Analyzing Plan Diagrams of Database Query Optimizers

Naveen Reddy   Jayant R. Haritsa[*]

Database Systems Lab, SERC/CSA
Indian Institute of Science, Bangalore 560012, INDIA

Abstract

A "plan diagram" is a pictorial enumeration of the execution plan choices of a database query optimizer over the relational selectivity space. In this paper, we present and analyze representative plan diagrams on a suite of popular commercial query optimizers for queries based on the TPC-H benchmark. These diagrams, which often appear similar to cubist paintings, provide a variety of interesting insights, including that current optimizers make extremely fine-grained plan choices, which may often be supplanted by less efficient options without substantively affecting the quality; that the plan optimality regions may have highly intricate patterns and irregular boundaries, indicating strongly non-linear cost models; that non-monotonic cost behavior exists where increasing result cardinalities decrease the estimated cost; and, that the basic assumptions underlying the research literature on parametric query optimization often do not hold in practice.

1 Introduction

Modern database systems use a *query optimizer* to identify the most efficient strategy to execute the SQL queries that are submitted by users. The efficiency of the strategies, called "plans", is usually measured in terms of query response time. Optimization is a mandatory exercise since the difference between the cost of the best execution plan and a random choice could be in orders of magnitude. The role of query optimizers has become especially critical in recent times due to the high degree of query complexity characterizing current data warehousing and mining applications, as exemplified by the TPC-H decision support benchmark [20].

[*]Contact Author: haritsa@dsl.serc.iisc.ernet.in

Proceedings of the 31st VLDB Conference,
Trondheim, Norway, 2005

While commercial query optimizers each have their own "secret sauce" to identify the best plan, the de-facto standard underlying strategy, based on the classical System R optimizer [16], is the following: Given a user query, apply a variety of heuristics to restrict the combinatorially large search space of plan alternatives to a manageable size; estimate, with a cost model and a dynamic-programming-based processing algorithm, the efficiency of each of these candidate plans; finally, choose the plan with the lowest estimated cost.

Plan and Cost Diagrams

For a query on a given database and system configuration, the optimal plan choice is primarily a function of the *selectivities* of the base relations participating in the query – that is, the estimated number of rows of each relation relevant to producing the final result. In this paper, we use the term "plan diagram" to denote a color-coded pictorial enumeration of the execution plan choices of a database query optimizer over the relational selectivity space. An example 2-D plan diagram is shown in Figure 1(a), for a query based on Query 7 of the TPC-H benchmark, with selectivity variations on the ORDERS and CUSTOMER relations[1].

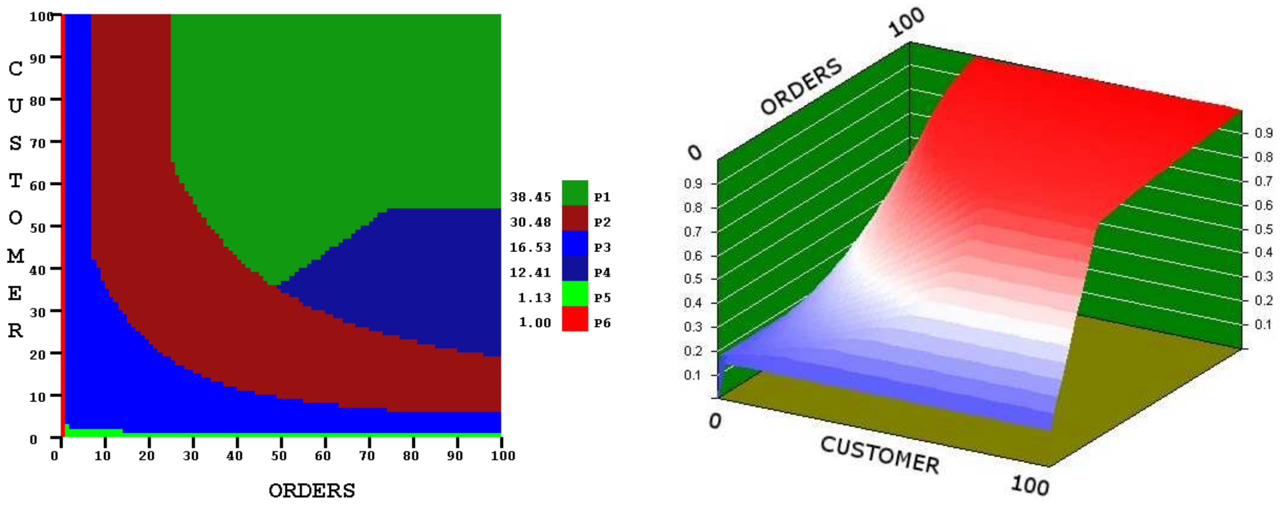
[*Note to Readers: We recommend viewing all diagrams presented in this paper directly from the color PDF file, available at [25], or from a color print copy, since the greyscale version may not clearly register the various features.*]

In this picture, produced with a popular commercial query optimizer, a set of six optimal[2] plans, P1 through P6, cover the selectivity space. The value associated with each plan in the legend indicates the percentage space coverage of that plan – P1, for example, covers about 38% of the area, whereas P6 is chosen in only 1% of the region.

