very similar arguments, we are able to show that
\[\text{Theorem 10.} \quad |\text{Cov}(\rho_n^2, (\rho')^2)| \leq f(n, m) h(\rho) g(\rho'),\]
where
\[
f(n, m) = [1 - (1 - \frac{1}{n})^{K'} (1 - \frac{2}{n})^m]
\cdot \sqrt{1 - (1 - \frac{1}{n})^{K'} \sqrt{1 - (1 - \frac{1}{n})^{K'}}},
\]
g(\rho) = \sqrt{\rho(1 - \rho)}, and \(h(\rho) = \sqrt{\rho(1 - \rho)(\rho - \rho^2 + 1)}\).

The approximate version is:
\[
f(n, m) \approx \left[1 - \left(1 - \frac{K'}{n}\right)(1 - \frac{2m}{n})\right] \frac{\sqrt{KK'}}{n^2}
\]
\approx \frac{(K + 2m) \sqrt{KK'}}{n^2}.

\section{The Complete Framework}

We present the complete framework of estimating the distribution of \(t_q\) in Algorithm 2. Note that, the framework is a two-stage one: we first obtain the marginal distributions of the selectivities via sampling, and then obtain the distribution of \(t_q\).

It is worth to point out that a more straightforward, one-stage alternative can also solve the problem: we keep running the query plan over different sample tables and observe the joint distribution of the selectivities. It will then directly give the distribution of the estimated running times: we simply plug in each observed selectivity vector \(X\) to the cost formulas and compute the running times. However, the overhead of this approach might be prohibitive in practice: we need the same number of sample runs as the observations we need to build the histogram of the running times.

Nonetheless, this conceptually simpler framework is of some theoretic interest. Note that in our current framework, we view the execution time as a function of the selectivities over all operators. Can we instead view the time as a function of the selectivities over just the leaf operators? The answer is no, because the selectivities of the internal operators cannot be simply determined by the selectivities of the leaf nodes. However, we can indeed view the time as a function of the leaf tables. That is, as long as we fix the input table of each leaf (i.e., scan) operator, the selectivity of each internal operator is also fixed and hence the running time can be determined. Since different leaf tables may lead to the same selectivity on a leaf operator, this explains why simply fixing the selectivities of the leaf operators may not be sufficient for characterizing the running time of the query plan. However, a function of tables is not feasible for mathematical analysis. The only way is to leave it as a black box and repeatedly feed it with different sample tables, which is costly and infeasible for our purpose of query execution time prediction. By instead representing the running time as a function of selectivities over all operators, we obtain something mathematically manipulatable and practically efficient, though we now need to address the new challenge of estimating the covariances between the selectivities.

\begin{algorithm}
\caption{Estimation of the distribution of \(t_q\)}
\textbf{Input}: \(q\), the input query; \(\mathcal{C}\), cost units calibrated offline
\textbf{Output}: \(t_q^\mathcal{X} \sim \mathcal{N}(E[t_q]; \text{Var}[t_q])\)
\begin{algorithmic}
\State \textbf{Agg} $\leftarrow$ \text{false};
\State \textbf{EstSelDistr}(\text{O});
\If {\(\text{O has left child } O_l\)}
\State \textbf{EstSelDistr}(\text{O}_l);
\EndIf
\If {\(\text{O has right child } O_r\)}
\State \textbf{EstSelDistr}(\text{O}_r);
\EndIf
\If {\text{is aggregate}}\State \text{Agg} $\leftarrow$ \text{true};\EndIf
\State Compute \(\rho_n\) and \(S_n^2\) for \(O\) using Algorithm 1;
\State \textbf{GetCostFunc}(\text{P}_q);
\For {\text{\(O \in \mathcal{P}_q\)}}
\For {\text{\(f \in \text{CostFunctions}(\text{O})\)}}
\If {\text{\(O\) is unary}}\State Collect (\text{X}, f)'s in \(\mathcal{I}_t\);\Else\State Collect (\text{X}, \text{X}, f)'s in \(\mathcal{I}_t \times \mathcal{I}_r\);
\EndIf
\EndFor
\EndFor
\EndIf
\EndIf
\State Compute \(f\) by solving the optimization problem;
\EndFor
\EndFor
\EndMain:
\State \textbf{P}_q $\leftarrow$ \text{GetQueryPlan}(\text{q})$;$
\State Run \(\mathcal{P}_q\) over the sample tables;
\State \textbf{O}_{\text{root}} $\leftarrow$ \text{GetRootOperator}(\text{P}_q);
\State \textbf{EstSelDistr}(\text{O}_{\text{root}});
\State \textbf{GetCostFunc}(\text{P}_q);
\State \text{Estimate} \(t_q^\mathcal{X} \sim \mathcal{N}(E[t_q]; \text{Var}[t_q])\) using Algorithm 3;
\end{algorithmic}
\end{algorithm}

