Characterizing Plan Diagram Reduction Quality and Efficiency

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Master of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

by

Harsh Shrimal

Computer Science and Automation

Indian Institute of Science

BANGALORE – 560 012

June 2009

To my Late Grandfather, whom I will always miss, and my Parents, whose dedication to my success and their continued support,
I shall always remember.

Acknowledgements

I would like to thank my advisor Prof. Jayant Haritsa for helping me to take the first step in the world of scientific research. I would like to sincerely acknowledge his invaluable guidance and encouragement in all forms through out my stay in IISc.

Life wihtout friends in IISc woulld have been difficult and dull, so I would like to thank all my friends in IISc especially Devansh and Vikas. I would also like to thanks my fellow DSLites for providing a stimulating and fun environment for work. Thanks, to all those whom I have not mentioned here, but have helped me in some way or other in completing this project.

Lastly but most importantly, I want to thank my parents and my sister for having faith and confidence in me, for encouraging and supporting me. I would also like to thank my brother-in-law for his unflinching support and encouragement.

Publications

1. M. Abhirama, S. Bhaumik, A. Dey, H. Shrimal and J. Haritsa,
   "*Stability-conscious Query Optimization*",
   Technical Report TR-2009-01, DSL/SERC, Indian Institute of Science,
   http://dsl.serc.iisc.ernet.in/publications/report/TR/TR-2009-01.pdf

Abstract

Estimates of predicate selectivities by database query optimizers often differ significantly from those actually encountered during query execution, leading to poor plan choices and inflated response times. Recently, a novel idea, SEER [1], of mitigating this problem by replacing selectivity error-sensitive plan choices with alternative plans that provide robust performance was proposed. The idea was based on the recent observation [10] that even the complex and dense "plan diagrams" associated with industrial strength optimizers can be efficiently reduced to "anorexic" equivalents featuring only a few plans, without degrading the query processing quality. The SEER algorithm proposed in [1] has a worst case running time complexity of $O(n.r^d)$, where n is the number of plans that feature in the original plan diagram, d is the dimensionality of the plan diagram and $r(>> 4)$ is the resolution of the plan diagram. Thus, though it works well for 2D plan diagrams, for higher dimensional diagrams, the computational overheads incurred by SEER are impractical. In this report, we propose **CC-SEER**, a resolution independent algorithm, with a worst case running time complexity of $O(n.4^d)$. Extensive experimentimation with a representative set of benchmark query templates on commercial optimizers indicates that CC-SEER is an order of magnitude faster than SEER while being competitive on reduction quality and robustness. The CC-SEER algorithm has been implemented in the recently released version 2.0 of Picasso optimizer visualization tool.

A related issue is the universality of the techniques proposed in [1] with regard to optimizer cost models. We investigate this issue and show that SEER/CC-SEER is applicable only to a special breed of optimizers which obey the cost model defined in [1], i.e. it is not feasible to extend these techniques to optimizers with more complex cost models. To address this issue, we propose **WC-SEER**, a cost model scalable technique, based on the concept of *matrix*

condition number. The worst case running time complexity of WC-SEER is $O(n.2^d)$. Experimental results show that WC-SEER outplays the previous techniques on all grounds (reduction quality, robustness and computational efficiency).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The query execution plan choices made by database engines often turn out to be poor in practice because the optimizer's selectivity estimates are significantly in error with respect to the actual values encountered during query execution. Such errors, which can even be in orders of magnitude in real database environments [18], arise due to a variety of reasons [23], including outdated statistics, attribute-value independence assumptions and coarse summaries.

1.1 Robust Plans.

To address this problem, one obvious approach is to improve the quality of the statistical metadata, for which several techniques have been presented in the literature ranging from improved summary structures [2] to feedback-based adjustments [23] to on-the-fly reoptimization of queries [15, 18, 4]. A complementary and conceptually different approach, which we consider in this report, is to identify *robust plans* that are relatively less sensitive to such selectivity errors. In a nutshell, to "aim for resistance, rather than cure", by identifying plans that provide comparatively good performance over large regions of the selectivity space. Such plan choices are especially important for industrial workloads where global stability is as much a concern as local optimality [17].

Over the last decade, a variety of strategies have been proposed to identify robust plans,

including the Least Expected Cost [6, 7], Robust Cardinality Estimation [3] and Rio [4, 5] approaches. These techniques provide novel and elegant formulations (summarized in Section 7), but have to contend with the following issues:

1. They are *intrusive* requiring, to varying degrees, modifications to the optimizer engine.
2. They require *specialized* information about the workload and/or the system which may not always be easy to obtain or model.
3. Their query capabilities may be *limited* compared to the original optimizer – e.g., only SPJ queries with key-based joins were considered in [3, 4]. Further, [4] has been implemented and evaluated on a non-commercial optimizer.
4. Most importantly, as explained in Section 7, none of them provide, on an individual query basis, quantitative *guarantees* on the quality of their final plan choice relative to the original (unmodified) optimizer's selection. That is, they "cater to the crowd, not individuals".

1.2 Anorexic Reduction of Plan Diagrams.

Our techniques are based on the *anorexic reduction of plan diagrams*, a notion that was recently presented and analyzed in [10]. Specifically, a "plan diagram" [21] is a color-coded pictorial enumeration of the plan choices of the optimizer for a parametrized query template over the relational selectivity space. That is, it visually captures the POSP geometry. For example, consider QT8, the parametrized 2D query template shown in Figure 1.2, based on Query 8 of TPC-H. Selectivity variations on the SUPPLIER and LINEITEM relations are specified through the s_acctbal :varies and l_ extendedprice :varies predicates, respectively. The associated plan diagram for QT8 is shown in Figure 1.2(a), produced with the Picasso optimizer visualization tool [20] on a popular commercial database engine.

As evident from Figure 1.2(a)[1], plan diagrams can be extremely complex and dense, with

[1]The figures in this report should ideally be viewed from a color copy, as the grayscale version may not clearly register the features.

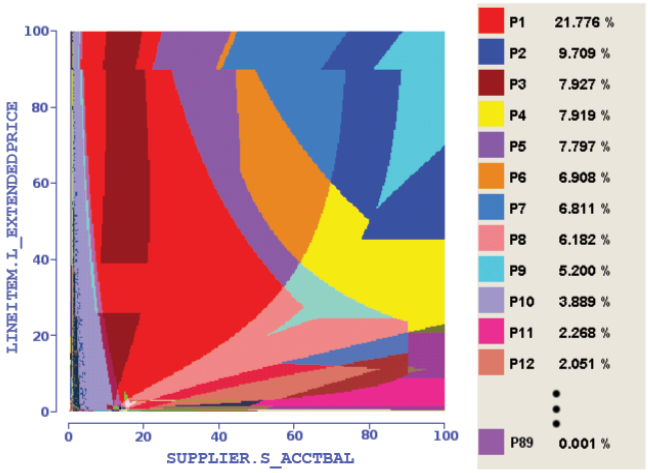
```
select o_year, sum(case when nation = 'BRAZIL' then volume else 0 end) / sum(volume)
from (select YEAR(o_orderdate) as o_year, l_extendedprice * (1 - l_discount) as volume,
      n2.n_name as nation
    from part, supplier, lineitem, orders, customer,
        nation n1, nation n2, region
    where p_partkey = l_partkey and s_suppkey = l_suppkey and l_orderkey = o_orderkey
        and o_custkey = c_custkey and c_nationkey = n1.n_nationkey and n1.n_regionkey
        = r_regionkey and s_nationkey = n2.n_nationkey and r_name = 'AMERICA' and
        p_type = 'ECONOMY ANODIZED STEEL' and
        s_acctbal :varies and l_extendedprice :varies
    ) as all_nations
group by o_year
order by o_year
```

Figure 1.1: **Example Query Template: QT8**

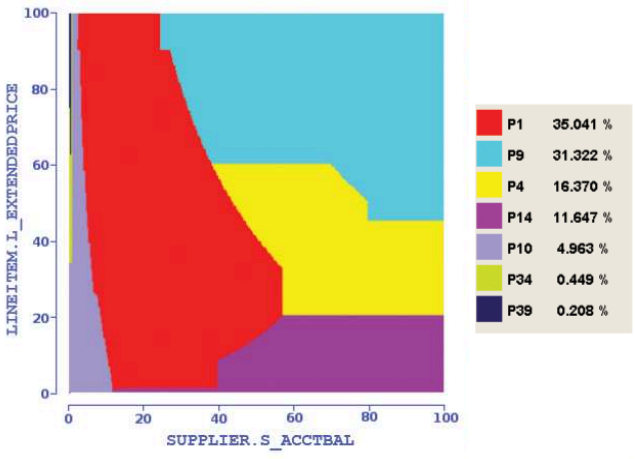
a large number of plans covering the space – several such instances spanning a representative set of benchmark-based query templates on industrial-strength optimizers are available at [20]. However, these dense diagrams can typically be "reduced" to much simpler pictures featuring significantly fewer plans, *without materially degrading the processing quality of any individual query*. For example in Figure 1.2(a), if users are willing to tolerate a minor cost increase (λ) of at most 10% for any query point in the diagram, relative to its original cost, the picture could be reduced to Figure 1.2(b), where only 7 plans remain – that is, most of the original plans have been "completely swallowed" by their siblings, leading to a highly reduced plan cardinality.