Complementary to the plan diagram is a "cost diagram", shown in Figure 1(b), which is a three-dimensional visualization of the estimated plan execution costs over the same relational selectivity space (in this picture, the costs are normalized to the maximum cost over the space, and the colors reflect the relative magnitudes with blue indicating low cost, white – medium cost, and red – high cost).

[1]Specifically, the variation is on the `o_totalprice` and `c_acctbal` attributes of these relations.

[2]The optimality is with respect to the optimizer's restricted search space, and not in a global sense.



(a) **Plan Diagram** (b) **Cost Diagram**

Figure 1: **Smooth Plan and Cost Diagram (Query 7)**

The Picasso Tool

As part of our ongoing project on developing value-addition software for query optimization [24], we have created a tool, called **Picasso**, that given a query and a relational engine, automatically generates the associated plan and cost diagrams. In this paper, we report on the features of the plan/cost diagrams output by Picasso on a suite of three popular commercial query optimizers for queries based on the TPC-H benchmark. [Due to legal restrictions, these optimizers are anonymously identified as OptA, OptB and OptC, in the sequel.]

Our evaluation shows that a few queries in the benchmark do produce "well-behaved" or "smooth" plan diagrams, like that shown in Figure 1(a). A substantial remainder, however, result in extremely complex and intricate plan diagrams that appear similar to *cubist paintings*[3], providing rich material for investigation. A particularly compelling example is shown in Figure 2(a) for Query 8 of the benchmark with optimizer OptA[4], where no less than 68 plans cover the space in a highly convoluted manner! Further, even this cardinality is a *conservative* estimate since it was obtained with a query grid of 100 x 100 – a finer grid size of 300 x 300 resulted in the plan cardinality going up to 80 plans!

Before we go on, we hasten to clarify that our goal in this paper is to provide a broad overview of the intriguing behavior of modern optimizers, but *not* to make judgements on specific optimizers, nor to draw conclusions about the relative qualities of their execution plans. Further, not being privy to optimizer internals, some of the conclusions drawn here are perforce speculative in nature and should therefore be treated as such. Our intention is primarily to alert database system designers and developers to the phenomena that we have encountered during the course of our study, with the hope that they may prove useful in building the next generation of optimizers.

[3]Hence, the name of our tool – Pablo Picasso is considered to be a founder of the cubist painting genre [23].

[4]Operating at its highest optimization level.

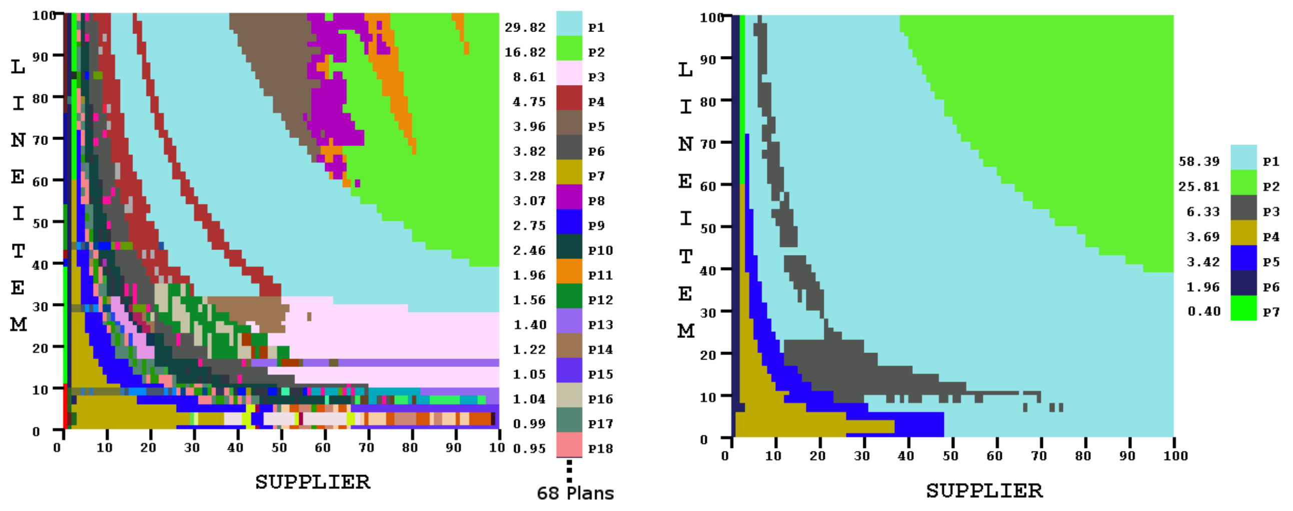
Features of Plan and Cost Diagrams

Analyzing the TPC-H-based query plan and cost diagrams provides a variety of interesting insights, including the following:

