Robustness Metrics for Relational Query Execution Plans

Florian Wolf
TU Ilmenau, SAP SE
florian.wolf@tu-ilmenau.de

Michael Brendle
University of Konstanz, SAP SE
michael.brendle@uni.kn

Norman May
SAP SE
norman.may@sap.com

Paul R. Willems
SAP SE
paul.willems@sap.com

Kai-Uwe Sattler
TU Ilmenau
kus@tu-ilmenau.de

Michael Grossniklaus
University of Konstanz
michael.grossniklaus@uni.kn

ABSTRACT

The quality of query execution plans in database systems determines how fast a query can be executed. It has been shown that conventional query optimization still selects sub-optimal or even bad execution plans, due to errors in the cardinality estimation. Although cardinality estimation errors are an evident problem, they are in general not considered in the selection of query execution plans. In this paper, we present three novel metrics for the robustness of relational query execution plans w.r.t. cardinality estimation errors. We also present a novel plan selection strategy that takes both, estimated cost and estimated robustness into account, when choosing a plan for execution. Finally, we share the results of our experimental comparison between robust and conventional plan selection on real world and synthetic benchmarks, showing a speedup of at most factor 3.49.

PVLDB Reference Format:
Florian Wolf, Michael Brendle, Norman May, Paul R. Willems, Kai-Uwe Sattler, and Michael Grossniklaus. Robustness Metrics for Relational Query Execution Plans. *PVLDB*, 11(11): 1360-1372, 2018.
DOI: https://doi.org/10.14778/3236187.3236191

1. INTRODUCTION

The goal of query optimization in database systems is to find a good query execution plan among the set of equivalent plans for a given declarative query, according to a cost model. Although query optimization is a well-studied problem with numerous approaches being proposed and developed since Selinger's seminal work [23], finding a good plan is still a challenge, even for mature commercial systems. Typically, two major problems arise in this context: (1) significantly increased query execution times, if the chosen query execution plan turns out to be sub-optimal or even bad, and (2) unpredictable execution time behavior due to small changes in the database, which can cause the selection of a fundamentally different query execution plan with a very different execution time.

Proceedings of the VLDB Endowment, Vol. 11, No. 11

DOI: https://doi.org/10.14778/3236187.3236191

Both are problems of *robustness*, which has become an important research question in query processing. It is discussed in multiple Dagstuhl seminars on *Robust Query Processing*, organized by Graefe and colleagues [10, 11, 12]. Although robust query processing has several aspects ranging from query planning to execution and scheduling, both problems share a common issue related to query robustness: errors in cardinality estimation as a core component of a cost model.

Recent work by Leis et al. [17, 18] has shown that rather than the cost model, cardinality estimation is the weak spot. Even simple cost models result in a strong correlation between true cost and query execution time. Lohman quantifies the performance impact of the cost model to at most 30% [20]. In contrast, errors in cardinality estimation are in principle unbounded, and studies have shown up to six orders of magnitude estimation errors [16, 17, 18].

In a nutshell, the root causes for cardinality estimation errors are wrong assumptions on: (1) data distribution, (2) column correlation, (3) join relationship, and (4) inaccuracy of statistics. Although the value frequencies in real-world data are frequently skewed, a usual assumption in cardinality estimation is uniform data distribution. Also, the assumption of Attribute Value Independence (AVI) for the correlation between columns in a table is not generally valid [20]. For joins, some cardinality estimators assume the principle of inclusion [18], which is only guaranteed to hold for foreign key joins. In the presence of data modifications, statistics for cardinality estimation can become stale or too expensive to be updated at each transaction. As a result the accuracy of statistics is often lower than expected.

Several approaches have been proposed to improve cardinality estimation. Histograms or sampling can handle different data distributions better than assuming uniformity, and column group statistics can improve the precision for column correlation in a table. However, cardinality estimation is still the main problem in query optimization [17, 18].

Although the issues of cardinality estimation are evident, the majority of query optimizers still chooses the estimated cheapest plan based on the cost model as optimal plan. Potential cardinality estimation errors are not taken into account when choosing a plan. Lohman advocated that "*robust and adaptable query plans are superior to optimal ones*" [19]. In this paper, we present novel metrics for the robustness of query execution plans towards estimation errors. They can assign a numeric value for the robustness of a plan, which can be considered next to the estimated cost in the selection of a plan. Compared to competing approaches for robustness metrics [1, 2], we can assign robustness values

independent of other query execution plans. In summary, our contributions are:

- a formal problem description and consistency requirements for plan robustness metrics (Section 3),
- three new metrics to quantify the robustness of relational query execution plans, supporting all kinds of operators, operator implementations, query execution plan trees, and monotonically increasing and differentiable cost functions (Section 4),
- a new plan selection strategy for query processing based on our plan robustness metrics (Section 5), and
- an experimental evaluation for runtime and robustness of our plan selection strategy using synthetic and real world data benchmarks (Section 6).

2. RELATED WORK

Open research questions in robust query processing are regularly discussed in *Dagstuhl seminars* organized by Graefe and colleagues [10, 11, 12]. One approach to robust query processing is robust plan selection as classified in a recent survey [25]. The design space for robust plan selection strategies has similarities to the design space of a conventional query optimization. We argue that the design space of robust plan selection strategies has three dimensions: (1) online selection vs. offline analysis, (2) robust plan candidates, and (3) robust plan selection. A robust plan can either be selected at optimization time (online), or identified in a more expensive offline analysis. The set of candidates for robust plans can be limited, e.g., to plans that are only optimal, for certain cardinalities [7, 15], plans that have costs close to the estimated optimal plan [1], or plans with a certain tree structure, e.g., only left-deep trees [2]. There are numerous approaches to choose the most robust plan in the set of candidates, e.g., a plan that is optimal for multiple cardinality and selectivity combinations [4], or the most robust plan according to a robustness metric [1, 2].

Robust Plan Diagram Reduction [7] and Plan Bouquets [8] reduce parametric optimal sets of plans (POSP) [14]. Robust Plan Diagram Reduction is a graphical plan space analysis that identifies robust plan clusters. Plan Bouquets iteratively explore different plans through execution, give a formal upper bound for execution time compared to the fastest plan, and do not rely on cardinality estimation. Enumerating POSPs and identifying the Plan Diagram or the Plan Bouquets causes a very high pre-calculation effort, and is not feasible with updates and ad-hoc queries. Our approach is based on estimations, and enables the specification of an upper bound for cost w.r.t. the estimated optimal plan. Due to the small pre-calculation effort, it can be applied at optimization time, and supports updates and ad-hoc queries.

Risk Score [15] is a metric for plan robustness, and indicates how fragile a plan during different execution conditions is. Since different execution times are necessary, a Risk Score cannot be predicted during optimization time. The robust plan candidates set is again limited to the POSP.

All following approaches perform online selection. Proactive Re-Optimization [4] searches the optimal plan for the estimated and two heuristically chosen cardinalities for each cardinality estimate. From the three plans, it tries to identify the optimal, a robust, or a switchable plan. If no such plan exists, it triggers a runtime re-optimization.

Robust Cardinality Estimation [3] instead uses random sampling to generate a probability density function for operator output cardinality. Based on the probability density function and a user defined risk level for the probability density function, it estimates the maximum output cardinality of an operator, and searches the optimal plan for it.

In contrast to Proactive Re-Optimization and Robust Cardinality Estimation, our approach defines a specific robustness value for different plans that allows to compare two plans w.r.t. their robustness. We also consider non-optimal plans in the robust plan candidates set, since a robust plan does not require optimality for certain cardinalities.

Minmax Regret Rule [2] is similar to Proactive Re-Optimization, but considers more plans and has a different robust plan selection criteria. It compares the costs of the plans at different cardinalities. Selected is the plan that has the smallest maximum cost difference to the optimal plans, over all cardinalities. Due to the increased number of plans, it is limited to left-deep trees. Since this limitation excludes possible robust plans, we consider all plan trees in our work.

An extension to the Minmax Regret Rule are Cost-Stable Plans [1], which choose the plan with the smallest average cost difference to the optimal plans, over all cardinalities. In addition, it limits the number of plans, e.g., by early pruning of outliers with a large cost difference to the optimal plan.

Due to its efficiency, our approach can be applied at optimization time (online selection). Compared to other approaches, we are not limited to certain tree structures [2] or plans that are optimal for some cardinalities [3, 4, 7, 15]. We limit the number of robust plan candidates to the cheapest plans encountered during the initial query optimization. In contrast to competing approaches [1, 2], we can assign robustness values independent of other query execution plans. Finally, we define a robustness metric that works with classical single point estimation and is not bound to more expensive cardinality estimation techniques [1, 2, 3, 4].

3. PROBLEM STATEMENT

Due to estimation errors, the estimated optimal plan chosen by conventional query optimizers frequently fails to be the fastest plan. We argue that choosing a robust plan can result in faster query execution times in the presence of cardinality estimation errors. We formalize the problem of finding a plan that is robust w.r.t. estimation errors, denoting the estimated cardinality as $\hat{f}$ and the estimated cost as $\hat{c}$.

Definition 1. The **true cardinality** $\mathring{f}$ is the exact cardinality for an edge in the query execution plan, collected during execution. The **true cost** $\mathring{c}$ is calculated using the true cardinalities $\mathring{f}$ instead of the estimated cardinalities $\hat{f}$.

Definition 2. The **cost error factor** c_{err} is the absolute quotient of estimated and true cost.

$$c_{\text{err}} = \begin{cases} \mathring{c}/\hat{c} & \text{if } \mathring{c} \geq \hat{c} \\ \hat{c}/\mathring{c} & \text{otherwise} \end{cases}$$

Definition 3. The **most robust plan** is the plan with the smallest cost error factor c_{err} within the set of robust plan candidates.

Since true costs $\mathring{c}$ are unknown at optimization time, the cost error factor c_{err} cannot be calculated. Therefore, we have to define a robustness metric.

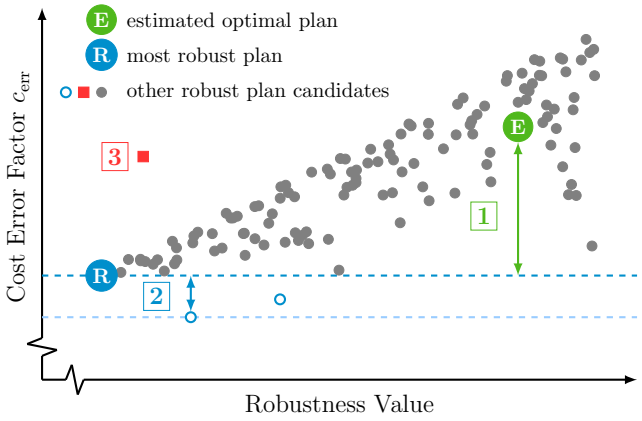


Figure 1: Illustrations of consistency requirements for robustness metrics, showing candidate plans with their assigned robustness value and their c_{err}.