A detailed study of the plan diagram reduction problem was presented in [10], and it was shown that a cost increase threshold of *only 20 percent* is usually amply sufficient to bring down the absolute number of plans in the final reduced picture to *within or around ten*. In short, that complex plan diagrams can be made "anorexic" while retaining acceptable query processing performance.

Example. We now show an example of how anorexic reduction helps to identify selectivity-error-resistant plans: In Figure 1.2(a), estimated selectivities of around (14%,1%) lead to a choice of plan P70. However, if the actual selectivities at runtime turn out to be



(a) **Plan Diagram**



(b) **Reduced Diagram (Threshold = 10%)**

Figure 1.2: **Sample Plan Diagram and Reduced Plan Diagram (QT8)**

significantly different, say (50%,40%), executing with P70, whose cost increases steeply with selectivity, would be disastrous. In contrast, this error would have had no impact with the reduced plan diagram of Figure 1.2(b), since P1, the replacement plan choice at (14%,1%), remains the preferred plan for a large range of higher values, including (50%,40%). Quantitatively, at the run-time location, plan P1 has a cost of 135, while P70's cost of 402 is about *three times* more expensive.

It is easy to see, as in the above example, that the replacement plan will, *by definition*, be a robust choice for errors that lie within its optimality region, i.e. its *"endo-optimal"* region. This is the advantage, mentioned earlier, of considering replacements only from the POSP set of plans. The obvious question then is whether the sizes of these regions are typically large enough to materially improve the system performance.

A second, and even more important question, is: What if the errors are such that the run-time locations are *"exo-optimal"* w.r.t. the replacement plan? For example, if the run-time location happens to be at (80%,90%), which is outside the optimality region of P1? In this situation, nothing can be said upfront – the replacement could be much better, similar or much worse than the original plan. Therefore, ideally speaking, we would like to have a mechanism through which one could assess whether a replacement is *globally safe* over the entire parameter space.

1.3 Contributions.

Following are the contibutions made in this report:

- **CC-SEER** We address the scalability concerns of the SEER algorithm proposed in [1]. Specifically, we propose CC-SEER, an order of magnitude faster algorithm with a worst case running time complexity of $O(n.4^d)$ as against $O(n.r^d)$ of SEER, while being comparable with regard to reduction quality and robustness. A novel feature of CC-SEER is that the computational overheads incurred by it are independent of resolution of the plan diagram.

- **Investigating universality of SEER/CC-SEER** The techniques discussed in [1] and CC-SEER are predicated on compliance of the optimizer with the cost model defined in [1]. We investigate the universality of these techniques and show that it is not feasible to extend them beyond simple cost models. As an implication of this, the problem of robust plan identification resurfaces for optimizers with complex cost models containing higher degree polynomials.

- **WC-SEER** We propose a cost model oblivious technique of identifying robust plans. Specifically, we propose WC-SEER, an algorithm based on the concept of matrix condition number. Because of its model oblivious nature, WC-SEER is potentially applicable to any optimizer. The running time complexity of WC-SEER is $O(n.2^d)$. Experimental investigations suggest that WC-SEER outplays CC-SEER on all grounds, i.e. reduction quality, robustness and computational efficiency.

1.4 Organization.

The remainder of this report is organized as follows: In Chapter 2, we present the overall problem background, framework and motivation. The plan cost models and the checks for replacement safety are discussed in Chapter 3. The design of the CC-SEER reduction algorithm and its analysis are presented in Chapter 4. In Chapter 5, we investigate the universality of SEER/CC-SEER. In Section 5.2, we present WC-SEER, a csost model scalable algorithm. Our experimental framework and performance results are highlighted in Section 6. Related work is overviewed in Section 7. Finally, in Section 8, we summarize our conclusions.

Chapter 2

Problem Framework

For ease of exposition, we assume in the following discussion that the SQL query template is 2-dimensional in its selectivity variations – the extension to higher dimensions is straightforward.

2.1 Plan and Reduced Plan Diagrams

From a query template Q, a plan diagram P is produced on a 2-dimensional $[0, 1]$ selectivity space S by making repeated calls to the optimizer. The selectivity space is represented by a grid of points where each point $q(x, y)$ corresponds to a unique query with selectivities x, y in the X and Y dimensions, respectively. Each q is associated with an optimal (as determined by the optimizer) plan P_i, and a cost $c_i(q)$ representing the estimated effort to execute q with plan P_i. Corresponding to each plan P_i is a unique color L_i, which is used to color all the query points that are assigned to P_i. As mentioned earlier, the plan diagram is essentially a visual characterization of the parametric optimal set of plans (POSP) [12]. We use P and S interchangeably in the remainder of the report based on the context.

2.1.1 Plan Diagram Reduction Problem

. This problem is defined as follows [10]: Given an input plan diagram P, and a maximum-cost-increase threshold λ ($\lambda \geq 0$), find a reduced plan diagram R with *minimum cardinality* such that for every plan P_i in P,

1. Either $P_i \in \mathsf{R}$, or
2. $\forall\, q \in P_i$, the assigned replacement plan $P_j \in \mathsf{R}$ guarantees $\frac{c_j(q)}{c_i(q)} \leq (1+\lambda)$

That is, find the maximum possible subset of the plans in P that can be completely "swallowed" by their sibling plans in the POSP set. A point worth reemphasizing here is that the threshold constraint applies on an *individual query* basis. For example, setting $\lambda = 10\%$ stipulates that the cost of *each* query point in the reduced diagram is within 1.1 times its original value.

It was proved in [10] that the above problem is NP-Hard. Therefore, an efficient heuristic-based online algorithm, called **CostGreedy**, was proposed and shown to deliver near-optimal "anorexic" levels of reduction, wherein the plan cardinality of the reduced diagram usually came down to around 10 or less for a λ-threshold of only 20%. In a nutshell, complex plan diagrams can be easily made very simple without materially affecting the query processing quality.

2.1.2 Selectivity Estimation Errors

Consider a specific query point q_e, whose optimizer-estimated location in S is (x_e, y_e). Denote the optimizer's optimal plan choice at point q_e by P_{oe}. Due to errors in the selectivity estimates, the *actual* location of q_e could be different at execution-time – denote this location by $q_a(x_a, y_a)$, and the optimizer's optimal plan choice at q_a by P_{oa}. Assume that P_{oe} has been swallowed by a sibling plan during the reduction process and denote the replacement plan assigned to q_e in R by P_{re}. Finally, extend the definition of query cost (which applied to the optimal plan) to have $c_i(t)$ denote the cost of an arbitrary POSP plan P_i at an arbitrary query point t in S.

With respect to R, the actual query point q_a will be located in one of the following disjoint regions of P_{re} that together cover S:

Endo-optimal region of P_{re}: Here, q_a is located in the optimality region of the replacement plan P_{re}, which also implies that $P_{re} \equiv P_{oa}$. Since $c_{re}(q_a) \equiv c_{oa}(q_a)$, it follows that the cost of P_{re} at q_a, $c_{re}(q_a) < c_{oe}(q_a)$ (by definition of a cost-based optimizer). Therefore, improved resistance to selectivity errors is always *guaranteed* in this region.

Swallow-region of P_{re}: Here, q_a is located in the region "swallowed" by P_{re} during the reduction process. Due to the λ-threshold constraint, we are assured that $c_{re}(q_a) \leq (1+\lambda)c_{oa}(q_a)$, and by implication that $c_{re}(q_a) \leq (1+\lambda)c_{oe}(q_a)$. Now, there are two possibilities: If $c_{re}(q_a) < c_{oe}(q_a)$, then the replacement plan is again guaranteed to improve the resistance to selectivity errors. On the other hand, if $c_{oe}(q_a) \leq c_{re}(q_a) \leq (1+\lambda)c_{oe}(q_a)$, the replacement is guaranteed to not cause any real harm, given the small values of λ that we consider in this report.

Exo-optimal region of P_{re}: Here, q_a is located outside both the endo-optimal and swallow-regions of P_{re}. At such locations, we cannot apriori predict P_{re}'s behavior, and therefore the replacement may not always be a good choice – in principle, it could be *arbitrarily worse*. Therefore, we would like to ensure that even if the replacement does not provide any improvement, it is at least guaranteed to not do any harm. That is, the *exo-optimal region should have the same performance guarantees as the swallow-region*. We show in Chapter 3 how this objective can be efficiently achieved through simple but powerful checks to decide when replacement is advisable.