Fine-grained Choices: Modern query optimizers often make extremely *fine-grained* plan choices, exhibiting a marked skew in the space coverage of the individual plans. For example, 80 percent of the space is usually covered by less than 20 percent of the plans, with many of the smaller plans occupying less than *one percent* of the selectivity space. Using the well-known Gini index [22], which ranges over [0,1], to quantify the skew, we find that all the optimizers, *across the board*, exhibit a marked skew in excess of 0.5 for most queries, on occasion going even higher than 0.8.

Further, and more importantly, we show that the small-sized plans may often be supplanted by larger siblings *without substantively affecting the quality*. For example, the plan diagram of Figure 2(a) which has 68 plans can be "reduced" to that shown in Figure 2(b) featuring as few as *seven* plans, without increasing the estimated cost of any individual query point by more than 10 percent.

Overall, this leads us to the hypothesis that current optimizers may perhaps be over-sophisticated in that



(a) **Complex Plan Diagram** (b) **Reduced Plan Diagram**

Figure 2: **Complex Plan and Reduced Plan Diagram (Query 8, OptA)**

they are "doing too good a job", not merited by the coarseness of the underlying cost space. Moreover, if it were possible to simplify the optimizer to produce only reduced plan diagrams, it is plausible that the considerable processing overheads typically associated with query optimization could be significantly lowered.

Complex Patterns: The plan diagrams exhibit a variety of intricate tessellated patterns, including *speckles*, *stripes*, *blinds*, *mosaics* and *bands*, among others. For example, witness the rapidly alternating choices between plans P12 (dark green) and P16 (light gray) in the bottom left quadrant of Figure 2(a). Further, the boundaries of the plan optimality regions can be highly irregular – a case in point is plan P8 (dark pink) in the top right quadrant of Figure 2(a). These complex patterns appear to indicate the presence of strongly non-linear and discretized cost models, again perhaps an over-kill in light of Figure 2(b).

Non-Monotonic Cost Behavior: We have found quite a few instances where, although the base relation selectivities and the result cardinalities are monotonically increasing, the cost diagram does *not* show a corresponding monotonic behavior.[5] Sometimes, the non-monotonic behavior arises due to a change in plan, perhaps understandable given the restricted search space evaluated by the optimizer. But, more surprisingly, we have also encountered situations where a plan shows such behavior even *internal* to its optimality region.

[5]Our query setup is such that in addition to the result cardinality monotonically increasing as we travel outwards along the selectivity axes, the result tuples are also *supersets* of the previous results.

Validity of PQO: A rich body of literature exists on *parametric query optimization* (PQO) [1, 2, 7, 8, 3, 4, 10, 11, 12]. The goal here is to apriori identify the optimal set of plans for the entire relational selectivity space at compile time, and subsequently to use at run time the actual selectivity parameter settings to identify the best plan – the expectation is that this would be much faster than optimizing the query from scratch. Much of this work is based on a set of assumptions, that we do not find to hold true, *even approximately*, in the plan diagrams produced by the commercial optimizers.

For example, one of the assumptions is that a plan is optimal within the *entire region* enclosed by its plan boundaries. But, in Figure 2(a), this is violated by the small (brown) rectangle of plan P14, close to coordinates (60,30), in the (light-pink) optimality region of plan P3, and there are several other such instances.

On the positive side, however, we show that some of the important PQO assumptions do hold approximately for *reduced* plan diagrams.

1.1 Organization

The above effects are described in more detail in the remainder of this paper, which is organized as follows: In Section 2, we present the Picasso tool and the testbed environment. Then, in Section 3, the skew in the plan space distribution, as well as techniques for reducing the plan set cardinalities, are discussed. The relationship to PQO is explored in Section 4. Interesting plan diagram motifs are presented in Section 5. An overview of related work is provided in Section 6. Finally, in Section 7, we summarize the conclusions of our study and outline future research avenues.

2 Testbed Environment

In this section, we overview the Picasso tool and the experimental environment under which the plan and cost diagrams presented here were produced.