Definition 4. A **robustness metric** assigns a robustness value to each robust plan candidate. Ideally, the robustness value is an approximation for the upper bound of c_{err}.

Figure 1 illustrates the behavior of an ideal robustness metric. The robustness value assigned by the robustness metric is denoted on the x-axis. The y-axis denotes the cost error factor c_{err} (see Definition 2). Furthermore, Figure 1 shows all robust plan candidates with an assigned robustness value and their cost error factor c_{err}. The estimated optimal plan is highlighted as E. The most robust plan according to the robustness metric, i.e., the leftmost plan, is depicted as R. We argue that an ideal robustness metric should fulfill the following three consistency requirements.

1 **Cost Error Factor Improvement:** Compared to the estimated optimal plan the most robust plan according to the robustness metric should always achieve a smaller cost error factor c_{err}.

2 **Cost Error Factor Dominance:** The most robust plan according to the robustness metric dominates all robust plan candidates w.r.t. the cost error factor c_{err}. This means there should be no plan with a smaller cost error factor c_{err} than the most robust plan, e.g., the empty circle ○ plans in Figure 1.

3 **Correlated Cost Error Factor Limit:** The robustness metric should give an upper bound for the cost error factor c_{err} of a plan. Plans with a large robustness value can have a large c_{err}, and plans with a small robustness value should have a small c_{err}. Plans, such as the square ■ plan in Figure 1 indicate a suboptimal robustness metric, because the metric classified the plan to be much more robust than it is. The upper bound of c_{err} should be proportional to the robustness value, but c_{err} itself does not have to be proportional to the robustness value, because the cardinality estimations can always be precise and result in a smaller c_{err}.

In practice, there is a trade-off between plan robustness and query execution time. On the one hand, a robust plan is less sensitive to estimation errors, but not necessarily fast. On the other hand, a fast plan is not necessarily robust. In Section 5, we define a robust plan selection strategy that balances plan robustness and query execution time.

3.1 Robust Plan Example

We consider Q17 of the Join Order Benchmark (JOB) [17] as a pure join query with a filter on all `movie_id` columns (cf. Section 6). Of course, we support all kinds of operators and operator implementations. In this example we use the C_{out} [22] cost function, which accumulates the operator output cardinalities:

$$C_{out}(T) = \begin{cases} |R| & \text{if } T = R \\ |T| + C_{out}(T_1) + C_{out}(T_2) & \text{if } T = T_1 \bowtie T_2 \end{cases}$$

C_{out} has a strong correlation to our query execution engine [24]. Next to C_{out}, we support every kind of monotonically increasing and differentiable cost function such as C_{mm} [18], which we use for the experimental evaluation. It is an extension of C_{out} and considers different operators and operator implementations. Next, we define the estimated selectivity $\hat{s}$, the true selectivity $\mathring{s}$, the absolute cardinality error Δf, the absolute cost error Δc, and the q-error [21], i.e., the absolute quotient of estimated and true cardinality.

$$\Delta f = \mathring{f} - \hat{f} \qquad \Delta c = \mathring{c} - \hat{c}$$

$$q\text{-error} = \begin{cases} \mathring{f}/\hat{f} & \text{if } \mathring{f} \geq \hat{f} \\ \hat{f}/\mathring{f} & \text{otherwise} \end{cases}$$

Figure 2 shows the query execution plan for the estimated optimal plan of JOB Query 17, identified by our query optimizer. We argue in Section 6 that our join optimizer's estimated optimal plan choice is very similar to the choice of popular free and commercial systems, due to its enumeration algorithm, cardinality estimator and cost function. Every edge in the query execution plan represents an intermediate result with estimated and true statistics. The true statistics of the final edge Ⓓ shows that the true cardinality of the estimated optimal plan is underestimated by a factor of 20.27 (q-error), and the true costs by a factor of 3.03 (c_{err}). In more detail, we see that the cardinality estimator underestimates two edges in the query execution plan, i.e., the dashed edges Ⓐ and Ⓒ. The first join Ⓐ is a m:n join between `movie_keyword` and `movie_companies`. The estimated cardinality on the outgoing edge of this join is 25,305. After executing this plan, it turns out that this is an expanding join, and the cardinality was underestimated, due to missing information about the data distribution. The true cardinality is 179,425, which results in a q-error of 7.09 and in a c_{err} of 2.22. The second underestimation occurs in the join between subtree Ⓑ and the `cast_info` table. Beside a foreign key join, there are two m:n joins involved. Again, this is an expanding join and the output cardinality is underestimated: the estimated output cardinality is 347,793, but the true cardinality is 7,050,333. Accordingly, the q-error is 20.27 and the c_{err} is 4.06. All other plan edges are estimated correctly, since the q-error is not growing w.r.t. the child edges. The reason is that all those joins are foreign key joins, for which the cardinality estimation is more precise.

Figure 3 shows the estimated robust plan for JOB Q17, chosen by our approach. While the estimated optimal plan in Figure 2 has smaller estimated cost (6,908,427 compared to 8,018,758), the plan chosen by our approach has a lower execution time in the presence of cardinality estimation errors. The major difference between the estimated optimal plan (Figure 2) and the estimated robust plan (Figure 3) is the deferred execution of m:n joins. Therefore, the first cardinality estimation error does not occur at the first join, as in the estimated optimal plan (see Ⓐ in Figure 2). In contrast,

there are no cardinality estimation errors in the subtrees Ⓗ and Ⓔ of the estimated robust plan. Their foreign key joins are estimated correctly. The first underestimation occurs when subtree Ⓔ is joined with `cast_info` (see Ⓕ). While the estimated cardinality is 705,030, the true cardinality is 1,646,933. Hence, the q-error is 2.34 and the c_{err} is 1.41. The second underestimation occurs at the final join between Ⓖ and Ⓗ, where the last two m:n joins are involved. As a result, the q-error is 20.30 and the c_{err} is 2.07.

Comparing the two plans, shows that the c_{err} of the estimated robust plan (2.07) is smaller than of the estimated optimal plan (3.03). Also, the true cost of the estimated robust plan (16,605,629) is smaller than of the estimated optimal plan (20,929,987). Therefore, the estimated robust plan (475 ms) achieves a speedup of factor two compared to the estimated optimal plan (995 ms). In Section 6, we show more complex queries with larger speedups in the presence of cardinality estimation errors.

4. ROBUSTNESS METRICS

In this section we answer the question: can we define robustness metrics that quantify the robustness for query execution plans before executing the plans? After running the query, robustness for a query execution plan or a sub-plan can be quantified by the q-error or the c_{err}. In order to quantify the robustness of a plan before execution, defining the *Parametric Cost Function (PCF)* is the first building block for our robustness metrics. We previously used PCFs as a building block in the calculation of optimality ranges [24].

Definition 5. A **Parametric Cost Function (PCF)** is the cost of a query execution plan or sub-plan, modeled as function of one cost parameter.

Figure 4 shows the PCF modeled as function of cardinality on a single edge in the plan for a volatile (PCF_{vol}) and for a robust plan (PCF_{rob}). The cardinality of the edge is denoted on the x-axis and the cost of a plan on the y-axis. It also shows the estimated cardinality $\hat{f}$ and the true cardinality $\mathring{f}$. Furthermore, it shows the estimated cost $\hat{c}$ and the true cost $\mathring{c}$ for both plans. Since a robust plan is not necessarily optimal at $\hat{f}$, a volatile plan might have smaller estimated costs $\hat{c}$. In the presence of estimation errors, the true cost of a volatile plan can rapidly increase or decrease. In contrast, the true cost for a robust plan are close to the estimated cost in the presence of cardinality estimation errors, i.e., a more moderate slope of PCF_{rob} compared to PCF_{vol}. Therefore, the slopes of PCFs around the estimated cardinality indicate the sensitivity of a plan towards estimation errors. If the true cardinality is underestimated, as $\mathring{f}$ in Figure 4, picking the robust plan will also lead to runtime speedups.

Conventional query optimizers select the plan with the smallest estimated cost, but do not consider the cost behavior in the presence of estimation errors. Consequently, the estimated optimal plan is not necessarily a robust plan. We argue that considering the cost behavior, i.e., the slopes of a PCF, in the plan selection, results in identifying more robust plans. Modeling the C_{out} cost of a plan as a function of one cost parameter for example, results in a linear PCF, i.e., a PCF with the same slope at every cardinality. In contrast, using a cost function other than C_{out} can result in a non-linear PCF. We support non-linear PCFs that are monotonically increasing and differentiable, i.e., have no jumps and there is a slope value at each point.

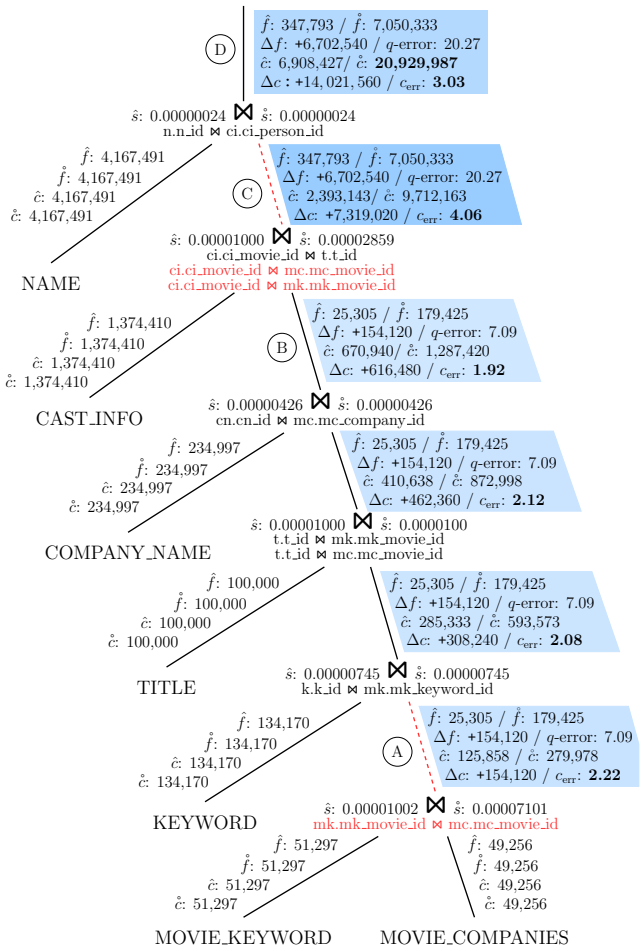


Figure 2: Estimated optimal plan for JOB Query 17, chosen by conventional plan selection strategy.