2.2 Robust Reduction

From the above discussion, it is clear that we need to ensure that only safe replacements are permitted. This means that replacement should be permitted only if the λ threshold criterion is satisfied not just at the estimated point, but *at all locations* in the selectivity space. At the same time, it is important to ensure that the safety check is not unnecessarily conservative, preventing most plan replacements, and in the process losing all the error-resistance benefits. Therefore, the overall goal is to maximize plan diagram reduction without violating safety considerations. More formally, our problem formulation is:

2.2.1 Robust Reduction Problem.

Given an input plan diagram P, and a maximum cost-increase-threshold λ ($\lambda \geq 0$), find a reduced plan diagram R with *minimum plan cardinality* such that for every plan P_i in P,

1. $P_i \in$ R, or

2. $\forall\ q \in P_i$, the assigned replacement plan $P_j \in R$ guarantees $\forall$ query points $q' \in$ P, $\frac{c_j(q')}{c_i(q')} \leq (1+\lambda)$

That is, find the minimum-sized error-resistant "cover" of plans that reduces the plan diagram P without increasing the cost of any reassigned query point by more than the cost increase threshold, *irrespective of the actual location of the query at run-time*.

It is easy to see that the Robust Reduction problem is NP-Hard, just like the standard Plan Diagram Reduction problem, and therefore we present a heuristic-based algorithm later in Chapter 4. But, prior to that, we show in the following section how replacement safety can be checked efficiently.

Chapter 3

Ensuring Robust Reduction

To find an error-resistant cover of the plan diagram, we need to evaluate the behavior of each replacement plan P_{re}, w.r.t. its swallowing target P_{oe}, at *all points* in S. This requires, in principle, finding the costs of P_{oe} and all potential P_{re} at every point in the diagram. Of course, P_{oe} and P_{re} need not be costed in their respective *endo-optimal* regions, since these values are already known through the plan diagram production process. The remaining *exo-optimal* costs can be obtained using the *Foreign-Plan-Costing* feature, hereafter referred to as **FPC**, a feature that has become available in the current versions of several industrial-strength optimizers, including DB2[30] (Optimization Profile), SQL Server[31] (XML Plan) and Sybase[32] (Abstract Plan).

While the above solution is conceptually feasible, it is practically unviable due to its enormous computational overheads. Plan-costing is certainly cheaper than the optimizer's standard optimal-plan-searching process [13], but the overall overhead is still $O(nm)$ where n and m are the number of plans and the number of points, respectively, in P. Typical values of n range from the several tens to several hundreds, while m is of the order of several thousands to several hundreds of thousands, making an exhaustive approach impractical.

The above situation motivates us to study whether it is possible, based on using FPC at only a few select locations, to *infer* the behavior in the rest of the space. In the remainder of this section, we describe our strategy for making such an inference. It was shown in [1] that it

is possible to characterize plan cost behavior using a simple and accurate parametrized mathematical model. The accuracy of the model was substantiated with several hundred distinct plans arising out of TPC-H and TPC-DS-based query templates on industrial optimizers.

3.1 Modelling Plan Cost Functions

The following cost model was proposed in [1] for characterizing plan cost behavior in a $d-$dimensional selectivity space :

$$\begin{aligned} Cost(x_1, ..., x_d) = &\sum_{i_1} (a_{i_1} x_{i_1} + b_{i_1} x_{i_1} \log x_{i_1}) + \\ &\sum_{i_1 < i_2} (a_{i_1 i_2} x_{i_1} x_{i_2} + b_{i_1 i_2} x_{i_1} x_{i_2} \log x_{i_1} x_{i_2}) \\ &+ ... + a_{12..d}(x_1 x_2 x_3 .. x_d) \\ &+ b_{12..d}(x_1 x_2 x_3 .. x_d) \log(x_1 x_2 x_3 .. x_d) \\ &+ a_0 \end{aligned} \tag{3.1}$$

where the a's and b's are the $(2^{d+1} - 1)$ coefficients and the $x_i, i = 1...d$ represent the d relational selectivities.

For instance, the cost model of a plan for a 2D selectivity space is of the form

$$\begin{aligned} Cost(x, y) = &a_1 x + a_2 y + a_3 xy + a_4 x \log x + a_5 y \log y + \\ &a_6 xy \log xy + a_7 \end{aligned} \tag{3.2}$$

where $a_1, a_2, a_3, a_4, a_5, a_6, a_7$ are coefficients, and x, y represent the selectivities of R_x and R_y, respectively.

Modeling a specific plan requires suitably choosing the seven coefficients, and this is achieved through standard surface-fitting techniques, described in [1]

3.2 Replacement Safety Conditions

For ease of presentation, we will initially assume that our objective is to model the cost behavior of plans with respect to a 2-D selectivity space (e.g. Figure 1.2(a)) corresponding to distinct relations R_x and R_y. The extension to higher dimensions is straightforward and is discussed later.

For the 2D scenario, using the above 7-coefficient cost model, our goal now is to come up with an efficient mechanism to assess, given an optimal plan P_{oe}, candidate replacement plan P_{re} and a cost-increase threshold λ, whether it would be safe from a *global* perspective to have P_{re} swallow P_{oe}.

Let the cost functions for P_{re} and P_{oe} be

$$f_{re}(x, y) = a_1x + a_2y + a_3xy + a_4x \log x + a_5y \log y + a_6xy \log xy + a_7 \tag{3.3}$$

and

$$f_{oe}(x, y) = b_1x + b_2y + b_3xy + b_4x \log x + b_5y \log y + b_6xy \log xy + b_7 \tag{3.4}$$

respectively. Now consider the **"safety function"**

$$f(x, y) = f_{re} - (1 + \lambda)f_{oe} \tag{3.5}$$

which captures the differences between the costs of P_{re} and a λ-inflated version of P_{oe} in the selectivity space. All points where $f(x, y) \leq 0$ are referred to as *SafePoints* whereas points that have $f(x, y) > 0$ are called *ViolatingPoints*. For a replacement to be globally safe, there should be no ViolatingPoints anywhere in the selectivity space.

For a specific value of y, the safety function $f(x, y)$ can be rewritten as

$$f_y(x) = g_1 * x + g_2 * x \log x + g_3$$

for appropriate coefficients g_1, g_2, g_3. Similarly, we can define $f_x(y)$. With this terminology, the following theorem provides us with conditions for checking whether the selectivity space is safe for the plan-pair (P_{oe},P_{re}) with regard to replacement.

Consider the following two lemmas, borrowed from [1], pertaining to safety function behavior – the first provides us with a condition that is sufficient to ensure safety of all points on the straight line segment joining a pair of safe points, while the second describes the behaviour of the slope. i.e. second derivative, of the safety function.

LEMMA 3.1 (Line Safety). *Given a fixed $y = y_o$, and a pair of safe points (x_1, y_o) and (x_2, y_o) with $x_2 > x_1$, the straight line joining the two points is safe if the slope $f'_{y_o}(x)$ is either (i) monotonically non-decreasing, OR (ii) strictly decreasing with $f'_{y_o}(x_1) \leq 0$ or $f'_{y_o}(x_2) \geq 0$. A similar result holds when x is fixed.*

Note that the slope value at a point (x_i, y_i) along a dimension can be approximated as the slope of the line joining that point to the next point on that particular dimension. For instance, $f'_{y_0}(x_1)$ can be approximated with $\dfrac{f(x_1, y_0) - f(x_2, y_0)}{x_1 - x_2}$

LEMMA 3.2 (Second Derivative Behavior). *If the slope of the safety function, $f'_y(x)$, is non-decreasing (resp. decreasing) along the line-segments $y = y_1$ and $y = y_2$, then it is non-decreasing (resp. decreasing) for all line segments in the interval (y_1, y_2). A similar result holds for $f'_x(y)$.*

In addition to the above two lemmas, we prove another lemma pertaining to behavior of first derivative of the safety function. CC-SEER critically draws much of its performance from this lemma.

LEMMA 3.3 (First Derivative Behavior). *Given a fixed $y = y_0$, let $g(x) = \left(\dfrac{\partial f(x,y)}{\partial y}\right)_{y_0}$. Now, given a pair of points (x_1, y_0) and (x_2, y_0) with $x_1 < x_2$ such that $g(x_1) \leq 0$ and $g(x_2) \leq 0$, $\forall x \in [x_1, x_2],\ g(x) \leq 0$ if the slope $g'(x)$ is either (i) monotonically non-decreasing, i.e. $g'(x_1) \leq g'(x_2)$, OR (ii) strictly decreasing with $g'(x_1) \leq 0$ or $g'(x_2) \geq 0$. A similar result holds when y is fixed.*

Lemma 3.3 basically says that by computing the value of $g'(x_1)$ and $g'(x_2)$ it is possible to determine whether or not $\forall x \in [x_1, x_2],\ g(x) \leq 0$. Similarly, it is also possible to determine whether or not $\forall x \in [x_1, x_2],\ g(x) \geq 0$, by just computing the values of $g'(x_1)$ and $g'(x_2)$. Slope approximation similar to that mentioned for computing the value of first derivative of the safety function at a point can be employed here also. For instance, $g'(x_1)$ can be approximated with $\dfrac{f'_{x_1}(y_0) - f'_{x_2}(y_0)}{x_1 - x_2}$.