2.1 Picasso Tool

The Picasso tool is part of our ongoing project on developing value-added tools for query optimization [24]. Through its GUI, users can submit a *query template* [5], the grid granularity at which instances of this template should be distributed across the plan space, the relations (axes) and their attributes on which the diagrams should be constructed, and the choice of query optimizer. A snapshot of the interface for a template based on Query 2 of the TPC-H benchmark, is shown in Figure 3 (the predicates "`p_size < C1`" and "`ps_supplycost < C2`" determine the selectivity axes).

With this information, the tool automatically generates SQL queries that are evenly spaced across the relational selectivity space (the statistics present in the database catalogs are used to compute the selectivities). For example, with a grid spacing of 100 x 100, a plan diagram is produced by firing 10000 queries, each query covering 0.01 percent of the plan diagram area. The resulting plans are stored persistently in the database, and in the post-processing phase, a unique color is assigned to each distinct plan, and the area covered by the plan is also estimated. The space is then colored according to this assignment, and the legend shows (in ranked order) the space coverage of each plan. Differences between specific plans are easily identified using a *PlanDiff* component that only requires dragging the cursor from one plan to the other in the plan diagram.

For each plan diagram, the corresponding cost diagram is obtained by feeding the query points and their associated costs to a 3-D visualizer – currently, the freeware Plot3D [21] is used for this purpose.

2.2 Database and Query Set

The database was created using the synthetic generator supplied with the TPC-H decision support benchmark, which represents a commercial manufacturing environment, featuring the following relations: REGION, NATION, SUPPLIER, CUSTOMER, PART, PARTSUPP, ORDERS and LINEITEM. A gigabyte-sized database was created on this schema, resulting in cardinalities of 5, 25, 10000, 150000, 200000, 800000, 1500000 and 6001215, for the respective relations.

All query templates were based on the TPC-H benchmark, which features a set of 22 queries, Q1 through Q22. To ensure coverage of the full range of selectivities, the relational axes in the plan diagrams are chosen only from the large-cardinality tables occurring in the query (i.e. NATION and REGION, which are very small, are not considered). An additional restriction is that the selected tables should feature only in join predicates in the query, but not in any constant predicates. For a given choice of such tables, additional one-sided range predicates on attributes with high-cardinality domains in these tables are added to the queries to support a fine-grained continuous variation of the associated relational selectivities. As a case in point, the plan diagram in Figure 2(a) on the SUPPLIER and LINEITEM relations, was produced after adding to Q8 the predicates `s_acctbal < C1` and `l_quantity < C2`, where `C1` and `C2` are constants that are appropriately set to generate the desired selectivities on these relations. In the remainder of this paper, for ease of exposition, we will use the benchmark query numbers for referring to the associated Picasso templates.

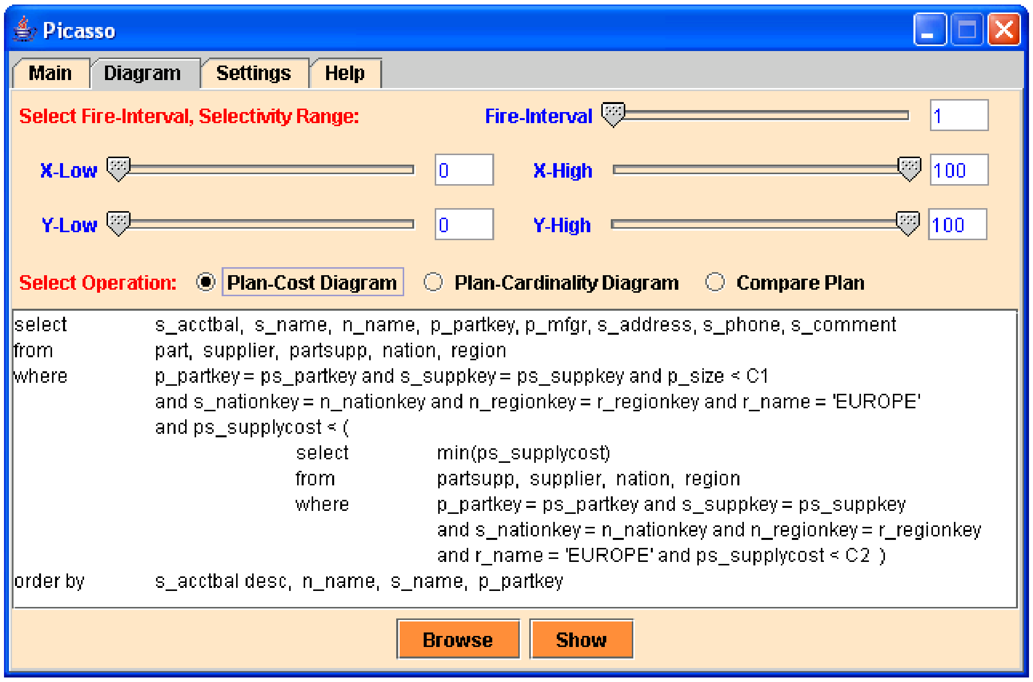
While plan and cost diagrams have been generated for most of the benchmark queries, we focus in the remainder of this paper only on those queries that have "dense" plan diagrams – that is, diagrams whose optimal plan set cardinality is 10 or more, making them interesting for analysis – for at least one of the commercial optimizers. For computational tractability, a query grid spacing of 100 x 100 is used, unless explicitly mentioned otherwise. Further, for ease of presentation and visualization, the query workloads are restricted to 2-dimensional selectivity spaces (with the exception of queries Q1 and Q6, which feature only a single relation, and therefore have a 1-D selectivity space by definition).