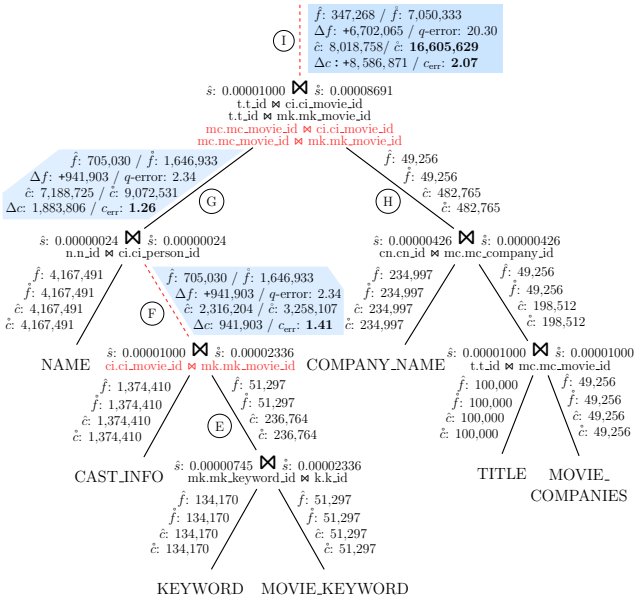


Figure 3: Estimated robust plan for JOB Query 17, chosen by our approach.

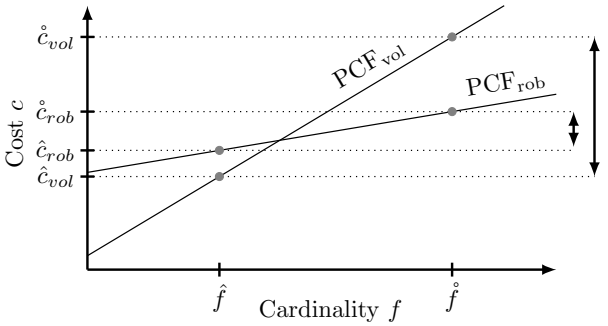


Figure 4: Cost behavior of a volatile plan and a robust plan in the presence of estimation errors.

Next, we give an example for the calculation of a PCF. We consider the estimated robust plan P_{rob} for Query 17 of the Join Order Benchmark in Figure 3. For the C_{out} cost function and the given statistics, P_{rob} has estimated costs of $8,018,758$. We assume that the output cardinality of edge Ⓕ in Figure 3 is not $705,030$ but an arbitrary value. Let us denote this variable as $f_{\text{CI,K,MK}}$ for the output cardinality of joining `cast_info` (CI), `keyword` (K), and `movie_keyword` (MK). Let us model the C_{out} costs of P_{rob} as a PCF on the variable $f_{\text{CI,K,MK}}$, i.e., not set $f_{\text{CI,K,MK}} = 705,030$ but leave it as parameter when calculating the C_{out} costs of P_{rob}:

$$\begin{aligned} C_{out}(P_{rob}) &= f_{\text{CI,K,MK}} + \hat{f}_{\text{N,CI,K,MK}} + \hat{f}_{\text{N,CI,K,MK,T,MC,CN}} \\ &\quad + \hat{f}_{\text{K}} + \hat{f}_{\text{MK}} + \hat{f}_{\text{K,MK}} + \hat{f}_{\text{CI}} + \hat{f}_{\text{N}} \\ &\quad + \hat{f}_{\text{T}} + \hat{f}_{\text{MC}} + \hat{f}_{\text{T,MC}} + \hat{f}_{\text{CN}} + \hat{f}_{\text{T,MC,CN}} \\ &= 2.49 \cdot f_{\text{CI,K,MK}} + 6,261,430 \end{aligned} \quad (1)$$

We see that for each output tuple on the edge $f_{\text{CI,K,MK}}$ the total cost of P_{rob} increases by 2.49. While $f_{\text{CI,K,MK}}$ is the deepest edge containing a m:n join for P_{rob}, the edge Ⓐ, denoted as $f_{\text{MK,MC}}$, is the deepest edge containing a m:n join of the estimated optimal plan P_{opt} in Figure 2. In order to consider the sensitivity of $f_{\text{MK,MC}}$, we calculate the costs of P_{opt} as a PCF on the variable $f_{\text{MK,MC}}$:

$$C_{out}(P_{opt}) = 31.49 \cdot f_{\text{MC,MK}} + 6,111,621 \quad (2)$$

Consequently, one additional tuple for $f_{\text{MK,MC}}$ increases the total cost of P_{opt} by 31.49. Therefore, the edge $f_{\text{MK,MC}}$ in P_{opt} has a steeper slope than the edge $f_{\text{CI,K,MK}}$ in P_{rob}.

4.1 Cardinality-Slope Robustness Metric

To define a robustness metric on PCFs for an online selection approach, we argue that the following design considerations have to be taken into account: (1) calculation effort, (2) potential cardinality estimation errors for different types of operators, and (3) potential propagation of cardinality estimation errors. A low calculation effort is mandatory for an online selection approach. The risk of cardinality estimation errors for different types of operators has to be considered in the robustness metric, since it has been shown that the precision of statistical models varies for different types of operators [5, 17, 20]. Finally, it has to be considered that cardinality estimation errors on deep edges (greater depth in the plan tree [6]) can be propagated to the cardinality estimations on higher edges (smaller depth in the plan tree). Consequently, cardinality estimation errors on deep edges can have a stronger impact on c_{err} compared to higher edges.

First, we denote a query execution plan $P = (O_P, E_P)$, where O_P is the set of operators and E_P the set of edges. We take the PCFs for all edges in a query execution plan into account. The next building block for the cardinality-slope robustness metric is the definition of a *cardinality-slope value* for an edge $e \in E_P$ based on a PCF of cardinality f on e.

Definition 6. The **cardinality-slope value $\delta_{f,e}$** for an edge $e \in E_P$ is the slope of $PCF_{f,e}$ at the estimated cardinality $\hat{f}$, where $PCF_{f,e}$ is the PCF that models the cost of a plan P as a function of cardinality f on e.

In theory, estimation errors can occur on all edges in the query execution plan. In practice, the precision of statistical models for cardinality estimation varies for different types of operators. For example, edges after foreign key joins can be estimated more precisely than after m:n joins, due to the constraint on keys [5, 17]. Also edges after base table scans can be estimated more precisely than edges after filter predicates. To consider the different risks for estimation errors, we define an *edge weighting function* φ as the next building block for the cardinality-slope robustness metric.

Definition 7. An **edge weighting function $\varphi : E_P \rightarrow [0.0, 1.0]$** assigns each edge $e \in E_P$ an error-sensitivity value between 0.0 (not sensitive) and 1.0 (very sensitive).

An edge after a m:n join should get a larger error-sensitivity value (e.g., 1.0) than an edge after a foreign key join (e.g., 0.0). The definition of the *cardinality-slope robustness metric* combines these building blocks.

Definition 8. The robustness value r_{δ_f} of the **cardinality-slope robustness metric** for a plan P is defined as the sum over the products of $\delta_{f,e}$ and $\varphi(e)$ for each edge $e \in E_P$:

$$r_{\delta_f}(P) = \sum_{e \in E_P} \varphi(e) \cdot \delta_{f,e}$$

Consequently, the smaller the robustness value, the more robust the plan. In Section 6, we experimentally evaluate the cardinality-slope robustness metric w.r.t. the consistency requirements of Section 3. The cardinality-slope robustness metric also follows our design considerations: Section 6 shows the low calculation overhead for r_{δ_f}. Potential cardinality estimation errors for different types of operators are weighted by φ. Finally, Definition 8 implicitly considers the potential propagation of cardinality estimation errors. As Theorem 1 shows, cardinality estimation errors on deep edges in query execution plans can have a stronger impact on the total cost and therefore the robustness value r_{δ_f}, compared to higher edges. This is not the case, when there is a very selective operator between the deep and the higher edge: a very selective operator can decrease the number of output tuples to almost zero. The cardinality estimation errors in the underlying sub-plan have almost no impact on the total cost and therefore on the robustness value r_{δ_f} anymore. We formalize and prove this observation in Theorem 1.

Theorem 1. Assuming a deep plan edge i with estimated cardinality $\hat{f}_i$ and cardinality-slope value $\delta_{f,i}$, as well as a higher plan edge j with estimated cardinality $\hat{f}_j$ and cardinality-slope value $\delta_{f,j}$. Then, for C_{out} it holds:

$$\delta_{f,i} \geq \delta_{f,j} \Leftrightarrow \frac{\hat{f}_j}{\hat{f}_i} \geq \frac{\delta_{f,j} - S}{1 + \delta_{f,j}}, \quad (3)$$

where $S \geq 0$ depends on the estimated cardinalities and selectivities between the deep edge i and the higher edge j.

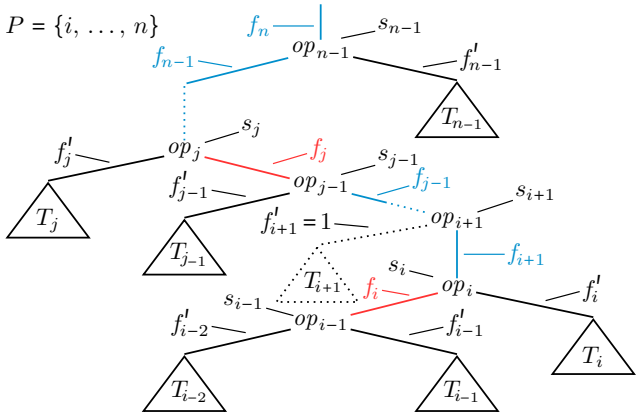


Figure 5: Arbitrary path in a query execution plan.