Chapter 4

The CC-SEER Algorithm

In this section, we will describe the safety checking procedure, which give a plan-pair (P_{oe}, Pre), responds whether the replacement of P_{oe} by P_{re} is globally safe throughout the selectivity space **S**. We then present and analyze the CC-SEER algorithm which uses this procedure to do error-resistant plan diagram reduction.

In the following, we will assume that the selectivity space **S** is represented by a grid **G**, with $m = r \times r$ points, i.e. the grid resolution in each dimension is r.

4.1 Safety Checking

Let the points on X axis of **G** be labelled $x_1, x_2, \cdots, x_r$ in ascending order of selectivity (Fig. 4.1). Similarly, let Y-axis points be labelled $y_1, y_2, \cdots, y_r$. Leveraging the Lemmas 3.1, 3.2 and 3.3, we use the following safety tests:

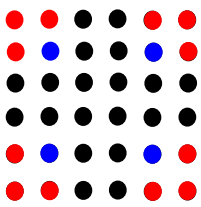


Figure 4.1: **2D Grid of Selectivity Points**

1. **Wedge Test**

   - Ensure safety at corners points of **G**, i.e. confirm that the safety function is non-positive at points: $(x_1, y_1),\ (x_r, y_1),\ (x_1, y_r), (x_r, y_r)$. If coner point are safe, goto next step. Else, return **Unsafe**.
   - If $f'_{y_1}(x_1) \leq f'_{y_1}(x_r)$ AND $f'_{y_r}(x_1) \leq f'_{y_r}(x_r)$, return **Safe**. Else, goto next step.
   - If $f'_{x_1}(y_1) \leq f'_{x_1}(y_r)$ AND $f'_{x_r}(y_1) \leq f'_{x_r}(y_r)$, return **Safe**. Else, return **Inconclusive**

     The wedge test corresponds to FPC operations at the points denoted by red color in Fig. 4.1. Thus, the total number of FPC operations done in the wedge test is atmost 24.

2. **CornerCube Test**

   - If $f'_{y_1}(x),\ f'y_r(x)$ are both strictly decreasing, use Lemma 3.2 to ensure that $\forall y \in [y_1, y_r]$, either (i) $f'_y(x_1) \leq 0$, OR (ii) $f'_y(x_r) \geq 0$. If true, return **Safe**. Else, goto next step.
   - If $f'_{x_1}(y),\ f'x_r(y)$ are both strictly decreasing, use Lemma 3.2 to ensure that $\forall x \in [x_1, x_r]$, either (i) $f'_x(y_1) \leq 0$, OR (ii) $f'_x(y_r) \geq 0$. If true, return **Safe**. Else, return **Inconclusive**.

   The cube test corresponds to the red and blue points in Fig. 4.1. Thus, the total number of FPC operations done in the cube test, in addition to the 24 done in the wedge test, is atmost 8. Note that inconclusive responses from the safety checking procedure are conservatively deemed UNSAFE.

The complete CC-SEER algorithm is given in Fig. 4.2. The algorithm, for each plan pair, calls the wedge test, and the cornercube test is called iff the wedge tests responds 'inconclusive'.

CC-SEER (Plan Diagram P, Threshold λ)

1. Create a Set-Cover Instance $I = (U, S)$, where $S = \{S_1, S_2, ..., S_n\}$, $U = \{1, 2, ..., n\}$, corresponding to the n plans in the original plan diagram P.
2. Set each $S_i = \{i\}, \forall i = 1...n$
3. For each pair of plans (P_i, P_j) do

   if (WEDGE_TEST (P_i,P_j,λ) == Unsafe) then

   $Continue$

   if (WEDGE_TEST (P_i,P_j,λ) == Safe) then

   $S_i = S_i \bigcup \{j\}$

   else if (WEDGE_TEST (P_i,P_j,λ) == Inconlusive AND
   CORNERCUBE_TEST (P_i,P_j,λ) == Safe) then

   $S_i = S_i \bigcup \{j\}$
4. Solve the set-cover instance I using the Greedy Setcover algorithm to identify the plans retained in R.

Figure 4.2: **The CC-SEER Reduction Algorithm**

4.1.1 Extension of CC-SEER to Higher Dimensions

In this section, we will chalk out the generic CC-SEER safety checking procedure. Given an plan pair (P_{oe}, P_{re}), the CC-SEER safety checking procedure responds whether the replacement of P_{oe} with P_{re} is globally safe throughout the selectivity space **S**.

In the following, we will assume that the selectivity space is represented by a d-dimensional grid **G**. The resolution on each dimension is r. Let $\mathbf{X_1}, \mathbf{X_2}, \cdots, \mathbf{X_d}$ be the d dimensions of G. Let $x_{i1}, x_{i2}, \cdots, x_{ir}$ be the selectivity values in the dimension $\mathbf{X_i}$ in ascending order of selectivity.

Let $\varphi : \mathbf{S} \rightarrow \mathbb{R}$ be a *target function* that assigns each selectivity point in **G** with some real number. A target function is said to be **IDEAL** if it is of the form given by Equation 3.1.

CC-SEER uses an auxillary recursive procedure, **IsNonPositive** (G, d, φ), which when given a d-dimensional grid **G** and an ideal target function φ as input, responds whether or not $\forall s \in S, \varphi(s) \leq 0$. An importannt point worth remphasising here is that this procedure is based on the Lemmas 3.1, 3.2 and 3.3.

CC-SEER (Plan Diagram P, Threshold λ)

1. Create a Set-Cover Instance $I = (U, S)$, where $S = \{S_1, S_2, ..., S_n\}$, $U = \{1, 2, ..., n\}$, corresponding to the n plans in the original plan diagram P.
2. Set each $S_i = \{i\}, \forall i = 1...n$
3. For each pair of plans (P_i, P_j) do

   let $f(.)$ be the safety function pertaining to the replacement of P_j with P_i.

   if IsNonPositive $(\mathbf{G}, d, f) ==$ Yes

   $S_i = S_i \bigcup \{j\}$
4. Solve the set-cover instance I using the Greedy Setcover algorithm to identify the plans retained in R.

Figure 4.3: **The Generic CC-SEER Reduction Algorithm**

4.1.2 Analysis

The total number of FPC operations required by CC-SEER for a d-dimensional selectivity space is $n.4^d$. The selectivity points at which the safety function is evaluated correspond to unit hypercubes at the corners of the selectivity space, that is why the name CC-SEER (CornerCube-SEER). In any robust reduction algorithm, FPC operations constitute the running time bottleneck. Thus, the worst case runnning time complexity of CC-SEER is $O(n.4^d)$.

IsNonPositive (G, d, φ)

1: **if** d == 0 **then**
2: $\mathbf{G}$ represent a single selectivity point. Let us call this point s.
3: **if** $\varphi(s) \leq 0$ **then**
4: **return** Yes
5: **else**
6: **return** No
7: **end if**
8: **end if**
9: **for** $i = 1$ to d **do**
10: Let $\mathbf{G_1}$ and $\mathbf{G_2}$ be the $d-1$ dimensional selectivity grids given by $\mathbf{X_i} = x_{i1}$ and $\mathbf{X_i} = x_{ir}$ respectively.
11: **if** IsNonPositive $(\mathbf{G_1}, d-1, \varphi) ==$ No **then**
12: **return** No
13: **end if**
14: **if** IsNonPositive $(\mathbf{G_2}, d-1, \varphi) ==$ No **then**
15: **return** No
16: **end if**
17: Let $\varphi : \mathbf{S} \rightarrow \mathbb{R}$ be a new ideal target function defined as: $\varphi'(s) = \dfrac{\partial \varphi(s)}{\partial \mathbf{X_i}}$
18: Also, let $\varphi''(s) = \dfrac{\partial \varphi'(s)}{\partial \mathbf{X_i}}$.
19: **if** φ'' is positive at the corners of $\mathbf{G_1}$ and $\mathbf{G_2}$ **then**
20: **return** Yes
21: **end if**
22: **if** IsNonPositive $(\mathbf{G_1}, d-1, \varphi') ==$ Yes OR IsNonPositive $(\mathbf{G_2}, d-1, -\varphi') ==$ Yes **then**
23: **return** Yes
24: **else**
25: Continue
26: **end if**
27: **end for**
28: **return** No

Figure 4.4: **The CC-SEER Safety Checking Procedure**

Chapter 5

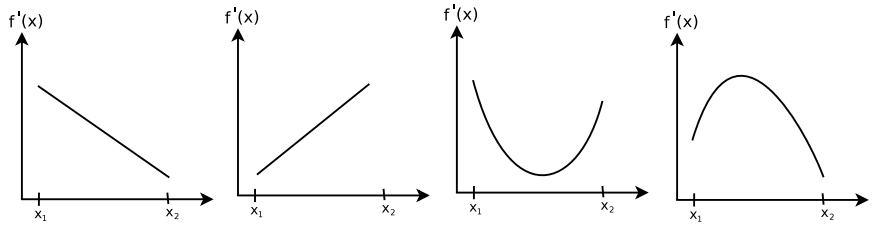
Richer Cost Models

The CC-SEER algorithm, wherein peripheral behavior determines global safety, are applicable only to the class of query optimizers which obey the cost model defined in [1], thereby limiting the applicability of the technique. Moreover, with advent of new operators, even the compatible optimizers might cease to be so. For instance, implementing new operators (e.g. skyline operator) might introduce a square term in the cost model. Hence, it becomes essential to investigate the problem of robust plan identification for cost models containing higher degree polynomials. In this report, we present a characterization of optimizer cost models with respect to applicability of the CC-SEER approach. Consequently, we identify the richest class of optimizers beyond which extending this technique is not feasible without incurring unviable computational overheads.