2.3 Relational Engines

A suite of three popular commercial relational optimizers were evaluated, but, as mentioned earlier, we are unable to provide their details due to legal restrictions. Some of the engines offer a range of optimization levels that tradeoff quality against time, or result latency versus response time. We have experimented with all these levels, but for ease of exposition, the diagrams presented here, unless explicitly mentioned otherwise, are restricted to those obtained with the default optimization levels. Also, we ensured that the full choice of candidate algorithms for each query operator was made available. To support the making of informed plan choices, commands were issued to collect statistics on all the attributes participating in the queries. Finally, for every query submitted to the database systems, commands were issued to only "explain" the plan – that is, the plan to execute the query was generated, but not executed. This is because our focus here is on plan choices, and not on evaluating the accuracy of the associated cost estimations.

2.4 Computational Platform

A vanilla platform consisting of a Pentium-IV 2.4 GHz PC with 1 GB of main memory and 120 GB of hard disk, running the Windows XP Pro operating system, was used in our experiments. For this platform, the complete set of evaluated queries and their associated plan, cost, and



Select Fire-Interval, Selectivity Range: Fire-Interval 1

X-Low 0 X-High 100

Y-Low 0 Y-High 100

Select Operation: [✓ Plan-Cost Diagram] Plan-Cardinality Diagram Compare Plan

```
select      s_acctbal, s_name, n_name, p_partkey, p_mfgr, s_address, s_phone, s_comment
from        part, supplier, partsupp, nation, region
where       p_partkey = ps_partkey and s_suppkey = ps_suppkey and p_size < C1
            and s_nationkey = n_nationkey and n_regionkey = r_regionkey and r_name = 'EUROPE'
            and ps_supplycost < (
                        select      min(ps_supplycost)
                        from        partsupp, supplier, nation, region
                        where       p_partkey = ps_partkey and s_suppkey = ps_suppkey
                                    and s_nationkey = n_nationkey and n_regionkey = r_regionkey
                                    and r_name = 'EUROPE' and ps_supplycost < C2 )
order by    s_acctbal desc, n_name, s_name, p_partkey
```

Figure 3: **Picasso GUI**

reduced-plan diagrams, over all three optimizers, are available at [25] – in the remainder of this paper, we discuss their highlights.

3 Skew in Plan Space Coverage

We start off our analysis of plan diagrams by investigating the *skew* in the space coverage of the optimal set of plans. In Table 1, we show for the various benchmark queries, three columns for each optimizer: First, the cardinality of the optimal plan set; second, the (minimum) percentage of plans required to cover 80 percent of the space; and, third, the Gini index [22], a popular measure of income inequality in economics – here we treat the space covered by each plan as its "income". Our choice of the Gini index is due to its desirable statistical properties including being Lorenz-consistent, and bounded on the closed interval [0,1], with 0 representing no skew and 1 representing extreme skew. Finally, the averages across all *dense queries* (10 or more plans in the plan diagram) are also given at the bottom of Table 1.

These statistics show that the cardinality of the optimal plan set can reach high values for a significant proportion of the queries. For example, the average (dense) cardinality is considerably in excess of *twenty*, across all three optimizers. Q9, in particular, results in more than 40 plans for all the optimizers. But it is also interesting to note that high plan density is not solely query-specific since there can be wide variations between the optimizers on individual queries – for example, Q18 results in 13 plans for OptB, but only 5 plans each for OptA and OptC. Conversely, OptB requires only 6 plans for Q7, but OptA and OptC employ 13 and 19 plans, respectively. It should also be noted that a common feature between Q8 and Q9, which both have large number of plans across all three systems, is that they are join-intensive nested queries with the outer query featuring *dynamic* base relations (i.e. the relations in the *from clause* are themselves the output of SQL queries).