PROOF. Figure 5 shows an example of an arbitrary path P from edge i over edge j to the root edge n, in an arbitrary query execution plan. The arbitrary execution plan can have arbitrary operators and an arbitrary tree structure. The path contains unary and binary operators. We denote the cardinality of an edge e as f_e, and the selectivity of an operator op as s_{op}. It also shows non-path edges $e' \notin P$ with an arbitrary sub-tree T_e. The cardinality of an edge $e \in P \setminus \{i\}$ is defined as $f_e = f_{e-1} \cdot f'_{e-1} \cdot s_{e-1}$. For a more convenient notation we define for edge i that $f_i = f'_{i-1} \cdot f'_{i-2} \cdot s_{i-1}$. For unary operators such as filters, there is only one input edge, and therefore we add w.l.o.g. a second invisible input edge with cardinality 1 (see f'_{i+1} in Figure 5). Therefore, we can rewrite the estimated cardinality on an edge $e \in P$:

$$\hat{f}_e = \prod_{k=i-2}^{e-1} \hat{f}'_k \cdot \prod_{k=i-1}^{e-1} \hat{s}_k \tag{4}$$

We assume an edge $j \in P$ (cf. Figure 5) such that $i < j$, i.e., the edge i is deeper than the edge j. To see the impact of the estimated cardinality of the deeper edge $\hat{f}_i$, we rewrite the estimated cardinality of the higher edge $\hat{f}_j$ as:

$$\hat{f}_j = \hat{f}_i \cdot \prod_{k=i}^{j-1} \hat{f}'_k \cdot \prod_{k=i}^{j-1} \hat{s}_k \quad = \hat{f}'_i \cdot X \tag{5}$$

Before using Equation 5, we use Equation 4 to rewrite C_{out} (cf. Section 3.1). C_{out} is the sum over the estimated cardinalities of all edges, i.e., all edges $e \in P$, and all other edges $e' \notin P$ including all edges from their sub-tree T_e.

$$C_{out} = \sum_{l=i}^{n} \left(\prod_{k=i-2}^{l-1} \hat{f}'_k \cdot \prod_{k=i-1}^{l-1} \hat{s}_k \right) + \sum_{k=i-2}^{n-1} \hat{f}'_k + \sum_{k=i-2}^{n-1} C_{out}(T_k) \tag{6}$$

Next, we construct a PCF that models the C_{out} costs as a function of f_i, i.e., $PCF_{f,i}$. To do so, we factor out $\hat{f}_i$ from Equation 6 and use variable f_i instead of the estimation $\hat{f}_i$.

$$C_{out} = f_i \cdot \underbrace{\left[\sum_{l=i}^{n} \left(\prod_{k=i}^{l-1} \hat{f}'_k \cdot \prod_{k=i}^{l-1} \hat{s}_k \right) \right]}_{\text{dependent on } f_i \ (\delta_{f,i})} + \underbrace{\sum_{k=i-2}^{n-1} \hat{f}'_k + \sum_{k=i-2}^{n-1} C_{out}(T_k)}_{\text{independent of } f_i \ (c_{const,i})} \tag{7}$$

We observe from Equation 7 that costs dependent on f_i are the cardinality-slope value $\delta_{f,i}$ for the edge i. Next, we construct the $PCF_{f,j}$ for the higher edge j. Therefore, we also have to separate the sum over edges of P from Equation 6 into edges higher and deeper than j.

$$C_{out} = f_j \cdot \underbrace{\left[\sum_{l=j}^{n} \left(\prod_{k=j}^{l-1} \hat{f}'_k \cdot \prod_{k=j}^{l-1} \hat{s}_k \right) \right]}_{\text{dependent on } f_j \ (\delta_{f,j})} + \underbrace{\sum_{l=i}^{j-1} \hat{f}_l + \sum_{k=i-2}^{n-1} \hat{f}'_k + \sum_{k=i-2}^{n-1} C_{out}(T_k)}_{\text{independent of } f_j \ (c_{const,j})} \tag{8}$$

Now, we reformulate the part of Equation 7 that depends on f_i, to quantify the impact of $\delta_{f,j}$ on $\delta_{f,i}$. We separate the sums into one running from i to $j-1$ and another from j to n, and factor out X (cf. Equation 5) from the latter sum.

$$\delta_{f,i} = \underbrace{\sum_{l=i}^{j-1} \left(\prod_{k=i}^{l-1} \hat{f}'_k \cdot \prod_{k=i}^{l-1} \hat{s}_k \right)}_{S+X} + \underbrace{\prod_{k=i}^{j-1} \hat{f}'_k \cdot \prod_{k=i}^{j-1} \hat{s}_k}_{X} \cdot \underbrace{\left[\sum_{l=j}^{n} \left(\prod_{k=j}^{l-1} \hat{f}'_k \cdot \prod_{k=j}^{l-1} \hat{s}_k \right) \right]}_{\delta_{f,j}} \tag{9}$$

We observe that the product, denoted as X, is the last term of $S + X$. Let us presume $\delta_{f,i} \geq \delta_{f,j}$:

$$\delta_{f,i} \geq \delta_{f,j} \overset{(9)}{\iff} S + X \cdot (1 + \delta_{f,j}) \geq \delta_{f,j} \tag{10}$$

$$\iff X \geq \frac{\delta_{f,j} - S}{1 + \delta_{f,j}} \overset{(5)}{\iff} \frac{\hat{f}_j}{\hat{f}_i} \geq \frac{\delta_{f,j} - S}{1 + \delta_{f,j}} \tag{11}$$

By inserting $\delta_{f,i}$ into Equation 10, we factored out X. From Equation 10 to 11, we first subtracted S and second divided it by $1 + \delta_{f,j}$. Finally, Equation 5 is inserted. □

From Theorem 1, we observe that the right hand side term of Equation 3 is always less than 1, since cardinalities and selectivities are non-negative. Therefore, if the estimated cardinality of the deep edge is smaller or equal to the estimated cardinality of the higher edge ($\hat{f}_j \,/\, \hat{f}_i \geq 1$), then the deep edge has a larger cardinality-slope value ($\delta_{f,i} > \delta_{f,j}$). In contrast, if there is a highly selective operator between the deep and the higher edge ($\hat{f}_j \,/\, \hat{f}_i < 1$), then the right hand side term of Equation 3 is a tight bound for $\delta_{f,i} \geq \delta_{f,j}$, i.e., for a highly selective operator $\delta_{f,i} < \delta_{f,j}$ can hold. Furthermore, Theorem 1 can be extended and proved for cost functions other than C_{out}. The reason is that cost functions in general have dependencies on cardinalities, and the cardinality of an edge is always a parameter in the following cardinality estimations towards the root.

Let us again consider JOB Query 17. Figure 6 shows the robustness value calculation of the cardinality-slope robustness metric for the estimated optimal plan P_{opt} and the estimated robust plan P_{rob}, i.e., the plan with the minimum robustness value r_{δ_f} from the robust plan candidates. For simplicity, we use an edge weighting function φ that assigns weight 1.0 to all m:n join edges and weight 0.0 to all foreign key and base table scan edges. For both plans, the dashed edges are the edges that include m:n joins, i.e., the edges that are more sensitive to estimation errors. The corresponding PCF, including the δ_f value, is shown right of those edges. For example, the cardinality-slope value δ_{f,e_F} for edge e_F of P_{rob} is 2.49, i.e., the slope of PCF_{f,e_F} as calculated in Equation 1. As a result, the robustness value r_{δ_f} for P_{opt} is 33.49 and for P_{rob} 3.49. Therefore, P_{rob} is more robust according to the cardinality-slope robustness metric than P_{opt}. The true statistics in Figure 2 and 3 result in a smaller c_{err} for P_{rob} compared to P_{opt}. Executing both plans on the real-world database of the JOB shows an execution time speedup of factor two for P_{rob} compared to P_{opt}.

4.2 Selectivity-Slope Robustness Metric

The cardinality-slope value δ_f for an edge $e \in E_P$ expresses the impact of one additional tuple on the total cost. Apart from the edge weighting function φ for potential cardinality estimation errors for different types of operators and the implicitly considered propagation of cardinality estimation errors, the edges in r_{δ_f} are not further weighted. In order to explain a derived robustness metric, we first denote $\hat{f}_{\max}$ as the estimated maximum output cardinality of an operator. Taking a binary join as an example, $\hat{f}_{\max}$ is the product of its estimated input cardinalities (cross-product). We argue that edges with a potentially larger absolute cardinality error Δf, i.e., the absolute difference between the estimated and the true cardinality, can have a stronger impact on the final c_{err}. Since Δf cannot be calculated before executing the plan, the derived robustness metric in this section considers the risk of a large Δf, by taking $\hat{f}_{\max}$ into account. The larger $\hat{f}_{\max}$, the larger the potential impact on the final c_{err}. Next, we define the *selectivity-slope value* δ_s and the corresponding *selectivity-slope robustness metric*.

Definition 9. The **selectivity-slope value $\delta_{s,op}$** for an operator $op \in O_P$ is the slope of $PCF_{s,op}$ at the estimated selectivity $\hat{s}$, where $PCF_{s,op}$ is the PCF that models the cost for a plan P as a function of the selectivity s on op.

Definition 10. The robustness value r_{δ_s} of the **selectivity-slope robustness metric** for a plan P is the sum over the products of $\delta_{s,op}$ and $\phi(op)$ for each operator $op \in O_P$:

$$r_{\delta_s}(P) = \sum_{op \in O_P} \phi(op) \cdot \delta_{s,op},$$

where $\phi : O_P \to [0.0, 1.0]$ is a weighting function for operators, instead for edges as φ.

We show that the selectivity-slope robustness metric implicitly weights the cardinality-slope value $\delta_{f,e}$ for the outgoing edge $e \in E_P$ of an operator $op \in O_P$ by $\hat{f}_{\max}$.

Theorem 2. For C_{out}, the selectivity-slope value $\delta_{s,op}$ of an operator $op \in O_P$ is the product of $\hat{f}_{\max}$ and $\delta_{f,e}$ on the outgoing edge $e \in E_P$ of op.

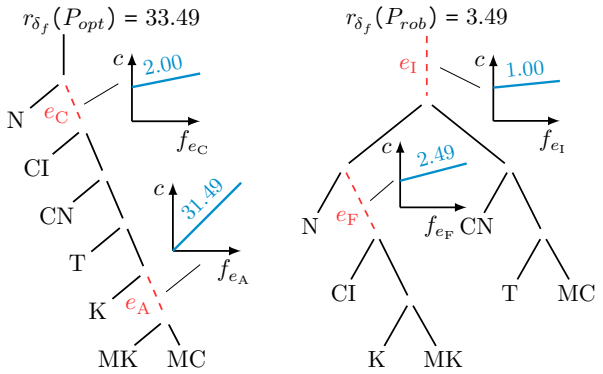
Proof. Without loss of generality, consider the edge i in Figure 5 with the cardinality f_i. From Equation 7 in the proof of Theorem 1, we see that the $PCF_{f,i}$ for edge i consists of costs independent of f_i, denoted as $c_{const,i}$, and costs dependent on f_i, i.e., the cardinality-slope value $\delta_{f,i}$.

$$C_{out} = f_i \cdot \delta_{f,i} + c_{const,i} \tag{12}$$

As in Figure 5, we denote s_{i-1} as the selectivity of operator op_{i-1}, i.e., the operator before edge i. Furthermore, we denote f'_{i-1} and f'_{i-2} as the input cardinalities of op_{i-1}. The cardinality of edge i is the product of both input cardinalities of op_{i-1} and the selectivity s_{i-1}, i.e., $f_i = f'_{i-1} \cdot f'_{i-2} \cdot s_{i-1}$. Note that for unary operators having only one input edge such as filters, we added in Theorem 1 without loss of generality a second invisible input edge with cardinality 1 (see f'_{i+1} in Figure 5). Therefore, we rewrite Equation 12 as:

$$C_{out} = f'_{i-1} \cdot f'_{i-2} \cdot s_{i-1} \cdot \delta_{f,i} + c_{const,i} \tag{13}$$

In order to rewrite Equation 13 into a PCF with s_{i-1} as a single cost parameter ($PCF_{s,i-1}$), the cardinality variables



(a) Estimated Optimal Plan (b) Estimated Robust Plan

Figure 6: Robustness values r_{δ_f} assigned to robust plan candidates for JOB Query 17.

for the input edges f'_{i-1} and f'_{i-2} are set to the corresponding estimated cardinalities $\hat{f}'_{i-1}$ and $\hat{f}'_{i-2}$.