5.1 A Motivating Example

Let us consider the following cost model which contains a square term:

$$\begin{aligned} Cost(x,y) =& a_1 x + a_2 y + a_3 xy + a_4 x \log x + a_5 y \log y + \\ & a_6 xy \log xy + a_7 x^2 + a_8 y^2 + a_9 (xy)^2 \end{aligned} \tag{5.1}$$



(a) **Non-Increasing** (b) **Non-Decreasing** (c) **One Local Maxima** (d) **One Local Minima**

Figure 5.1: **Safety Function- First Derivative Behavior**

$$Cost(x) = c_1 x \ + \ c_2 x \log x \ + \ c_3 x^2 \ + \ c_4$$

The safety function is defined as $f(x, y) = Cost_{P_{re}}(x, y) \ - \ (1 + \lambda) Cost_{P_{oe}}(x, y)$.

For a particular value of y, resulting in a 1D selectivity space, the safety function $f(x, y)$ can be rewritten as

$$f_y(x) = g_1 x \ + \ g_2 x \log x \ + \ g_3 x^2 \ + \ g_4$$

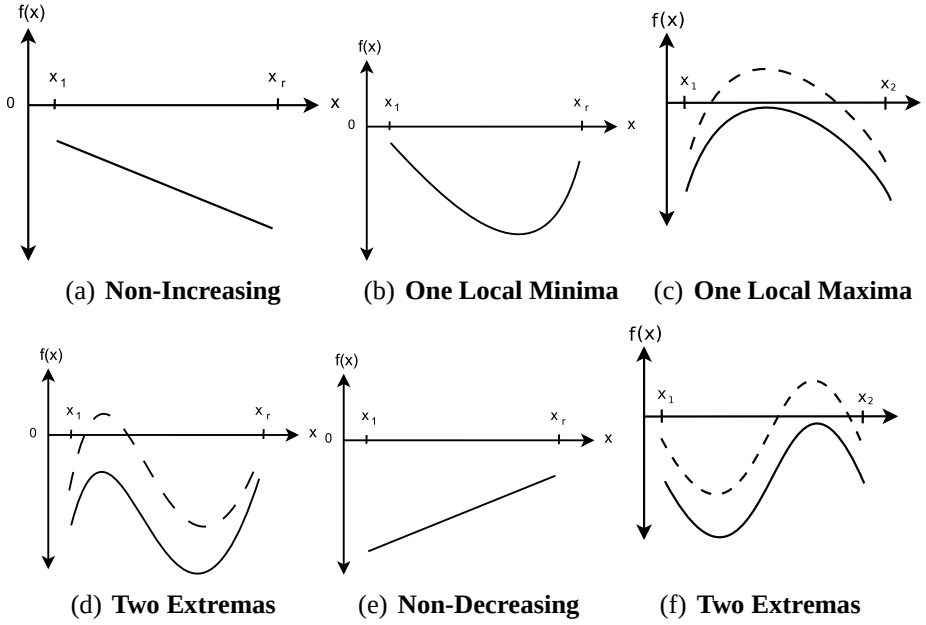
$$\Rightarrow f'_y(x) = g_1 + g_2(1 + \log x) + 2g_3 x$$

$$\Rightarrow f''_y(x) = \frac{g_2}{x} + 2g_3$$

$$\Rightarrow f'''_y(x) = \frac{-g_2}{x^2}, \text{ i.e. } f'''(x) \text{ is monotonic}$$

Our goal is to investigate global safety, i.e. safety at all selectivity points in the 1D selectivity space, of P_{re} for P_{oe} given the fact that P_{re} is safe for P_{oe} at x_1 and x_r. In other words, we address the following question : "does the safety function $f_y(x)$ attain local maxima at any intermediate point x^*". If not, then global safety can be guaranteed just by ensuring safety at the end points. On the other hand, in case $f_y(x)$ does attain local maxima at some selectivity point $x^* \in (x_1, x_r)$, global safety of P_{re} for P_{oe} depending on the value of $f_y(x^*)$. However, in such an event, there is no efficient mechanism to determine the value of $f_y(x^*)$. Therefore, we are bound to conservatively deny global safety.

The various possible behaviors of $f(_y(x)$ are shown in Fig. 5.2. For instance, Fig. 5.2(a) says that the safety function is non-negative and non-increasing as well. The various possible



(a) **Non-Increasing** (b) **One Local Minima** (c) **One Local Maxima**

(d) **Two Extremas** (e) **Non-Decreasing** (f) **Two Extremas**

Figure 5.2: **Safety Function Behavior**

behaviors of $f'_y(x)$ with regard to "number of extremas" attained by it in (x_1, x_r), for this cost model, are shown in Figure 5.1. We hasten to clarify that Fig. 5.1 only captures the possibilities with regard to the gradient of $f'_y(x)$. For instance, Figure 5.1(a) essentially says that $f'_y(x)$ in this case is non-decreasing, and not that $f'_y(x)$ is non-negative. Now, the question posed above can be categorically addressed by looking at the values of $f'_y(x_1)$ and $f'_y(x_r)$.

Case 1: $f'_y(x_1) > 0$, $f'_y(x_r) < 0$

The safety function attains local maxima at an intermediate point, as shown in Figure 5.2(c). Hence, safety at end points does not gurantee global safety. We conservatively deny global safety.

Case 2: $f'_y(x_1) < 0$, $f'_y(x_r) > 0$

The safety function attains local minima (but never local maxima) at an intermediate point, as shown in Figure 5.2(b). Terminal safety therefore gurantees global safety.

Case 3: $f'_y(x_1) \leq 0$, $f'_y(x_r) \leq 0$

The safety function will either be non-increasing, as shown in Fig. 5.2(a), or will attain two

extremas, as shown in Fig. 5.2(f). The latter behavior of $f_y(x)$ corresponds to $f'(x)$ attaining a local maxima at some intermediate point x^*, as shown in Fig. 5.1(d), such that $f'_y(x^*) > 0$. However, finding out whether or not $f'_y(x)$ attains local maxima is not possible without computing the value of second or higher derivatives of $f_y(x)$ (for e.g. $f''_y(x_1)$). The important point here is that it is not possible to efficiently compute or approximate such values. We therefore, conservatively deny safety.

Case 4: $f'_y(x_1) \geq 0$, $f'_y(x_r) \geq 0$

The safety function will either be non-decreasing, as shown in Fig. 5.2(d), or will attain two extremas, as shown in Fig. 5.2(d). The latter behavior of $f_y(x)$ corresponds to $f'_y(x)$ attaining a local minima at som intermediate point x^*, as shown in Fig. 5.1(c), such that $f'_y(x^*) < 0$. For reasons similar to those in Case 3, we conservatively deny global safety.

Note that with the cost model defined in [1], wherein $f''_y(x)$ is monotonic, we would have been able to guarantee safety in Cases 3 and 4.

Extending our investigation to a more complex cost model, consider the following cost model containing a cubic term:

$$\begin{aligned} Cost(x,y) =& a_1x + a_2y + a_3xy + a_4x\log x + a_5y\log y + \\ & a_6xy\log xy + a_7x^2 + a_8y^2 + a_9(xy)^2 + \\ & a_{10}x^3 + a_{11}y^3 + a_{12}(xy)^3 \end{aligned} \tag{5.2}$$

The safety function for this model can have upto two extremas in (x_1, x_r). The conlusions and reasonings with regard to global safety would remain the same as above, in Cases 1, 3 and 4. However in Case 2, unlike the simpler one defined in (5.1), global safety has to be denied. This is because of the fact that $f'_y(x)$ can attain two extremas (local maxima followed by local minima) leading to the situation where $f_y(x)$ can attain local maxima. It is again hard to determine whether or not such a situation occurs without knowing the values of higher derivatives of $f_y(x)$. Thus, for this cost model, the CC-SEER based approach would always deny global safety.