We further summarize the procedure of computing \(\text{Var}[t_q]\) in Algorithm 3. We first compute the variances for the \(t_k\)'s, since we have shown that they are directly computable. We then collect all the paths from the leaf operators to the root and compute or bound the covariances between the operators along each path. Based on Lemma 3, these are all the pairs of operators that we need to check the covariances. Based on if \(\text{Cov}(Z, Z')\) is computable, we directly compute or provide some upper bound for it.
Algorithm 3: Estimation of the variance of \( t_q \)

| \textbf{Input:} & \( \mathcal{P}_q \), the query plan of \( q \) |
| \textbf{Output:} & \( \text{Var}[t_q] \), the estimated variance of \( t_q \) |
|\textbf{1} & \( \text{VarOps} \leftarrow 0, \text{CovOpsUb} \leftarrow 0; \) |
|\textbf{2} & \textbf{foreach} \( O_k \in \mathcal{P}_q \) \textbf{do} |
|\textbf{3} & \begin{align*} & \text{VarOps} \leftarrow \text{VarOps} + \text{Var}[t_k]; \\
& \textbf{end} \end{align*} |
|\textbf{4} & \( \mathcal{L} \leftarrow \text{GetLeafOps}(\mathcal{P}_q); \) |
|\textbf{5} & \textbf{foreach} \( L \in \mathcal{L} \) \textbf{do} |
|\textbf{6} & \( \mathcal{P} \leftarrow \text{GetPath}(L); \) |
|\textbf{7} & \textbf{foreach} \( O,O' \in \mathcal{P} \text{ s.t. } O \neq O' \) \textbf{do} |
|\textbf{8} & \begin{align*} & \textbf{foreach} \text{Cov}(Z,Z') \textbf{ do} \\
& \begin{align*} & \text{if Cov}(Z,Z') \text{ is computable then} \\
& \text{VarOps} \leftarrow \text{VarOps} + \text{ComputeCov}(O,O',Z,Z'); \\
& \text{else} \\
& \text{CovOpsUb} \leftarrow \text{CovOpsUb} + \text{UpperBoundCov}(O,O',Z,Z'); \\
& \textbf{end} \end{align*} \end{align*} |
|\textbf{10} & \textbf{end} |
|\textbf{11} & \textbf{end} |
|\textbf{12} & \( \text{Var}[t_q] \leftarrow \text{VarOps} + \text{CovOpsUb}; \) |
|\textbf{13} & \textbf{return} \( \text{Var}[t_q] \); |

C. MORE EXPERIMENTAL RESULTS

In this section we present additional experimental results.

C.1 Correlations

Table 4 reports the \( r_a \)'s (and the corresponding \( r_p \)'s) for the benchmark queries over different hardware and database settings. Here, SR stands for the sampling ratio (see Section 6.3). Values below 0.7 are highlighted. As we mentioned, these are cases where the correlations are not strong.

C.2 Distributional Distances

Table 5 reports the complete results of distributional distances for the benchmark queries (see Section 6.3). Values above 0.3 are highlighted. The closer a value is to 0, the better the proximity of two distributions is.

C.3 Comparison with Simplified Versions

Figure 10 presents more results on comparison of the four alternatives discussed in Section 6.3 over skewed databases for the TPCH queries. The observations are similar to that over uniform databases as presented in Section 6.3.