When the fractional cardinality required to cover 80 percent of the space is considered, we see that on average it is in the neighborhood of 20 percent, highlighting the inequity in the plan space distribution. This is comprehensively captured by the Gini index values, which are mostly in excess of 0.5, and even reaching 0.8 on occasion, indicating very high skew in the plan space distribution. Further, note that this skew is present, *across the board, in all the optimizers*.

Overall, the statistics clearly demonstrate that modern optimizers tend to make extremely fine-grained choices. Further, these numbers are *conservative* in that they were obtained with a 100 x 100 grid – with finer-granularity grids, as mentioned in the Introduction, the number of plans often increased even further. For example, using a 1000 x 1000 grid for Q9 on OptB, the number of plans increased from 44 to 60!

3.1 Plan Cardinality Reduction by Swallowing

Motivated by the above skewed statistics, we now look into whether it is possible to replace many of the small-sized plans by larger-sized plans in the optimal plan set, without unduly increasing the cost of the query points associated with the small plans. That is, can small plans be "completely swallowed" by their larger siblings, leading to a reduced plan set cardinality, without materially affecting the associated queries.

TPC-H		OptA			OptB			OptC	
Query Number	**Plan Card**	**80% Coverage**	**Gini Index**	**Plan Card**	**80% Coverage**	**Gini Index**	**Plan Card**	**80% Coverage**	**Gini Index**
2	22	18%	0.76	14	21%	0.72	35	20%	0.77
5	21	19%	0.81	14	21%	0.74	18	17%	0.81
7	13	23%	0.73	6	50%	0.46	19	15%	0.79
8	31	16%	0.81	25	25%	0.72	38	18%	0.79
9	63	9%	0.88	44	27%	0.70	41	12%	0.83
10	24	16%	0.78	9	22%	0.69	8	25%	0.75
18	5	60%	0.33	13	38%	0.57	5	20%	0.75
21	27	22%	0.74	6	17%	0.80	22	18%	0.81
Avg(dense)	**28.7**	**17%**	**0.79**	**24.5**	**23%**	**0.72**	**28.8**	**16%**	**0.8**

Table 1: Skew in Plan Space Coverage

To do this, we first fix a threshold, λ, representing the maximum percentage cost increase that can be tolerated. Specifically, no query point in the original space should have its cost increased, *post-swallowing*, by more than λ. Next, to decide whether a plan can be swallowed, we use the following formulation:

Cost Domination Principle: Given a pair of distinct query points $q_1(x_1, y_1)$ and $q_2(x_2, y_2)$ in the two-dimensional selectivity space, we say that point q_2 dominates q_1, symbolized by $q_2 \succ q_1$, if and only if $x_2 \geq x_1$, $y_2 \geq y_1$, and result cardinality $R_{q_2} \geq R_{q_1}$ (note that result cardinality estimations are, in principle, independent of plan choices).[6] Then, if points $q_1(x_1, y_1)$ and $q_2(x_2, y_2)$, are associated with distinct plans P_1 and P_2, respectively, in the original space, C_1^2, the cost of executing query q_1 with plan P_2 is upper-bounded by C_2^2, the cost of executing q_2 with P_2, if and only if $q_2 \succ q_1$.

Intuitively, what is meant by the cost domination principle is that we expect the optimizer cost functions to be monotonically non-decreasing with increasing base relation selectivities and result cardinalities. Equivalently, a plan that processes a superset of the input, and produces a superset of the output, as compared to another plan, is estimated to be more costly to execute. However, as discussed later in Section 5, this (surprisingly) does not always prove to be the case with the current optimizers, and we therefore have to explicitly check for the applicability of the principle.

Based on the above principle, when considering swallowing possibilities for a query point q_s, we only look for replacements by "foreign" (i.e. belonging to a different plan) query points that are in the *first quadrant* relative to q_s as the origin, since these points upper-bound the cost of the plan at the origin. This is made clear in Figure 4, which shows that, independent of the cost model of the dominating plan, the cost of any foreign query point in the first quadrant will be an upper bound on the cost of executing the foreign plan at the swallowed point. We now need to find the set of dominating foreign points that are within the λ threshold, and if such points exist, choose one replacement from among these – currently, we choose the point with the lowest cost as the replacement. Finally, an *entire plan* can be swallowed if and only if *all* its query points can be swallowed by either a single plan or a group of plans. In our processing, we first order the plans in ascending order of size, and then go up the list, checking for the possibility of swallowing each plan.

[6]Result cardinalities are usually monotonically non-decreasing with increasing x and y, but this need not always be the case, especially for nested queries.

Note that the cost domination principle is conservative in that it does not capture all swallowing possibilities, due to restricting its search only to the first quadrant. But, as we will show next, substantial reductions in plan space cardinalities can be achieved even with this conservative approach.