5.1.1 Characterizing Cost Models

It is possible to characterize various cost models with respect to first derivative of the safety function, i.e. $f'_y(x)$. Let k be the maximum number of extremas that $f'_y(x)$ can possibly attain in (x_1, x_r), i.e. $f'_y(x)$ can attain k, but never more than k, extremas in (x_1, x_r). With such a characterization of optimizer cost models, it can be shown that if $k > 2$ for a given optimizer cost model then it is infeasible to extend the CC-SEER approach to such an optimizer. Note that $k = 0$ for the optimizer considered in [1]. Thus, the CC-SEER approach will not work for the cost models with $k > 2$.

5.2 WC-SEER: A Cost Model Scalable Technique

In light of the discussion in Chapter 5, we propose a technique of identifying robust plans, *Well Conditioned SEER* (WC-SEER), which in principle, because of its cost model scalable nature, is not confined to a special breed of query optimizers. Specifically, we propose an efficient and practically accurate technique of identifying robust plans, based on the concept of **condition number** of a matrix. Unlike SEER and CC-SEER, which rely on the cost behavior of plans on the periphery of the selectivity space to deduce global safety, this technique, given an upper bound f on the number of foreign plan costings, tries to capture the notion: "which f selectivity points when used to learn the cost function of an execution plan, lead to the most accurate fit". In essence, WC-SEER does not necessarily restrict itself to the periphery of the selectivity space.

5.2.1 Condition Number of a Matrix

The condiiton number is a measure of stability or sensitivity of a matrix (or the linear system it represents) to numerical operations. The condition number of a square matrix A is defined by

$$\kappa(A) = \begin{cases} \|A\|\|A^{-1}\| & : |A| \neq 0 \\ \infty & : \text{ otherwise} \end{cases}$$

where $\|.\|$ denotes the norm of a matrix.

The condition number of a singular matrix is alternatively defined as the ratio of its largest eigen value to smallest eigen value. The condition number forms the basis of the following well known theorem from linear algebra [24]

THEOREM 1. *Given an equation of the form $Ax = b$ and that the measurement of b is inexact, the relative error in the solution $x = A^{-1}b$ satisfies*

$$\frac{\|\delta x\|}{\|x\|} \leq c\frac{\|\delta b\|}{\|b\|}$$

where c is the condition number of matrix A.

Matrices with condition numbers near 1 are said to be *well-conditioned*, i.e. stable, whereas those with large condition numbers (e.g. 10^5 for a 5×5 Hilbert matrix [25]) are said to be *ill-conditioned*, i.e. highly sensitive. Matrices with condition number around 10 are considered to be reasonably stable for all practical purposes.

5.2.2 Efficiently Determining the Plan Cost Functions

In this section, we describe an efficient mechanism for determining the parametric coefficients of a given query execution plan. For the purpose of illustration, let us the consider the optimizer studied in [1]. However, as it will turn out, the effectiveness of our technique is independent of the optimizer cost model. Also, let us assume that the plan diagram under consideration is 2-dimensional. The extension to higher dimensions is straightforward. The cost model of a plan for a 2D selectivity space is of the form,

$$\begin{aligned} Cost(x, y) =& a_1 x + a_2 y + a_3 xy + a_4 x \log x + a_5 y \log y + \\ & a_6 xy \log xy + a_7 \end{aligned} \tag{5.3}$$

Consider the system of linear equations given by,

$$XA = C \tag{5.4}$$

where,

$$X = \begin{pmatrix} x_1 & y_1 & x_1y_1 & x_1\log x_1 & y_1\log y_1 & x_1y_1\log x_1y_1 & 1 \\ x_2 & y_2 & x_2y_2 & x_2\log x_2 & y_2\log y_2 & x_2y_2\log x_2y_2 & 1 \\ x_3 & y_3 & x_3y_3 & x_3\log x_3 & y_3\log y_3 & x_3y_3\log x_3y_3 & 1 \\ x_4 & y_4 & x_4y_4 & x_4\log x_4 & y_4\log y_4 & x_4y_4\log x_4y_4 & 1 \\ x_5 & y_5 & x_5y_5 & x_5\log x_5 & y_5\log y_5 & x_5y_5\log x_5y_5 & 1 \\ x_6 & y_6 & x_6y_6 & x_6\log x_6 & y_6\log y_6 & x_6y_6\log x_6y_6 & 1 \\ x_7 & y_7 & x_7y_7 & x_7\log x_7 & y_7\log y_7 & x_7y_7\log x_7y_7 & 1 \end{pmatrix} \tag{5.5}$$

is a 7×7 **location matrix**.

$$A = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} \tag{5.6}$$

is the to be determined 7×1 **coefficient vector** for a given plan P, and,

$$C = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} \tag{5.7}$$

is the 7×1 **cost vector**. Here, c_i is the actual cost of plan P at the selectivity point (x_i, y_i).

Let $\{s_i = (x_i, y_i)$: $i = 1$ to $7\}$ be some seven selectivity points in the 2D selectivity space. We can find the cost of a given plan P at these seven selectivity points using the *foreign plan costing* feature now available in all major industrial strength optimizers. Thus the matrices X and C are known to us. If the cost model decribed in (5.3) is a perfect fit for plan P, i.e. $Cost_P(x_i, y_i) = c_i$, then the seven parametric coefficients for P can be obtained by solving the linear system of equations given by (5.4) , i.e. $A = X^{-1}C$. However, the cost model is least likely to be a perfect fit for any query execution plan. Nonetheless, we know from [1] that this cost model is an almost perfect fit for all execution plans, i.e. the RMS error is very small.

Let $\Delta_i = Cost(x_i, y_i) - c_i$ be the residual error between the actual cost and fitted cost of plan P at selectivity point (x_i, y_i). Let $\mathbf{\Delta} = [\Delta_i]$ be the **residual vector** for plan P wrt the mentioned selectivity points. We know that the vector $\mathbf{\Delta}$ consists of very small values, i.e. Δ_i is very close to zero for all i. Now, let us ask the question that how big is the difference between $X^{-1}C$ and $X^{-1}(C + \mathbf{\Delta})$ is going to be, given that the vector $\mathbf{\Delta}$ consists of values very close to zero. The answer to this question depends on the condition number of matrix X. If X is a well-conditioned matrix then this difference between the two solutions will be very small. Thus, our task boils down to finding a set of seven selectivity points in the 2D selectivity space which correspond to a well-conditioned location matrix.

Fig. 5.2.3 gives the pseudocode listing for WC-SEER (for 2D diagrams).

5.2.3 Minimizing Condition Number

In previous subsection, we elaborated on how minimizing the condition number of location matrix can be leveraged to accurately, and without incurring huge overheads, learn the plan cost functions. In this subsection, we talk about the computaional difficulty of lining up a well conditioned location matrix. In a space comprised of m selectivity points, there are ${}^{m}C_{7}$ distinct location matrices. As mentioned earlier in the report, m is of the order of several thousands to several hundreds of thousands. Given this herculian size of the location matrix search space, exhaustive search can not be used. Therefore, we resort to *neighbourhood search* techniques.

Neigbourhood search techniques typically start with a feasible solution. Given a feasible solution, all its neighbours are considered and the best amongst them is chosen. This process is repeated till a locally optimal solution is obtained.

In our case, a set of seven uniformly distributed points on principal diagonal of the selectivity space was fed as solution seed to the search algorithm. The neigbourhood space is defined such that a solution M' is considered to be a neigbour of another solution M iff M' differs from M in exactly one of the seven selectivity points. With the above settings, we were able to find, without incurring large overheads, well-conditioned location matrices. For e.g., for 2D selectivity spaces, the following seven selectivity points corresponding to a location matrix with condition number of 6 can be used:

$$\begin{aligned} M = \{&s_1 : (0.3293, 0.9697), s_2 : (0.9744, 0.9802), \\ &s_3 : (0.1276, 0.3843), s_4 : (0.9714, 0.2455), \\ &s_5 : (0.6919, 0.0281), s_6 : (0.087, 0.00048), \\ &s_7 : (0.0226, 0.9812)\} \end{aligned} \tag{5.8}$$

An important point to note here is that size of the neigbourhood space scales exponentially with dimensionality of the search space, i.e. $O(r^d)$. Hence, exploring the entire neigbourhhod space of a candidate solution, particularly in case of higher dimensional selectivity spaces, is

impractical. Thus, for higher dimensional selectivity spaces, *very large scale neighbourhood search* (VLSN) techniques [26] can be employed to avoid explicitly searching over a large neighbourhood. We hasten to point out that these seven selectivity points are irrespective of the database or query template. Thus, finding a well conditioned location matrix is a one time cost and can be done offline.

To confirm the scalability of our approach with the complexity of cost models, we tried to construct a well conditioned location matrix for a few complex cost models as well. For instance, for the cost model defined in (5.2), we were able to find a set of 13 selectivity points which correspond to location matrix with a condition number of 5.