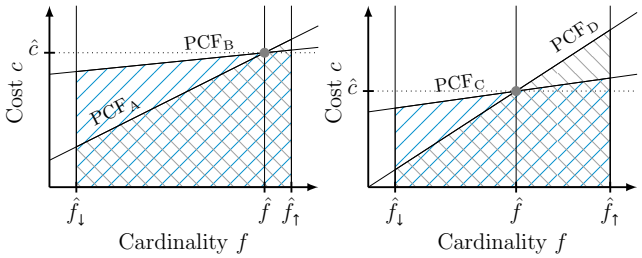
$$C_{out} = \hat{f}'_{i-1} \cdot \hat{f}'_{i-2} \cdot \delta_{f,i} \cdot s_{i-1} + c_{const,i} = \delta_{s,i-1} \cdot s_{i-1} + c_{const,i} \tag{14}$$

Since $c_{const,i}$ is independent of f_i, it has no cost depending on s_{i-1}. Therefore, $\delta_{s,i-1}$ for the operator op_{i-1} is the product of $\hat{f}_{\max} = \hat{f}'_{i-1} \cdot \hat{f}'_{i-2}$ and the cardinality-slope value $\delta_{f,i}$ of the outgoing edge i of the operator op_{i-1}. Note that for unary operators having only one input edge such as filters, $\hat{f}'_{i-2}$ or $\hat{f}'_{i-1}$ is set to 1, and therefore has no impact. □

In Section 6, we experimentally evaluate the selectivity-slope robustness metric w.r.t. the consistency requirements of Section 3. The selectivity-slope robustness metric also follows our design considerations: the calculation effort is small, potential cardinality estimation errors are weighted, and the propagation of cardinality estimation is considered. A proof for the latter can be constructed analogous to the proof of Theorem 1 by adding the additional weight of $\hat{f}_{\max}$. In summary, the selectivity-slope robustness metric additionally considers the risk of a large Δf on all edges, compared to the cardinality-slope robustness metric.

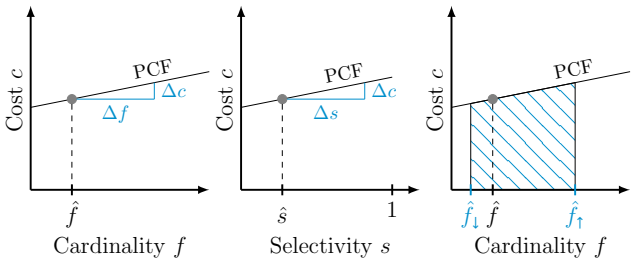
4.3 Cardinality-Integral Robustness Metric

The next robustness metric is a trade-off between plan robustness and estimated costs. Both the cardinality-slope and the selectivity-slope robustness metric use the slopes of PCFs as the robustness indicator. However, a plan with a steep slope could still have smaller costs for a significant range of cardinality values, compared to a plan with a more moderate slope. Figure 7(a) shows PCF_A and PCF_B of two different plans as a function of cardinality on a single plan edge. The cardinality of the edge is denoted on the x-axis and the cost on the y-axis. Furthermore, it shows $\hat{f}$, $\hat{f}_\downarrow$, and $\hat{f}_\uparrow$, where $\hat{f}_\downarrow$ is the lower bound for the estimated cardinality of an edge $e \in E_P$, and $\hat{f}_\uparrow$ is the upper bound for the estimated cardinality of $e \in E_P$. We argue that a lower and a upper bound for the estimated cardinality can make the robustness metric more precise. In practice, histograms, sampling, or bounds for cardinality estimation [21] can give estimations for $\hat{f}_\downarrow$ and $\hat{f}_\uparrow$. In the evaluation in Section 6, we set $\hat{f}_\downarrow$ to 0, and $\hat{f}_\uparrow$ to $\hat{f}_{\max}$. Note that PCF_A and PCF_B have the same estimated costs $\hat{c}$ at $\hat{f}$. Since PCF_B has a more moderate slope than PCF_A, the cardinality-slope robustness



(a) $\int_{\hat{f}_\downarrow}^{\hat{f}_\uparrow} \text{PCF}_\text{A} < \int_{\hat{f}_\downarrow}^{\hat{f}_\uparrow} \text{PCF}_\text{B}$ (b) $\int_{\hat{f}_\downarrow}^{\hat{f}_\uparrow} \text{PCF}_\text{C} > \int_{\hat{f}_\downarrow}^{\hat{f}_\uparrow} \text{PCF}_\text{D}$

Figure 7: Conceptual comparison between slope and integral robustness indicator.



(a) Card.-Slope (b) Selectiv.-Slope (c) Card.-Integral

Figure 8: Cardinality-slope, selectivity-slope and cardinality-integral robustness metric.

metric would assign PCF_B a smaller robustness value than PCF_A. By considering the costs between $\hat{f}_\downarrow$ and $\hat{f}_\uparrow$, a robustness metric that is a trade-off between plan robustness and estimated costs would consider PCF_A to be more robust than PCF_B. The reason is that PCF_A has significant less cost for a majority of cardinalities between $\hat{f}_\downarrow$ and $\hat{f}_\uparrow$, i.e., PCF_A has significantly less cost between $\hat{f}_\downarrow$ and $\hat{f}$ than PCF_B, and is competitive to PCF_B between $\hat{f}$ and $\hat{f}_\uparrow$. To model plan robustness and estimated costs in a single value, we consider the integral of the PCF between $\hat{f}_\downarrow$ and $\hat{f}_\uparrow$. In Figure 7(a), the integral of PCF_A is smaller than the integral of PCF_B. Next, we define the *cardinality-integral value* $\int_f$ as a trade-off between plan robustness and cost, and afterwards the *cardinality-integral robustness metric*.

Definition 11. The **cardinality-integral value** $\int_{f,e}$ for an edge e is $\int_{\hat{f}_\downarrow}^{\hat{f}_\uparrow} PCF_{f,e}$.

Definition 12. The robustness value $r_{\int_f}$ of the **cardinality-integral robustness metric** for a plan P is defined as the sum over the products of $\int_{f,e}$ and $\varphi(e)$ for each edge $e \in E_P$:

$$r_{\int_f}(P) = \sum_{e \in E} \varphi(e) \cdot \int_{f,e}$$

A second scenario in Figure 7(b) shows PCF_C and PCF_D. The integral between $\hat{f}_\downarrow$ and $\hat{f}_\uparrow$ of PCF_D is slightly smaller compared to PCF_C. Hence, the cardinality-integral robustness metric considers PCF_D to be more robust than PCF_C. In contrast to Figure 7(a), both plans in Figure 7(b) have smaller cost for a wide cardinality range. PCF_D has smaller cost from $\hat{f}_\downarrow$ to $\hat{f}$ and PCF_C from $\hat{f}$ to $\hat{f}_\uparrow$. Furthermore, the difference between the estimated and the true cost for all cardinalities from $\hat{f}_\downarrow$ to $\hat{f}_\uparrow$ is smaller for PCF_C than for PCF_D (cf. Figure 4). This means that PCF_C should get a smaller robustness value than PCF_D. Note that the cardinality-slope robustness metric assigns PCF_C a smaller robustness value than PCF_D, because of the more moderate slope of PCF_C. Both scenarios in Figure 7 show how a lower and a upper bound, $\hat{f}_\downarrow$ and $\hat{f}_\uparrow$, for the estimated cardinality of an edge $e \in E_P$ can impact the robustness of a plan.

Calculating the integral makes the metric independent of $\hat{f}$ and the slope at this point. In addition, we can support arbitrary PCF shapes, because integrals can always be approximated numerically [13]. Section 6 shows the experimental evaluation of the cardinality-integral robustness metric w.r.t. the consistency requirements of Section 3. The cardinality-integral metric follows two design considerations: it has a low calculation effort, and potential cardinality estimation errors are weighted. Since the cardinality-integral metric calculates integrals to balance plan robustness and costs, it considers high plan edges stronger than deeper plan edges. This is because plan edges always contain the cost of their sub-plans. Consequently, the integrals are larger on high plan edges compared to the deeper plan edges, and therefore have a higher impact on the robustness value.

4.4 Robustness Metrics Overview

Figure 8 summarizes the three robustness metrics. The cardinality-slope metric (Figure 8(a)) reflects the expected difference between estimated and true cost for cardinality estimation errors on all edges in the query execution plan. Furthermore, it implicitly considers the potential propagation of cardinality estimation errors, and takes the potential cardinality estimation errors for different types of operators into account. In addition, the selectivity-slope metric (Figure 8(b)) considers the risk of a large absolute cardinality error Δf on all edges. Therefore, it models the PCFs as function of operator selectivity, compared to the cardinality-slope metric. In contrast to the cardinality-slope and the selectivity-slope metric, the cardinality-integral metric (Figure 8(c)) does not purely focus on plan robustness, but also takes estimated costs into consideration. Furthermore, it can consider a more realistic range for the cardinality of an edge, instead of considering all numerically possible cardinalities. All three metrics support any kind of operator, operator implementation and query execution plan trees because the cost of a plan can always be modeled as a PCF of cardinality. In addition, the metrics can be extended to consider estimation errors in other cost parameters, such as consumed memory. We also experimented with a fourth metric, namely *selectivity-integral*, but found no substantial improvement over the cardinality-integral metric.

5. PLAN CANDIDATES AND SELECTION

Our novel robust plan selection strategy has three phases: First, we enumerate the set of *robust plan candidates*. Every robust plan candidate is a plan for the entire query, and not a sub-plan. Second, we calculate the robustness value for each robust plan candidate by applying one of the three robustness metrics. Third, we select the *estimated most robust plan*, i.e., the robust plan candidate with the smallest robustness value for execution. Apart from robustness, selecting a cheap query execution plan is still a major optimization goal. Consequently, our first criteria for the robust plan candidates is that they have to be the *k-cheapest plans*:

Definition 13. The ***k*-cheapest plans** are the k query execution plans with the smallest estimated cost.

The k-cheapest plans significantly reduce the number of plan candidates, and give a tight upper bound for the number of plans independent of the plan space. In addition, the k-cheapest plans can be utilized to apply additional constraints, such as memory consumption, on the plan set. Section 6 shows that $k = 500$ has a low optimization overhead. Furthermore, we show that the estimated robust plan inside $k = 500$ is competitive w.r.t. an estimated robust plan with larger k. Enumerating the k-cheapest plans is just a small modification in the optimizer. The trivial approach in a dynamic programming enumerator is to keep the k-cheapest plans in each plan class, instead of the cheapest plan. The k-cheapest plans of two plan classes can be combined to create plans of another plan class. We show in Section 6 that enumerating the k-cheapest plans is a reasonable overhead.