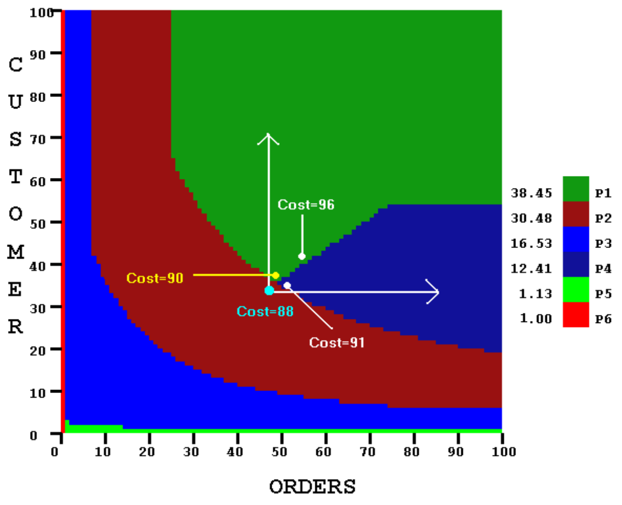


Figure 4: **Dominating Quadrant**

For the experiments presented here, we set λ, the cost increase threshold, to 10 percent. Note that in any case the cost computations made by query optimizers are themselves statistical *estimates*, and therefore allowing for a 10 percent "fudge factor" may be well within the bounds of the *inherent* error in the estimation process. In fact, as mentioned recently in [13, 18], cost estimates can often be signficantly off due to modeling errors, prompting the new wave of "learning" optimizers (e.g. LEO [18]) that iteratively refine their models to improve their estimates.

When the above plan-swallowing technique is implemented on the set of plans shown in Table 1, and with $\lambda = 10\%$, the resulting reductions (as a percentage) in the plan cardinalities are shown in Table 2. We see here that the reductions are very significant – for example, Q8 reduces by 87% (31 to 4), 84% (25 to 4) and 86% (38 to 5), for OptA, OptB and OptC, respectively. On average over dense queries, the reductions are of the order of 60% across all three optimizers, with OptC going over 70%. Also note that these reductions are *conservative* because when the grid granularity is increased – from 100 x 100 to, say, 1000 x 1000 – the new plans that emerge tend to be very small and are therefore highly likely to be subsequently swallowed. In a nutshell, the following thumb rule emerges from our results: *"two-thirds of the plans in a dense plan diagram are liable to be eliminated through plan swallowing"*.

In Table 2, we have also shown the *average* percentage increase in the costs of swallowed query points, as well as the *maximum* cost increase suffered across all query points. Note that, although the threshold is set to 10%, the *actual* average cost increase is rather low – less than 2%, which means that most of the swallowed query points *hardly* suffer on account of the replacement by an alternative plan. In fact, even the maximum increase does not always reach the threshold setting. Further, note that these averages and maxima are *upper bounds*, and the real cost estimates of the replacement plans at the swallowed points may be even lower in practice. Overall, our observation is that there appears to be significant potential to *drastically reduce the complexity of plan diagrams without materially affecting the query processing quality*.

A key implication of the above observation is the following: *Suppose it were possible to simplify current optimizers to produce only reduced plan diagrams, then the considerable computational overheads typically associated with the query optimization process may also be substantially lowered.* We suggest that this may be an interesting avenue to be explored by the database research community.

3.2 Plan Reduction $\neq$ Optimization Levels

As mentioned earlier, optimizers typically have multiple optimization levels that trade off plan quality versus optimization time, and at first glance, our plan reduction technique may appear equivalent to choosing a coarser optimization level. However, the two concepts are completely different because the optimal plan sets chosen at different levels by the optimizer may be vastly dissimilar. A striking example is Q8, where *none* of the 68 plans chosen by OptA at the highest level are present among the 8 plans chosen at the lowest level. Further, going to a coarser level of optimization does not *necessarily* result in lower plan cardinalities – a case in point is OptA on Q2, producing only 4 plans at the highest level, but as many as 22 plans at a lower level. Again, there is zero overlap between the two optimal plan sets.

In contrast, with plan reduction by swallowing, only a subset of the *original plans* chosen by the optimizer are used to cover the entire plan space. In fact, plan reduction fits in perfectly with the query clustering approach previously proposed in our Plastic plan recycling tool [5, 15, 17, 24], where queries that are expected to have identical plan templates are grouped together based on similarities in their feature vectors. This is because the cluster regions *inherently* coarsen the plan diagram granularity.