WC-SEER (Plan Diagram P, Threshold λ)

1: Create a Set-Cover Instance $I = (U, S)$, where $S = \{S_1, S_2, \cdots, S_n\}$, $U = \{1, 2, \cdots, n\}$, corresponding to the n plans in the original plan diagram.
2: Set each $S_i = \{i\}$, $\forall i = 1 \cdots n$
3: **for** each plan P_i in the original plan diagram **do**
4: construct $P_i's$ cost function, $Cost_{P_i}(.)$, as explained in Section 5.2.2, using the seven selectivity points prescribed in (5.8).
5: **end for**
6: **for** each pair of plans (P_i, P_j) **do**
7: $safety \leftarrow TRUE$
8: **for** each selectivity point $s \in S$ **do**
9: **if** $Cost_{P_i}(s) \leq (1 + \lambda) Cost_{P_j}(s)$ **then**
10: $saftey \leftarrow FALSE$
11: **end if**
12: **end for**
13: **if** $safety == TRUE$ **then**
14: $S_i = S_i \cup \{j\}$
15: **end if**
16: **end for**
17: Solve the set-cover instance I using the Greedy SetCover algorithm to identify the plans retained in R.

Figure 5.3: **The WC-SEER Reduction Algorithm**

5.2.4 Analysis

The total number of FPC operations required in the WC-SEER algorithm is equal to n times the number of parameters in the underlying cost model. For instance, the cost model defined in

(3.1) has $2^{(d+1)} - 1$ parameters for a $d-$dimensional selectivity space. Thus the total number of FPC operations required is $n.(2^{(d+1)} - 1)$. As we have mentioned before in the report that FPC operations are the bottleneck of robust reduction algorithms, the running time complexity of WC-SEER is $O(n.2^d)$.

Chapter 6

Experimental Results

6.1 Experimental Setup

The testbed used in our experiments is the Picasso optimizer visualization tool [20], executing on a Sun Ultra 20 workstation equipped with an Opteron Dual Core 4GHz processor, 4 GB of main memory and 720 GB of hard disk, running the Windows XP Pro operating system. The experiments were conducted over plan diagrams produced from a variety of two and three-dimensional **TPC-H** and **TPC-DS**-based query templates. The TPC-H database contains *uniformly* distributed data of size 1GB, while the TPC-DS database hosts skewed data that occupies 100GB. The plan diagrams were generated with a industrial-strength commercial database query optimizer.

6.1.1 Physical Design.

We considered two different physical design configurations in our study: **PrimaryKey (PK)** and **AllIndex (AI)**. PK represents the default physical design of our database engine, wherein a clustered index is created on each primary key. AI, on the other hand, represents an "index-rich" situation wherein (single-column) indices are available on all query-related schema attributes.

In the subsequent discussion, we use QT*x* to refer to a query template based on Query *x* of

Query Template	Orig Plans	SEER			CC-SEER			WC-SEER		
		Plans	Agg	Min	Plans	Agg	Min	Plans	Agg	Min
	#	#	SERF	SERF	#	SERF	SERF	#	SERF	SERF
QT2	60	7	0.22	-0.1	10	0.22	-0.05	6	0.25	-0.1
QT5	51	3	0.48	0	4	0.45	-0.1	2	0.50	0.2
QT8	121	2	0.88	0.1	3	0.88	0.02	2	0.88	0.1
QT9	137	4	0.59	-0.1	5	0.52	-0.3	4	0.62	-0.3
QT10	44	3	0.18	0.01	6	0.2	0	3	0.23	0.01
QT16	32	5	0.21	-0.1	6	0.18	-0.1	3	0.28	0
3DQT5	68	3	0.24	-0.02	3	0.22	-0.01	3	0.27	0
3DQT8	191	4	0.54	0	4	0.50	-0.04	4	0.56	-0.01
3DQT10	75	4	0.32	-0.03	11	0.31	0.01	4	0.43	0.01
AIQT2	87	15	0.71	-0.02	19	0.68	-0.04	10	0.72	-0.05
AIQT5	126	13	0.40	0	15	0.41	0.02	11	0.51	0.01
AIQT8	121	6	0.38	0.02	7	0.38	0.02	7	0.46	0.02
AIQT9	132	12	0.48	-0.01	13	0.42	0	12	0.52	0
AIQT10	37	6	0.11	0	7	0.10	0	6	0.21	0.07
AIQT16	35	8	0.56	-0.03	9	0.55	-0.03	8	0.64	-0.1
3DAIQT5	139	10	0.48	0	15	0.48	0.01	10	0.52	0.01
3DAIQT8	168	9	0.28	-0.03	13	0.30	-0.03	6	0.35	-0.01
3DAIQT10	77	12	0.22	-0.03	14	0.20	0.05	8	0.30	0.05
DSQT12	25	2	0.32	0.01	4	0.32	0	2	0.42	0.07
DSQT18	114	2	0.68	-0.06	3	0.59	-0.06	2	0.73	0
DSQT19	55	4	0.51	0	5	0.48	0	2	0.57	0
3D-DSQT12	33	2	0.36	-0.1	4	0.35	-0.2	4	0.44	-0.1
3D-DSQT183D	222	8	0.68	0	10	0.70	0.1	4	0.70	0.1
3D-DSQT193D	98	7	0.64	-0.01	10	0.60	0	3	0.68	-0.01

Table 6.1: Plan Stability Performance

the TPC-H benchmark, and DSQT*x* to refer to a query template based on Query *x* of the TPC-DS benchmark, operating in the default PK configuration. We prefix AI the query template identifiers in describing our results for the AllIndex specialized configuration.

6.1.2 Query Location Distribution.

All the performance results shown initially in this section are for plan diagrams generated with *exponentially* distributed locations for the query points across the selectivity space, resulting in higher query densities near the selectivity axes and towards the origin. This choice is based on earlier observations in the literature (e.g. [12, 13, 21]) that plans tend to be densely packed in precisely these regions of the selectivity space. From a performance perspective, these diagrams represent the "tough-nut" challenging situations with respect to obtaining anorexic reduction due to their high plan densities and substantially broader range of plan cost values.

6.1.3 Error Resistance Metrics

Our quantification of the stability delivered through plan replacement is based on the **SERF** error resistance metric introduced in [1]. For a specific error instance, with estimated query location q_e and cost-optimal plan P_{oe}, and a run-time location q_a, the *Selectivity Error Resistance Factor* (SERF) of a replacement P_{re} w.r.t. P_{oe} is computed as

$$SERF(q_e, q_a) = 1 - \frac{c(P_{re}, q_a) - c(P_{oa}, q_a)}{c(P_{oe}, q_a) - c(P_{oa}, q_a)} \tag{6.1}$$

Intuitively, SERF captures the *fraction of the performance gap* between P_{oe} and P_{oa} at q_a that is closed by P_{re}. In principle, SERF values can range over $(-\infty, 1]$, with the following interpretations: SERF in the range $(0, 1]$, indicates that the replacement is beneficial, with values close to 1 implying immunity to the selectivity error. For SERF in the range $[-\lambda, 0]$, the replacement is indifferent in that it neither helps nor hurts, while SERF values noticeably below λ highlight a harmful replacement that materially worsens the performance.

To capture the *aggregate* impact of plan replacements on improving the resistance to selectivity errors in the entire space S, we compute **AggSERF** as:[1]

$$AggSERF = \frac{\sum_{q_e \in rep(\mathsf{S})} \sum_{q_a \in exo_{oe}(\mathsf{S})} SERF(q_e, q_a)}{\sum_{q_e \in \mathsf{S}} \sum_{q_a \in exo_{oe}(\mathsf{S})} 1} \tag{6.2}$$

where $rep(\mathsf{S})$ is the set of query instances in S whose plans were replaced, and the normalization is with respect to the number of error locations that could benefit from improved robustness. Specifically, from the universe of all possible (q_e, q_a) combinations, we exclude those scenarios which inherently do not require help – that is, when the error location falls in $safe_{oe}$, i.e. the region comprising those selectivity points at which P_{oe} is either optimal or within $(1 + \lambda)$ of the optimal plan.

Note that in the above formulation, we assume for simplicity that the actual location q_a is equally likely to be anywhere in P_{oe}'s exo-optimal space, that is, that the errors are randomly distributed over this space. In our future work, we plan to investigate the more generic case

[1] In [1], the aggregate impact was evaluated based on the locations where replacements were made, whereas our current formulation is based on the locations where robustness is desired.

where the error locations have an associated probability distribution.

Apart from AggSERF, we also compute metrics **MinSERF** and **MaxSERF**, representing the minimum and maximum values of SERF over all replacement instances. MaxSERF values close to the upper bound of 1 indicate that some replacements provided immunity to specific instances of selectivity errors. On the other hand, large negative values for MinSERF indicate that some replacements were harmful. We measure the proportion of such harmful instances in our experiments.