C.4 Sampling Overhead

Figure 11 presents the complete experimental results for the relative overhead of running the queries over the sample tables, which were omitted in Section 6.4.

C.5 Selectivity Estimates

Given a query \( q \), the quality of the estimated distribution of \( t_q \) depends on a number of factors such as the accuracy of the distributions of the \( c \)'s and the \( X \)'s, the quality of the approximated cost functions, and the closeness of the upper bounds of the covariances to the actual values. Note that how well we could estimate the potential errors in selectivity estimates plays a crucial role here, for it directly impacts the accuracy of the estimated distributions of the \( X \)'s, which further impacts the accuracy of the approximated cost functions as well as the estimated covariances. Therefore, we further studied the correlations between the estimated and actual errors in selectivity estimates. Since the selectivities are modeled as Gaussian variables, we again focus on measuring the correlations between the standard deviations of the Gaussian distributions and the actual errors in the selectivity estimates, as what we did in Section 6.3 for the distributions of the \( t_q \)'s. For this sake, we examined the correlations for the selective operators (i.e., selections and joins) of each query in the benchmarks under different hardware and database settings. Table 6 presents the results.

We observe that the correlations are not as good as that of the \( t_q \)'s in Table 4. In particular, there seems to be no linear correlations for the SELJOIN and TPCH queries by examining the corresponding \( r_p \)'s. One possible reason for this phenomenon is that the actual errors are usually too small. To verify this, in Table 7 we present the correlations between the estimated and the actual selectivities, and in Table 8 we compute the mean relative errors in the selectivity estimates, where the relative error of an estimated selectivity \( \rho \) is defined as \( \frac{\mid\rho - \rho\rangle}{\rho} \). We find that the \( r_p \)'s between the estimated and actual selectivities are almost 1 for almost all the cases we tested, which suggests a very strong linear dependency. As a case study, in Figure 12 we show the scatter plots of the MICRO, SELJOIN, and TPCH queries over skewed 1GB database on PC1 with SR = 0.05. We can see that the estimated selectivities are almost the same as the ground truths. On the other hand, the average relative errors are usually below 20% according to Table 8. Note that our sampling based method cannot be very effective when the errors are too small unless we can have very large number of samples. This is because we estimate the variances of the distributions based on the observed variances in the samples. Since the samples are taken randomly, different batches of samples may present different sample...
Table 4: \( r_s \) (\( r_p \)) of the benchmark queries over different hardware and database settings (values below 0.7 are highlighted)

<table>
<thead>
<tr>
<th>Sampling Ratio</th>
<th>MICRO</th>
<th>SELJOIN</th>
<th>TPCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform TPC-H 1GB Database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.2312 (0.9830)</td>
<td>0.9400 (0.5691)</td>
<td>0.7554 (0.8989)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.9381 (0.9875)</td>
<td>0.9813 (0.7904)</td>
<td>0.8551 (0.9724)</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9415 (0.9862)</td>
<td>0.9740 (0.8252)</td>
<td>0.7209 (0.7571)</td>
</tr>
<tr>
<td>Skewed TPC-H 1GB Database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.9418 (0.9753)</td>
<td>0.9827 (0.9236)</td>
<td>0.8545 (0.5575)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.9435 (0.9762)</td>
<td>0.9838 (0.9130)</td>
<td>0.8495 (0.8656)</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9431 (0.9765)</td>
<td>0.9840 (0.9168)</td>
<td>0.8546 (0.6644)</td>
</tr>
<tr>
<td>Uniform TPC-H 10GB Database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.9397 (0.9518)</td>
<td>0.9853 (0.9549)</td>
<td>0.9660 (0.8263)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.9379 (0.9675)</td>
<td>0.9855 (0.9536)</td>
<td>0.9904 (0.9567)</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9383 (0.9760)</td>
<td>0.9853 (0.9539)</td>
<td>0.9708 (0.8574)</td>
</tr>
<tr>
<td>Skewed TPC-H 10GB Database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.9674 (0.9665)</td>
<td>0.9819 (0.9830)</td>
<td>0.9636 (0.9896)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.9669 (0.9812)</td>
<td>0.9841 (0.9831)</td>
<td>0.9650 (0.9519)</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9675 (0.9905)</td>
<td>0.9840 (0.9830)</td>
<td>0.9663 (0.9580)</td>
</tr>
</tbody>
</table>