Since the k-cheapest plans can contain expensive plans for small queries, and only the cardinality-integral robustness metric takes plan cost into consideration, we further limit the robust plan candidates for the cardinality-slope and the selectivity-slope robustness metric to near-optimal plans:

Definition 14. The **near-optimal plans** are a sub-set of the plan space, containing the query execution plans with estimated cost at most λ-times larger than the estimated cost of the estimated optimal plan.

The near-optimal plans guarantee that robust plan candidates are competitive to the optimal plan. Cost-Stable Plans [1] argue for $\lambda = 1.2$, which we confirm in Section 6.

In sum, our plan selection strategy has very low risks: First, we enumerate the k-cheapest plans. Second, we calculate the robustness value for each robust plan candidate. Though it is a reasonable overhead, it can be significant in very short running queries. It is not significant in our real-world experiments in Section 6.1. In addition, dynamic programming enumeration is no limitation, but shows that our approach can be integrated into enterprise class optimizers.

6. EVALUATION

We implemented the three robustness metrics in our dynamic programming join optimizer. We use the same optimizer to determine the baseline plan for each query, i.e., the estimated cheapest or optimal plan. Our join optimizer relies on dynamic programming [23], such as DB2 [9] and Postgres [17]. As Postgres, it exhaustively searches the plan space including bushy trees. We have shown that its cardinality estimator is competitive [24]. In addition, we use the C_{mm} [18] cost function, which is an extension of C_{out} that considers different operator types and operator implementations. It also has a strong correlation to our main-memory execution engine[1], which we use to determine query execution times. Therefore, we argue that our join optimizer's choice of the estimated optimal plan is very similar to the choice of popular commercial and free systems, for the considered join queries. We denote its estimated optimal plan choice as conventional plan, and consider it as the baseline.

We experimentally evaluate the plan selection strategies w.r.t. their end-to-end query execution times (Section 6.1), and plan robustness (Section 6.2). The numbers we report in Section 6.2 only depend on the robustness metrics implementation and not on the machine the experiments run on. Reported execution times were taken on a two socket Intel Xeon E5-2660 v3 system with 128 GB of main memory, running a Linux 4.4.120 kernel. Our engine[1] performs join operators as hash join. The dynamic programming optimizer and metric implementations are single-threaded. The entire system is compiled with GCC 7.2.0 using option -O3.

Our first workload is based on the Join Order Benchmark (JOB) [17]. JOB uses real-world data from IMDb with skew, correlations, and different join relationships that cause estimation errors. We modified the original queries to be pure join queries, which results in 33 complex queries containing cycles and multiple join conditions between subplans. Since pure join queries without any filters on base tables create large results, we limit the `movie_id` of the tables to 100,000 rows. In the end, we ran 31 different queries. Note that this does not limit the applicability of the results.

Our second benchmark is synthetic with generated data and join queries. The query topologies are: chain, cycle, and snowflake. All topologies join 10 tables. The snowflake topology has a fact table with three dimension tables, and each dimension again two sub-dimensions. For each topology, we create one query and 100 different data sets. Furthermore, we generate 100 queries with a random topology and a corresponding data set. The random topology generator starts with a random connected query graph, into which additional edges are randomly inserted to create cycles. The random topologies also join 10 tables. For all generated data sets, the base table cardinalities are uniform random numbers between 10,000 and 100,000. The data sets contain skew and arbitrary correlations between columns to generate expanding and selective joins. There are foreign key and m:n join relationships. The join cardinalities between two base tables R_i and R_j are uniform random numbers between $max(|R_i|, |R_j|) - 5000$ and $max(|R_i|, |R_j|) + 5000$.

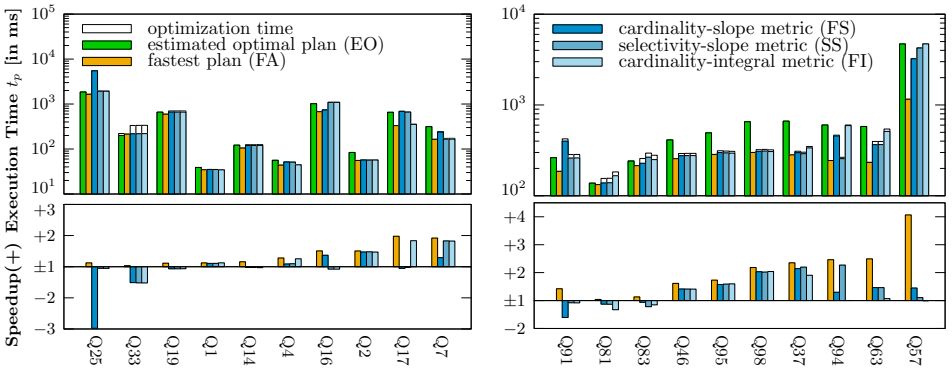
Each experiment starts with enumerating the robust plan candidates. For the cardinality-slope and the selectivity-slope metric, the robust plan candidates are defined by the near-optimal plans ($\lambda = 1.2$) and the k-cheapest plans ($k = 500$). For the cardinality-integral metric it is only the k-cheapest plans ($k = 500$). By definition, the k-cheapest plans contain the estimated optimal plan, which is the baseline in our experiments. To select the estimated robust plan, each metric assigns a robustness value to every robust plan candidate. Both workloads contain only join queries with at least one m:n join, and there are no estimation errors for foreign key joins and base table scans in our setup. Therefore, we define the weighting functions φ and ϕ to be 1.0 for m:n joins, and 0.0 for foreign key joins and base table scans.

We compare our baseline, the estimated optimal plan (EO), with the estimated most robust plan according to one of the metrics: cardinality-slope (FS), selectivity-slope (SS), and cardinality-integral metric (FI). We also perform a best-case offline analysis to show the potential of robust plan selection. We execute all robust plan candidates and denote the plan with the lowest execution time as the fastest plan (FA).

6.1 Query Execution Time

Figure 9(a) shows the JOB queries plotted along the x-axis. The y-axis shows the median end-to-end query execution time t_p for a plan p in milliseconds (log-scale) over 101 executions. In addition, the y-axis shows the resulting speedup ($+\,t_{EO}/t_p$) or regression ($-\,t_p/t_{EO}$) of robust plan selection w.r.t. conventional plan selection (EO). We show typical results, including the queries with the best speedup

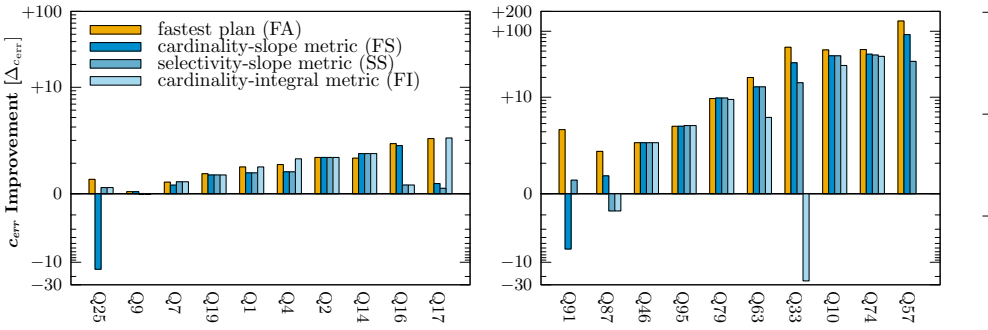
[1]Finalist in the 2018 ACM SIGMOD Programming Contest



		Σ	↑EO	↓EO
Chain	EO	18798 ms	–	–
	FA	14562 ms	+4.23×	1.00×
	FS	16091 ms	+3.31×	−1.26×
	SS	17061 ms	+1.79×	−1.36×
	FI	16865 ms	+3.49×	−1.13×
Cycle	EO	41084 ms	–	–
	FA	25539 ms	+2.94×	1.00×
	FS	34587 ms	+2.43×	−1.21×
	SS	32279 ms	+2.43×	−1.27×
	FI	33193 ms	+2.43×	−1.25×
Snowflake	EO	53793 ms	–	–
	FA	44843 ms	+2.11×	1.00×
	FS	54437 ms	+1.53×	−2.07×
	SS	51579 ms	+1.91×	−1.36×
	FI	53327 ms	+1.78×	−1.38×

(a) Join Order Benchmark (b) Random Topology (c) Other Topologies

Figure 9: Comparison of end-to-end query execution times for plan selection strategies.



		$\mu_{\Delta_{c_{err}}}$	$\uparrow_{\Delta_{c_{err}}}$	$\downarrow_{\Delta_{c_{err}}}$
Chain	FA	+0.85	+20.69	0.00
	FS	+0.80	+19.19	0.00
	SS	+0.73	+15.43	0.00
	FI	+0.55	+17.98	−0.20
Cycle	FA	+5.62	+25.71	0.00
	FS	+4.51	+25.04	0.00
	SS	+4.72	+24.87	0.00
	FI	+3.30	+20.02	−0.25
Snowflake	FA	+2.19	+39.35	−2.03
	FS	+1.28	+18.07	−1.87
	SS	+1.48	+37.13	−2.87
	FI	+0.75	+13.47	−4.90

(a) Join Order Benchmark (b) Random Topology (c) Other Topologies

Figure 10: Comparison of cost error factor improvement for plan selection strategies.

(Q2 and Q7) and the worst regression (Q25 and Q33).

The best speedup is achieved for FS in Q2 (1.47), and for SS and FI in Q7 (1.83). In contrast, the worst regression for SS and FI is only 1.52 (Q33). For FS the worst regression is 2.98 (Q25), but the second worst regression is again only 1.51 (Q33). By considering near-optimal plans and the k-cheapest plans, the estimated most robust plan does not necessarily have the minimum estimated costs and small regressions for some queries can be the result. Comparing the results of Q2 and Q7 to the fastest plan (FA), found in a brute-force analysis, shows that all robust plan selection strategies are close to the true optimum in these cases.

We show the results of the synthetic benchmark in Figures 9(b) and 9(c). Figure 9(b) shows typical results for random topologies, including Q37, Q94, and Q98 with the best speedup as well as Q81, Q83, and Q91 with the worst regression. Results for chain, cycle, and snowflake topologies are summarized in Figure 9(c), by cumulative query execution time (Σ), best speedup ($\uparrow_{\text{EO}}$), and worst regression ($\downarrow_{\text{EO}}$) w.r.t. EO over 100 different data sets. For all three metrics, robust plan selection achieves a better cumulative query execution time than conventional plan selection. Furthermore, all three metrics achieve larger speedup than regression factors for all query topologies. Comparing the results of Q37, Q95, and Q98 to FA, shows that all robust plan selection strategies are close to the true optimum in these cases.

6.2 Plan Robustness

In every subsection, we evaluate one consistency requirement for the robustness metric presented in Section 3.