An important point to note here is that while it is not possible to provide meaningful assistance in $safe_{oe}$, we still need to consider the possibility that replacements may end up causing *harm*, reflected through negative SERF values, in these regions. This is taken into account in our calculation of MinSERF by evaluating it over the *entire* selectivity space.

Query Template	SEER	CC-SEER	WC-SEER
QT2	5.43	0.8	0.35
QT5	2.99	0.67	0.31
QT8	12.41	1.88	0.86
QT9	10.59	2.14	0.92
QT10	4.28	0.52	0.24
QT16	2.43	0.45	0.26
3DQT5	20.12	3.08	1.44
3DQT8	84.16	8.17	3.5
3DQT10	19.29	3.94	2.0
AIQT2	6.83	1.2	0.52
AIQT5	5.72	1.7	0.73
AIQT8	6.6	1.83	0.81
AIQT9	6.25	2.08	1.0
AIQT10	2.65	0.48	0.2
AIQT16	2.78	0.49	0.21
3DAIQT5	40.97	6.52	2.8
3DAIQT8	50.91	8.96	3.96
3DAIQT10	17.07	4.01	1.72

Table 6.2: Running Time (Sec)

6.1.4 Performance Metrics

A variety of performance metrics are used to characterize the behavior of the various replacement algorithms:

1. Plan Stability The overall effect of plan replacements on stability is measured through the AggSERF, MaxSERF and MinSERF statistics.

2. Plan Diagram Cardinality This metric tallies the number of unique plans present in the plan diagram, with cardinalities that are less than or around *twenty* considered as *anorexic diagrams*

The various performance numbers for SEER, CC-SEER, and WC-SEER are given in Table 6.1. The computational overheads incurred by these algorithms in terms of running time are given in Table 6.2. As is evident, CC-SEER is an order of magnitude faster than SEER, while giving comparable performance in terms of reduction quality and robustness. Also is evident that WC-SEER virtuallly beats SEER and CC-SEER on all the three grounds of running time, reduction quality and robustness.

Chapter 7

Related Work

Over the last decade, a variety of *compile-time* strategies have been proposed to identify robust plans. For example, in the Least Expected Cost (LEC) approach [6, 7], it is assumed that the distribution of predicate selectivities is apriori available, and then the plan that has the least-expected-cost over the distribution is chosen for execution. While the performance of this approach is likely to be good on average, it could be arbitrarily poor for a specific query as compared to the optimizer's optimal choice for that query. Moreover, it may not always be feasible to provide the selectivity distributions.

An alternative Robust Cardinality Estimation (RCE) strategy proposed in [3] is to model the selectivity dependency of the cost functions of the various competing plan choices. Then, given a user-specified "confidence threshold" T, the plan that is expected to have the *least upper bound* with regard to cost in T percentile of the queries is selected as the preferred choice. The choice of T determines the level of risk that the user is willing to sustain with regard to worst-case behavior. Like the LEC approach, this too may be arbitrarily poor for a specific query as compared to the optimizer's optimal choice.

Finally, in the (initial) optimization phase of the Rio approach [4, 5], a set of uncertainty modeling rules from [15] are used to classify selectivity errors into one of six categories (ranging from "no uncertainty" to "very high uncertainty") based on their derivation mechanisms. Then, these error categories are converted to hyper-rectangular error boxes drawn around the optimizer's point estimate. Finally, if the plans chosen by the optimizer at the corners of

the principal diagonal of the box are the same as that chosen at the point estimate, then this plan is *assumed* to be robust throughout the box. However, the conditions under which this assumption is likely to be valid are not outlined.

Chapter 8

Conclusions

Errors in selectivity estimates are well-documented causes of poor plan choices by database optimizers. In this report, we investigated whether the optimizer's choices could be replaced by alternative plans, more resilient to these errors, from the parametric optimal set over the selectivity space. In particular, we proposed CC-SEER, an order of magnitude faster algorithm than the SEER algorithm proposed in [1], while being competitive on reduction quality and robustness. We also investigated the scalability of these techniques with the cost model complexity, thereby showing that how they will not work with complex cost models. To address this issue, we proposed WC-SEER, a cost model scalable algorithm. Experimental evidence suggests that the WC-SEER algorithm beats the previous techniques, i.e. SEER/CC-SEER, on all grounds.

Bibliography

[1] Harish D., P. Darera and J. Haritsa, "Identifying Robust Plans through Plan Diagram Reduction", *Proc. of 34th Intl. Conf. on Very Large Data Bases (VLDB)*, August 2008

[2] A. Aboulnaga and S. Chaudhuri, "Self-tuning Histograms: Building Histograms without Looking at Data", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 1999.

[3] B. Babcock and S. Chaudhuri, "Towards a Robust Query Optimizer: A Principled and Practical Approach", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 2005.

[4] S. Babu, P. Bizarro and D. DeWitt, "Proactive Re-Optimization", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 2005.

[5] S. Babu, P. Bizarro and D. DeWitt, "Proactive Re-Optimization with Rio", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 2005.

[6] F. Chu, J. Halpern and P. Seshadri, "Least Expected Cost Query Optimization: An Exercise in Utility", *Proc. of ACM Symp. on Principles of Database Systems (PODS)*, May 1999.

[7] F. Chu, J. Halpern and J. Gehrke, "Least Expected Cost Query Optimization: What Can We Expect", *Proc. of ACM Symp. on Principles of Database Systems (PODS)*, May 2002.

[8] A. Deshpande, Z. Ives and V. Raman "Adaptive Query Processing", *Foundations and Trends in Databases*, 2007.

[9] U. Feige, "A threshold of ln n for approximating set cover", *Journal of ACM*, 45(4), 1998.

[10] Harish D., P. Darera and J. Haritsa, "On the Production of Anorexic Plan Diagrams", *Proc. of 33rd Intl. Conf. on Very Large Data Bases (VLDB)*, September 2007.

[11] Harish D., P. Darera and J. Haritsa, "Robust Plans through Plan Diagram Reduction", *Tech. Rep. TR-2007-02, DSL/SERC, Indian Inst. of Science*, 2007. *http://dsl.serc.iisc.ernet.in/publications/report/TR/TR-2007-02.pdf*

[12] A. Hulgeri and S. Sudarshan, "Parametric Query Optimization for Linear and Piecewise Linear Cost Functions", *Proc. of 28th Intl. Conf. on Very Large Data Bases (VLDB)*, August 2002.

[13] A. Hulgeri and S. Sudarshan, "AniPQO: Almost Non-intrusive Parametric Query Optimization for Nonlinear Cost Functions", *Proc. of 29th Intl. Conf. on Very Large Data Bases (VLDB)*, September 2003.

[14] Y. Ioannidis and S. Christodoulakis, "On the Propagation of Errors in the Size of Join Results", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 1991.

[15] N. Kabra and D. DeWitt, "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 1998.

[16] E. Kreyszig, *Advanced Engineering Mathematics*, New Age International, 5th ed, 1997.

[17] L. Mackert and G. Lohman, "R* Optimizer Validation and Performance Evaluation for Local Queries", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 1986.

[18] V. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh and M. Cilimdzic, "Robust Query Processing through Progressive Optimization", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 2004.

[19] J. Patel, M. Carey and M. Vernon, "Accurate Modeling of the Hybrid Hash Join Algorithm", *Proc. of ACM SIGMETRICS Intl. Conf. on Measurement and Modeling of Computer Systems*, May 1994.

[20] Picasso Database Query Optimizer Visualizer, *http://dsl.serc.iisc.ernet.in/projects/PICASSO/picasso.html*

[21] N. Reddy and J. Haritsa, "Analyzing Plan Diagrams of Database Query Optimizers", *Proc. of 31st Intl. Conf. on Very Large Data Bases (VLDB)*, August 2005.

[22] P. Slavik, "A tight analysis of the greedy algorithm for set cover", *Proc. of 28th ACM Symp. on Theory of Computing*, 1996.

[23] M. Stillger, G. Lohman, V. Markl and M. Kandil, "LEO – DB2's LEarning Optimizer", *Proc. of 27th Intl. Conf. on Very Large Data Bases (VLDB)*, September 2001.

[24] G. Strang. *Linear Algebra and its Applications.* Thomson Learning Inc., 1988.

[25] R. Agrawal and R. Srikant. Privacy-preserving data mining. *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 2000.

[26] R. Ahuja, O. Ergun, J. Orlin, and A. Punnen. A survey of very large-scale neighborhood search techiques. *Discrete Applied Mathematics*, 123:75-102, 2002.

[27] MATLAB, *http://www.mathworks.com*

[28] *http://www.tpc.org/tpch*

[29] *http://www.tpc.org/tpcds*

[30] *http://publib.boulder.ibm.com/infocenter/db2luw/ v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/t00245*

[31] *http://msdn2.microsoft.com/en-us/library/ms189298.aspx*

[32] *http://infocenter.sybase.com/help/index.jsp? topic=/com.sybase.dc34982_1500/html/mig_gde/BABIFCAF.htm*