Table 5: \( \overline{d}_{\text{MC}} \) of the benchmark queries over different hardware and database settings (values above 0.3 are highlighted)

<table>
<thead>
<tr>
<th>Sampling Ratio</th>
<th>MICRO</th>
<th>SELJOIN</th>
<th>TPCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform TPC-H 1GB Database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.5873</td>
<td>0.6235</td>
<td>0.2228</td>
</tr>
<tr>
<td>0.05</td>
<td>0.2728</td>
<td>0.3885</td>
<td>0.1563</td>
</tr>
<tr>
<td>0.1</td>
<td>0.2312</td>
<td>0.3236</td>
<td>0.1170</td>
</tr>
<tr>
<td>Skewed TPC-H 1GB Database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.1665</td>
<td>0.2532</td>
<td>0.0766</td>
</tr>
<tr>
<td>0.05</td>
<td>0.1657</td>
<td>0.2532</td>
<td>0.0765</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1622</td>
<td>0.2532</td>
<td>0.0722</td>
</tr>
<tr>
<td>Uniform TPC-H 10GB Database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.1665</td>
<td>0.2532</td>
<td>0.0766</td>
</tr>
<tr>
<td>0.05</td>
<td>0.1657</td>
<td>0.2532</td>
<td>0.0765</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1622</td>
<td>0.2532</td>
<td>0.0722</td>
</tr>
<tr>
<td>Skewed TPC-H 10GB Database</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.1665</td>
<td>0.2532</td>
<td>0.0766</td>
</tr>
<tr>
<td>0.05</td>
<td>0.1657</td>
<td>0.2532</td>
<td>0.0765</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1622</td>
<td>0.2532</td>
<td>0.0722</td>
</tr>
</tbody>
</table>

Figure 11: Relative overhead of benchmark queries

variances although the variation should be small due to the strong consistency property. However, if the errors in selectivity estimates are already very small, then the small variation in sample variances now cannot be negligible. The impact on the correlations due to this variation is unpredictable since this variation is totally random. But the correlations here seem to be not very important, since based on the small variances we can still infer that the errors are small although we have no idea of the correlations. In Table 9 we further present the correlations when the relative errors are above 0.2. We find that now the correlations are much better. This implies that the estimated errors are strongly correlated with the actual errors in selectivity estimates when the errors are relatively large.
<table>
<thead>
<tr>
<th>SR</th>
<th>MICRO</th>
<th>SELJOIN</th>
<th>TPCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PC1</td>
<td>PC2</td>
<td>PC1</td>
</tr>
<tr>
<td>0.01</td>
<td>0.8127 (0.8730)</td>
<td>0.9245 (0.9621)</td>
<td>0.7731 (0.2294)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.8670 (0.7109)</td>
<td>0.8497 (0.8542)</td>
<td>0.7818 (0.0470)</td>
</tr>
<tr>
<td>0.1</td>
<td>0.8116 (0.7556)</td>
<td>0.8246 (0.8216)</td>
<td>0.7907 (0.0276)</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8102 (0.6451)</td>
<td>0.8897 (0.6541)</td>
<td>0.8065 (-0.0034)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.9107 (0.5566)</td>
<td>0.9274 (0.7311)</td>
<td>0.7940 (0.0056)</td>
</tr>
<tr>
<td>0.4</td>
<td>0.8654 (0.4994)</td>
<td>0.8988 (0.7034)</td>
<td>0.7923 (-0.0314)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SR</th>
<th>MICRO</th>
<th>SELJOIN</th>
<th>TPCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PC1</td>
<td>PC2</td>
<td>PC1</td>
</tr>
<tr>
<td>0.01</td>
<td>0.8736 (0.7780)</td>
<td>0.8938 (0.8807)</td>
<td>0.6172 (-0.0178)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.9007 (0.7815)</td>
<td>0.9066 (0.8129)</td>
<td>0.6274 (0.0265)</td>
</tr>
<tr>
<td>0.1</td>
<td>0.7748 (0.2938)</td>
<td>0.9320 (0.8578)</td>
<td>0.6347 (-0.0306)</td>
</tr>
<tr>
<td>0.2</td>
<td>0.7566 (0.5545)</td>
<td>0.8772 (0.6146)</td>
<td>0.6360 (-0.0202)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7880 (0.4806)</td>
<td>0.9137 (0.5158)</td>
<td>0.6505 (-0.0218)</td>
</tr>
<tr>
<td>0.4</td>
<td>0.8063 (0.1580)</td>
<td>0.8722 (0.6483)</td>
<td>0.6808 (-0.0520)</td>
</tr>
</tbody>
</table>