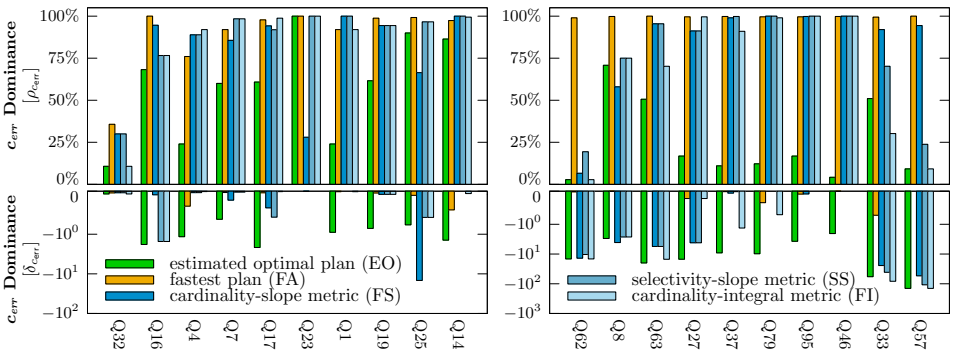
6.2.1 Cost Error Factor Improvement

According to the first consistency requirement, the estimated most robust plan should have a smaller cost error factor c_{err} than the estimated optimal plan (cf. Section 3). To measure the cost error factor improvement, we calculate the difference between the c_{err} of the estimated optimal plan ($c_{\text{err,EO}}$) and the c_{err} of another plan p ($c_{\text{err},p}$):

$$\Delta_{c_{\text{err},p}} = c_{\text{err,EO}} - c_{\text{err},p}$$

Consequently, a positive $\Delta_{c_{\text{err},p}}$ shows the c_{err} improvement of p compared to EO. Figure 10(a) shows typical $\Delta_{c_{\text{err},p}}$ values (y-axis, log-scale) of JOB queries (x-axis), including the queries with the largest $\Delta_{c_{\text{err},p}}$ (Q14, Q16, and Q17) and the smallest $\Delta_{c_{\text{err},p}}$ (Q9 and Q25). Robust plan selection using SS and FI achieves a positive $\Delta_{c_{\text{err},p}}$ in 30 of the 31 queries. Furthermore, robust plan selection using FS achieves a positive $\Delta_{c_{\text{err},p}}$ in 29 of 31 queries. Comparing with the fastest plan (FA) shows that the fastest plan is not necessarily as robust as the estimated most robust plan, since there can be a large difference between estimated and true cost for FA. Considering, e.g., JOB Q14, robust plan selection with FS, SS, and FI achieves a larger $\Delta_{c_{\text{err},p}}$ than FA.

The results of the synthetic benchmark are shown in Figures 10(b) and 10(c). Figure 10(b) shows typical results for random topologies, including the queries with the largest $\Delta_{c_{\text{err},p}}$ (Q57 and Q74) and the smallest $\Delta_{c_{\text{err},p}}$ (Q33, Q87, and Q91). Results for chain, cycle, and snowflake topologies are summarized in Figure 10(c), by average ($\mu_{\Delta_{c_{\text{err}}}}$), largest ($\uparrow_{\Delta_{c_{\text{err}}}}$), and smallest ($\downarrow_{\Delta_{c_{\text{err}}}}$) $\Delta_{c_{\text{err},p}}$ over 100 different data sets. For all three robustness metrics, robust



(a) Join Order Benchmark (b) Random Topology

		$\mu_{\rho_{c_{\text{err}}}}$	$\mu_{\delta_{c_{\text{err}}}}$	$\downarrow_{\delta_{c_{\text{err}}}}$
Chain	EO	31.75%	−0.94	−20.69
	FA	92.95%	−0.07	−1.47
	FS	93.09%	−0.11	−2.82
	SS	89.37%	−0.18	−4.49
	FI	63.97%	−0.38	−2.71
Cycle	EO	20.63%	−6.04	−26.27
	FA	95.13%	−0.36	−5.42
	FS	91.53%	−1.02	−7.81
	SS	90.50%	−0.80	−6.05
	FI	63.21%	−2.71	−13.60
Snowflake	EO	39.16%	−2.51	−41.33
	FA	93.91%	−0.30	−3.73
	FS	86.50%	−1.22	−30.62
	SS	88.87%	−1.01	−40.72
	FI	74.26%	−1.74	−40.72

(c) Other Topologies

Figure 11: Comparison of cost error factor dominance for plan selection strategies.

plan selection achieves a positive $\mu_{\Delta_{c_{\text{err},p}}}$. Also, FS and SS achieves a positive $\Delta_{c_{\text{err},p}}$ for all generated chain and cycle queries. A comparison of $\uparrow_{\Delta_{c_{\text{err}}}}$ and $\downarrow_{\Delta_{c_{\text{err}}}}$ for FS, SS and FI shows that the maximum gain is always larger than the maximum loss of c_{err}. Considering FS in Q57 of the random topology, shows the best $\Delta_{c_{\text{err},p}}$ of 89.02. The comparison of FI to FS and SS in Figure 10(c) shows that FI always achieves worse results. The reason is that the cardinality-integral robustness metric already balances plan robustness and estimated costs. In contrast, the cardinality-slope and selectivity-slope robustness metrics only focus on plan robustness and achieve this trade-off by limiting robust plan candidates to near-optimal plans. We also observe that the fastest plan (FA) is not necessarily as robust as the estimated most robust plan. For instance, FS and SS achieves a larger $\Delta_{c_{\text{err},p}}$ than FA for Q79 of the random topology.

6.2.2 Cost Error Factor Dominance

According to the second consistency requirement, the estimated most robust plan, chosen by robust plan selection, should dominate all robust plan candidates, denoted as RPC, with respect to their c_{err} (cf. Section 3). In order to measure the cost error factor dominance, we define $\rho_{c_{\text{err},p}}$ of a plan p:

$$\rho_{c_{\text{err},p}} = \frac{\left|\{r \mid c_{\text{err},p} \leq c_{\text{err},r}, r \in RPC\}\right|}{|RPC|}$$

A $\rho_{c_{\text{err},p}}$ of 100% indicates that a plan has the smallest c_{err} of all robust plan candidates, i.e., is the most robust plan. In practice, a $\rho_{c_{\text{err},p}}$ of 100% cannot be achieved for every query, since the robustness value assigned by a robustness metric is an approximation for an upper bound of c_{err}. Therefore, we additionally define $\delta_{c_{\text{err},p}}$ as the difference between c_{err} of a plan p and c_{err} of the most robust plan $r \in RPC$:

$$\delta_{c_{\text{err},p}} = min\{c_{\text{err},r} \mid r \in RPC\} - c_{\text{err},p}$$

A $\delta_{c_{\text{err},p}}$ close to 0 indicates that a plan p has a similar c_{err} as the plan with the smallest c_{err} from the robust plan candidates, i.e., the most robust plan. Figure 11(a) plots $\rho_{c_{\text{err},p}}$ and $\delta_{c_{\text{err},p}}$ for JOB queries (x-axis). The y-axes show $\rho_{c_{\text{err},p}}$ (in percent) and $\delta_{c_{\text{err},p}}$ (in log-scale). We show typical results, including the queries with the best $\rho_{c_{\text{err},p}}$ and $\delta_{c_{\text{err},p}}$ (Q14 and Q23) and the worst $\rho_{c_{\text{err},p}}$ and $\delta_{c_{\text{err},p}}$ (Q16, Q23, Q25, and Q32). Overall, robust plan selection with FS and SS achieves a $\rho_{c_{\text{err},p}}$ of 100% for 13 of the 31 executed queries, i.e., robust plan selection chooses the most robust plan. A $\rho_{c_{\text{err},p}} \geq 80\%$ is achieved for FS and SS for 25 of the 31 executed queries. In contrast, conventional plan selection with EO achieves $\rho_{c_{\text{err},p}} \geq 80\%$ for only 12 of the 31 executed queries. The average $\delta_{c_{\text{err},p}}$ over all 31 JOB queries is better for SS (−0.11) and FI (−0.12) compared to EO (−0.50). Considering the fastest plan (FA), we again observe that it is not necessarily as robust as the estimated most robust plan. The average $\rho_{c_{\text{err},p}}$ over all 31 JOB queries with SS (92.65%) is larger than the average $\rho_{c_{\text{err},p}}$ of FA (89.37%).

Figures 11(b) and 11(c) show the synthetic benchmark results. Figure 11(b) plots typical results for random topologies, including Q46 and Q95 with the best $\rho_{c_{\text{err},p}}$ and $\delta_{c_{\text{err},p}}$, and Q57 and Q62 with the worst $\rho_{c_{\text{err},p}}$ and $\delta_{c_{\text{err},p}}$. Figure 11(c) summarizes the results for chain, cycle, and snowflake topologies, by average $\rho_{c_{\text{err},p}}$ ($\mu_{\rho_{c_{\text{err}}}}$), average $\delta_{c_{\text{err},p}}$ ($\mu_{\delta_{c_{\text{err}}}}$), and worst $\delta_{c_{\text{err},p}}$ ($\downarrow_{\delta_{c_{\text{err}}}}$) over 100 different data sets. FS and SS achieve a significantly larger $\mu_{\rho_{c_{\text{err},p}}}$ (83%–93%) compared to EO (21%–47%) for all query topologies. The worst $\delta_{c_{\text{err},p}}$ for EO is substantially larger for chain (−20.69) or cycle queries (−26.27). Considering only Q79 of the random topology shows that FS and SS chooses the most robust plan with $\rho_{c_{\text{err},p}} = 100\%$ and $\delta_{c_{\text{err},p}} = 0.0$, whereas EO chooses a volatile plan with $\rho_{c_{\text{err},p}} = 12.20\%$ and $\delta_{c_{\text{err},p}} = -9.74$. A comparison of $\mu_{\rho_{c_{\text{err}}}}$, $\mu_{\delta_{c_{\text{err}}}}$ and $\downarrow_{\delta_{c_{\text{err}}}}$ of FI to FS and SS in Figure 11(c) shows that FI is outperformed by FS and SS. Again, the reason is that FI balances plan robustness and estimated costs. However, $\mu_{\rho_{c_{\text{err}}}}$, $\mu_{\delta_{c_{\text{err}}}}$ and $\downarrow_{\delta_{c_{\text{err}}}}$ for FI are still substantially better w.r.t. the estimated optimal plan.