Table 6: $r_s(r_p)$ between the estimated and actual errors in selectivity estimates (values below 0.7 are highlighted)

<table>
<thead>
<tr>
<th>SR</th>
<th>MICRO</th>
<th>SELJOIN</th>
<th>TPCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PC1</td>
<td>PC2</td>
<td>PC1</td>
</tr>
<tr>
<td>0.01</td>
<td>0.9365 (0.6938)</td>
<td>0.8742 (0.8389)</td>
<td>0.6187 (0.0487)</td>
</tr>
<tr>
<td>0.02</td>
<td>0.9080 (0.6594)</td>
<td>0.8781 (0.7153)</td>
<td>0.6524 (0.0069)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.9004 (0.2230)</td>
<td>0.9208 (0.6030)</td>
<td>0.6366 (-0.0132)</td>
</tr>
<tr>
<td>0.1</td>
<td>0.8733 (0.2993)</td>
<td>0.7862 (0.3673)</td>
<td>0.6696 (-0.0470)</td>
</tr>
</tbody>
</table>

Table 7: $r_s(r_p)$ between the estimated and actual selectivities (values below 0.7 are highlighted)
Table 8: Relative errors in the selectivity estimates of the benchmark queries (values above 0.2 are highlighted)

<table>
<thead>
<tr>
<th>SR</th>
<th>MICRO</th>
<th>SELJOIN</th>
<th>TPCH</th>
<th>MICRO</th>
<th>SELJOIN</th>
<th>TPCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PC1</td>
<td>PC2</td>
<td>PC1</td>
<td>PC2</td>
<td>PC1</td>
<td>PC2</td>
</tr>
<tr>
<td>Uniform TPC-H 1GB Database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.1328</td>
<td>0.2299</td>
<td>0.4678</td>
<td>0.4216</td>
<td>0.5349</td>
<td>0.4738</td>
</tr>
<tr>
<td>0.05</td>
<td>0.0340</td>
<td>0.0551</td>
<td>0.1824</td>
<td>0.2946</td>
<td>0.3121</td>
<td>0.2212</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0306</td>
<td>0.0318</td>
<td>0.1586</td>
<td>0.1484</td>
<td>0.1988</td>
<td>0.1967</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0197</td>
<td>0.0122</td>
<td>0.1134</td>
<td>0.0836</td>
<td>0.1017</td>
<td>0.1590</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0144</td>
<td>0.0132</td>
<td>0.0577</td>
<td>0.0422</td>
<td>0.0583</td>
<td>0.0734</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0146</td>
<td>0.0200</td>
<td>0.0389</td>
<td>0.0371</td>
<td>0.0585</td>
<td>0.0534</td>
</tr>
</tbody>
</table>

Skewed TPC-H 1GB Database

| Uniform TPC-H 10GB Database |       |         |       |       |         |       |
| 0.01 | 0.0381 | 0.0492  | 0.3162 | 0.3396 | 0.4475  | 0.5894 |
| 0.02 | 0.0342 | 0.0241  | 0.2419 | 0.2344 | 0.3439  | 0.3533 |
| 0.05 | 0.0147 | 0.0101  | 0.1491 | 0.1713 | 0.2018  | 0.2130 |
| 0.1  | 0.0068 | 0.0096  | 0.1047 | 0.0800 | 0.1291  | 0.1499 |

Skewed TPC-H 10GB Database

Figure 12: A case study of the correlations between the estimated and actual selectivities

Table 9: $r_s$ ($r_p$) of selectivity estimates with relative errors above 0.2 (values below 0.7 are highlighted)

<table>
<thead>
<tr>
<th>SR</th>
<th>MICRO</th>
<th>SELJOIN</th>
<th>TPCH</th>
<th>MICRO</th>
<th>SELJOIN</th>
<th>TPCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PC1</td>
<td>PC2</td>
<td>PC1</td>
<td>PC2</td>
<td>PC1</td>
<td>PC2</td>
</tr>
<tr>
<td>Uniform TPC-H 1GB Database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>0.6980</td>
<td>(1.0000)</td>
<td>0.8223</td>
<td>(0.9952)</td>
<td>0.9101</td>
<td>(0.8918)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.9920</td>
<td>(1.0000)</td>
<td>0.9983</td>
<td>(1.0000)</td>
<td>0.8423</td>
<td>(0.9536)</td>
</tr>
<tr>
<td>0.1</td>
<td>1.0000</td>
<td>(0.9998)</td>
<td>1.0000</td>
<td>(0.9942)</td>
<td>0.8696</td>
<td>(0.9955)</td>
</tr>
<tr>
<td>0.2</td>
<td>1.0000</td>
<td>(1.0000)</td>
<td>N/A (N/A)</td>
<td>0.9747</td>
<td>(0.9706)</td>
<td>0.9688</td>
</tr>
<tr>
<td>0.3</td>
<td>N/A (N/A)</td>
<td>0.9850</td>
<td>(0.9751)</td>
<td>0.9884</td>
<td>(0.9754)</td>
<td>0.9696</td>
</tr>
<tr>
<td>0.4</td>
<td>N/A (N/A)</td>
<td>0.9900</td>
<td>(0.9994)</td>
<td>0.9693</td>
<td>(0.9847)</td>
<td>0.9688</td>
</tr>
</tbody>
</table>

Skewed TPC-H 10GB Database

| Uniform TPC-H 10GB Database |       |         |       |       |         |       |
| 0.01 | 0.7253 | (0.9090) | 0.9067 | (1.0000) | 0.7964 | (0.0291) |
| 0.05 | 0.9903 | (1.0000) | 0.9728 | (0.9984) | 0.8863 | (1.0000) |
| 0.1  | 0.9905 | (1.0000) | 1.0000 | (0.9981) | 0.9081 | (0.9541) |
| 0.2  | 1.0000 | (0.9847) | 1.0000 | (0.9803) | 0.9966 | (1.0000) |
| 0.3  | 1.0000 | (0.9624) | 1.0000 | (0.9999) | 0.9937 | (1.0000) |
| 0.4  | 1.0000 | (0.9999) | 1.0000 | (0.9966) | 0.9954 | (1.0000) |

Skewed TPC-H 10GB Database

| Uniform TPC-H 10GB Database |       |         |       |       |         |       |
| 0.01 | 0.9441 | (0.9929) | 0.9950 | (1.0000) | 0.7533 | (0.9824) |
| 0.05 | 0.9818 | (0.9637) | 1.0000 | (1.0000) | 0.7937 | (0.9996) |
| 0.1  | N/A (N/A) | 0.8029 | (0.8149) | 0.7450 | (0.7202) | 0.8093 | (0.7995) |

Skewed TPC-H 10GB Database

| Uniform TPC-H 10GB Database |       |         |       |       |         |       |
| 0.01 | 0.9805 | (1.0000) | 0.8878 | (0.9984) | 0.8917 | (0.7953) |
| 0.02 | 0.9759 | (0.9280) | 0.9647 | (0.9237) | 0.9617 | (0.9994) |
| 0.05 | 0.8810 | (0.9741) | 1.0000 | (0.9991) | 0.7759 | (0.9970) |
| 0.1  | 1.0000 | (1.0000) | 1.0000 | (0.9999) | 0.9873 | (0.9999) |