6.2.3 Correlated Cost Error Factor Limit

According to the third consistency requirement, a large c_{err} for a plan with a small robustness value indicates a failure of the metric (cf. Section 3). Since cardinality estimations can be precise and always result in a small cost error factor c_{err}, even if a large robustness value is assigned, the correlation between the robustness value and c_{err} cannot be used to evaluate the third requirement. To evaluate the requirement, we draw all robust plan candidates of a query into a single plot. Figure 12 shows some typical results for the selectivity-slope metric, including the JOB queries 7, 14, 19, and 25. The assigned robustness value r_{δ_s} is plotted on the x-axis in logarithmic scale, and the c_{err} on the y-axis in logarithmic scale. Additionally, we highlight the estimated optimal plan, the fastest plan and the estimated most robust plan. For Q14, we see a strong correlation between r_{δ_s} and c_{err}, i.e., there is no robust plan candidate with a smaller r_{δ_s} and a larger c_{err}. For Q7 and Q19, we see that the cor-

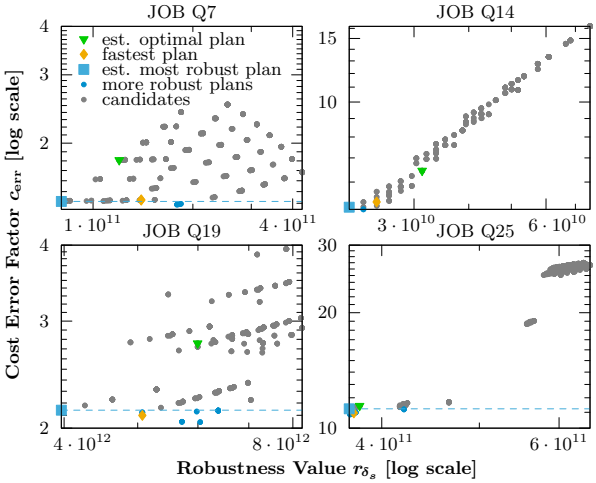


Figure 12: Correlated cost error factor limit for the selectivity-slope robustness metric.

Table 1: Optimization time relative to the end-to-end query execution time.

	JOB	Chain	Cycle	Snowflake	Random
est. optimal plan	0.98%	0.14%	0.12%	0.17%	0.13%
cardinality-slope	3.94%	3.70%	3.10%	3.06%	3.08%
selectivity-slope	4.78%	3.51%	3.32%	3.20%	3.10%
cardinality-integral	4.91%	3.68%	3.62%	3.11%	2.96%

related cost error factor limit requirement is fulfilled, even if there is no strong correlation between r_{δ_s} and c_{err}. Finally, Q25 shows a stronger correlation between r_{δ_s} and c_{err} than Q7 and Q19. In contrast to Q14, Q25 has three clusters of robust plan candidates. The estimated most robust plan, the estimated optimal plan and the fastest plan have a small r_{δ_s} and result in a small c_{err}. A majority of other robust plan candidates have a large r_{δ_s} and result in a large c_{err}. Similar to Q7, Q14, and Q19, no plan with a small r_{δ_s} results in a large c_{err} for Q25. The plots of cardinality-slope and cardinality-integral metric look similar to these results, although the cardinality-slope metric has an outlier for Q25, which can be also seen in Figures 10(a) and 11(a).

6.3 Robust Plan Candidates

In this section, we evaluate the impact of the robust plan candidates on execution time and on plan robustness. For the cardinality-slope and selectivity-slope robustness metric, the robust plan candidates are limited to near-optimal plans with $\lambda = 1.2$, whereas the cardinality-integral robustness metric balances costs and plan robustness by definition. Figure 9 shows that robust plan selection with the cardinality-slope or selectivity-slope robustness metric suffers less from estimation errors than conventional plan selection.

Our robust plan selection is an online approach, since it requires low calculation effort for the metric and limits the plan candidates to the k-cheapest plans. Table 1 shows optimization time relative to end-to-end query execution time (i.e., optimization time/query execution time) for both conventional and robust plan selection on both workloads. Since robust plan selection introduces additional computational overhead, this ratio is smaller for conventional plan selection than for robust plan selection. However, the optimization time for robust plan selection is still very small w.r.t. the end-to-end query execution time. The optimization time depends on the number of enumerated plans, i.e., the query graph topology and the number of robust plan candidates.

Table 2: Average $\gamma_{c_{\text{err}}}$ for the Join Order Benchmark (JOB) and the Synthetic Benchmark.

	JOB	Chain	Cycle	Snowflake	Random
cardinality-slope	5.86	−0.05	−0.39	−1.91	−3.04
selectivity-slope	−0.85	−0.08	−0.18	−1.75	−2.62
cardinality-integral	−1.00	−0.04	0.00	−0.48	−0.24

Finally, we demonstrate that selecting the estimated robust plan from k-cheapest plans with $k = 500$ is competitive w.r.t. an estimated robust plan without this limit. As setting $k = \infty$ is infeasible, especially for the complex query graph topologies of some JOB queries, we limit k for this experiment to 10,000. We denote the difference between c_{err} of the estimated most robust plan with $k = 10,000$ and the estimated most robust plan with $k = 500$ as $\gamma_{c_{\text{err}}}$. A negative $\gamma_{c_{\text{err}}}$ indicates that robust plan selection found a more robust plan with a larger k, whereas a $\gamma_{c_{\text{err}}}$ close to 0 indicates that robust plan selection will not find a considerably more robust plan with a larger k.

Table 2 shows the average $\gamma_{c_{\text{err}}}$ for robust plan selection with our three robustness metrics for both, JOB and the synthetic benchmark. For JOB, the average $\gamma_{c_{\text{err}}}$ is close to 0 for the selectivity-slope and cardinality-integral metrics, i.e., a larger k will not yield substantially more robust plans. For the cardinality-slope metric, the average $\gamma_{c_{\text{err}}}$ is even positive. The reason is that robust plan selection with a larger k will choose a plan for Q15 and Q21 that results in a significantly larger c_{err}. For the synthetic benchmark, the average $\gamma_{c_{\text{err}}}$ is close to 0 for all three robustness metrics on chain and cycle queries. For snowflake queries, the average $\gamma_{c_{\text{err}}}$ value is more negative compared to chain and cycle queries due to the larger plan space. Finally, for random queries, the average $\gamma_{c_{\text{err}}}$ is close to 0 for the cardinality-integral robustness metric. In contrast, robust plan selection with the cardinality-slope metric will lead to a $\gamma_{c_{\text{err}}}$ smaller than -1 for 43 of the 100 generated queries, and with the selectivity-slope metric for 57 of the 100 generated queries. Overall, $k = 500$ achieves a good trade-off between plan robustness and query execution time, since for a large number of queries $\gamma_{c_{\text{err}}}$ is close to 0 and the optimization overhead is small.

From our experiments we conclude that FS is more conservative than SS. It achieves more moderate speedups, but also smaller regressions. FI achieves similar results as SS, and supports arbitrary PCF shapes and is independent of $\hat{f}$.

7. CONCLUSION

The three novel robustness metrics presented in this paper are valuable and general building blocks for robust query processing. They efficiently quantify the robustness of query execution plans at optimization time and consider the impact of potential cardinality estimation errors during plan selection. Despite their simplicity, our experimental evaluation has demonstrated the effectiveness of all three robustness metrics. Compared to competitive approaches for robust plan selection, we do not limit the plan topology, can calculate a robustness value for a single plan independent of other plans, and are not bound to expensive statistical models. In the presence of cardinality estimation errors, our comparison of end-to-end query execution times clearly shows that selection of robust plans outperforms conventional plan selection. Finally, our formal specification of the problem and requirements for robustness metrics build a solid foundation for future research on robust query processing.

8. REFERENCES

[1] M. Abhirama, S. Bhaumik, A. Dey, H. Shrimal, and J. R. Haritsa. On the stability of plan costs and the costs of plan stability. *PVLDB*, 3(1-2):1137–1148, 2010.

[2] K. H. Alyoubi. *Database query optimisation based on measures of regret*. PhD thesis, Birkbeck, University of London, 2016.

[3] B. Babcock and S. Chaudhuri. Towards a robust query optimizer: A principled and practical approach. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 119–130. ACM, 2005.

[4] S. Babu, P. Bizarro, and D. DeWitt. Proactive re-optimization. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 107–118. ACM, 2005.

[5] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. Towards estimation error guarantees for distinct values. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '00, pages 268–279. ACM, 2000.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[7] H. Doraiswamy, P. N. Darera, and J. R. Haritsa. Identifying robust plans through plan diagram reduction. *PVLDB*, 1(1):1124–1140, 2008.

[8] A. Dutt and J. R. Haritsa. Plan bouquets: Query processing without selectivity estimation. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 1039–1050. ACM, 2014.

[9] P. Gassner, G. M. Lohman, K. B. Schiefer, and Y. Wang. Query optimization in the IBM DB2 family. *IEEE Data Engineering Bulletin*, 16:4–18, 1993.

[10] G. Graefe, R. Borovica-Gajic, and A. Lee. Robust performance in database query processing (Dagstuhl seminar 17222). *Dagstuhl Reports*, 7(5):169–180, 2017.

[11] G. Graefe, W. Guy, H. A. Kuno, and G. N. Paulley. Robust query processing (Dagstuhl seminar 12321). *Dagstuhl Reports*, 2(8):1–15, 2012.

[12] G. Graefe, A. C. König, H. A. Kuno, V. Markl, and K. Sattler, editors. *Robust Query Processing (Dagstuhl Seminar 10381)*, volume 10381 of *Dagstuhl Seminar Proceedings*, Leibniz-Zentrum für Informatik, Germany, 2010. Schloss Dagstuhl.

[13] R. W. Hamming. *Numerical Methods for Scientists and Engineers*. Dover Publications, Inc., 1986.

[14] A. Hulgeri and S. Sudarshan. Parametric query optimization for linear and piecewise linear cost functions. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 167–178. VLDB Endowment, 2002.

[15] F. Hüske. *Specification and optimization of analytical data flows*. PhD thesis, TU Berlin, 2016.

[16] Y. E. Ioannidis and S. Christodoulakis. On the propagation of errors in the size of join results. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, SIGMOD '91, pages 268–277. ACM, 1991.

[17] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? *PVLDB*, 9(3):204–215, 2015.

[18] V. Leis, B. Radke, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. Query optimization through the looking glass, and what we found running the join order benchmark. *The VLDB Journal*, pages 1–26, 2017.

[19] G. Lohman. Query optimization–are we there yet? In *Datenbanksysteme für Business, Technologie und Web*, BTW '17. Gesellschaft für Informatik, Bonn, 2017.

[20] G. M. Lohman. Is query optimization a solved problem. In *Proceedings of the Workshop on Database Query Optimization*, page 13. Oregon Graduate Center Comp. Sci. Tech. Rep, 2014.

[21] G. Moerkotte, T. Neumann, and G. Steidl. Preventing bad plans by bounding the impact of cardinality estimation errors. *PVLDB*, 2(1):982–993, 2009.

[22] W. Scheufele and G. Moerkotte. On the complexity of generating optimal plans with cross products (extended abstract). In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '97, pages 238–248. ACM, 1997.

[23] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, SIGMOD '79, pages 23–34. ACM, 1979.

[24] F. Wolf, N. May, P. R. Willems, and K.-U. Sattler. On the calculation of optimality ranges for relational query execution plans. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 663–675. ACM, 2018.

[25] S. Yin, A. Hameurlain, and F. Morvan. Robust query optimization methods with respect to estimation errors: A survey. *SIGMOD Record*, 44(3):25–36, 2015.