4 Relationship to PQO

A rich body of literature exists on *parametric query optimization* (PQO) [1, 2, 7, 8, 3, 4, 10, 11, 12]. The goal here is to apriori identify the optimal set of plans for the entire relational selectivity space at compile time, and subsequently to use at run time the actual selectivity parameter settings to identify the best plan – the expectation is that this would be much faster than optimizing the query from scratch. Most of this work is based on assuming cost functions that would result in one or more of the following:

Plan Convexity: If a plan P is optimal at point A and at point B, then it is optimal at all points on the line joining the two points;

Plan Uniqueness: An optimal plan P appears at only one contiguous region in the entire space;

Plan Homogeneity: An optimal plan P is optimal within the entire region enclosed by its plan boundaries.

However, we find that *none of the three* assumptions hold true, even approximately, in the plan diagrams produced by the commercial optimizers. For example, in Figure 2(a), plan convexity is severely violated by the regions covered by plans P12 (dark green) and P16 (light gray). The plan uniqueness property is violated by plan P4 (maroon) which appears in two non-contiguous locations in the top left quadrant, while plan P18 appears in finely-chopped pieces. Finally, plan homogeneity is violated by the small (brown) rectangle of plan P14, close to coordinates (60,30), in the (light-pink) optimality region of plan P3.

The prior literature [8, 12] had also estimated that *high plan densities* are to be expected only along the selectivity axes – that is, where one or both base relations in the plan diagram are extremely selective, providing only a few tuples. However, we have found that high plan densities can be present elsewhere in the selectivity space also – for example, see the region between plans P5 (dark brown) and P11 (orange) in Figure 2(a). This is also the reason for our choosing a uniform distribution of the query instances, instead of the exponential distribution towards lower selectivity values used in [8].

In the following section, more detailed statistics about the violations of the above assumptions are presented, as part of a discussion on interesting plan diagram patterns.

TPC-H Query Number	OptA Percent Card Decrease	OptA Average Cost Increase	OptA Maximum Cost Increase	OptB Percent Card Decrease	OptB Average Cost Increase	OptB Maximum Cost Increase	OptC Percent Card Decrease	OptC Average Cost Increase	OptC Maximum Cost Increase
2	59.2	1.0	4.4	64.2	0.6	5.9	77.1	3.2	6.4
5	67.3	2.6	8.1	42.9	0.1	0.6	61.1	0.2	8.1
7	46.1	0.1	9.5	16.6	0.4	0.7	54.5	1.1	9.5
8	87.6	0.4	9.4	84	0.9	9.1	86.8	1.2	8.4
9	84.4	1.6	8.6	36.4	1.4	8.9	80.5	2.1	8.3
10	67.6	0.8	4.4	44.4	0.5	6.1	62.5	0.4	2.4
18	40.0	0.1	0.5	46.2	3.7	9.6	0	0	0
21	59.8	0.0	0.2	66.7	0.9	2.5	68.2	0.7	6.9
Avg(dense)	**67.4**	**0.9**	**6.4**	**56.9**	**0.7**	**6.1**	**71.4**	**1.4**	**7.9**

Table 2: Plan Cardinality Reduction by Swallowing

5 Interesting Plan Diagram Patterns

We now move on to presenting representative instances of a variety of interesting patterns that emerged in the plan diagrams across the various queries and optimizers that we evaluated in our study.

5.1 Plan Duplicates and Plan Islands

In several plan diagrams, we noticed that a given optimal plan may have *duplicates* in that it may appear in several different disjoint locations. Further, these duplicates may also be spatially quite separated. For example, consider the plan diagram for Q10 with OptA in Figure 5. Here, we see that plan P3 (dark pink) is present twice, being present both in the center, as well as along the right boundary of the plan space. An even more extreme example is plan P6 (dark green), which is present around the 20% and 95% markers on the CUSTOMER selectivity axis.

A different kind of duplicate pattern is seen for Q5 with OptC, shown in Figure 6, where plan P7 (magenta) is present in three different locations, all within the confines of the region occupied by plan P1 (dark orange). When plans P7 and P1 are compared, we find that the former uses a nested-loops join between the small relations NATION and REGION, whereas the latter employs a sort-merge-join instead.

Apart from duplicates, we also see that there are instances of *plan islands*, where a plan region is completely enclosed by another. For example, plan P18 is a (magenta) island in the optimality region of the (dark green) plan P6 in the lower left quadrant of Figure 5. Investigating the internals of these plans, we find that plan P18 has a hash-join between CUSTOMER and NATION followed by a hash-join with a sub-tree whose root is a nested-loop join. The only difference in plan P6 is that it first hash-joins the CUSTOMER relation with the sub-tree, and then performs the hash-join with NATION.

The number of such duplicates and islands for each optimizer, over all dense queries of the benchmark, is presented in Table 3 (Original columns). We see here that all three optimizers generate a significant number of duplicates; OptA also generates a large number of islands, whereas OptB and OptC have relatively few islands.

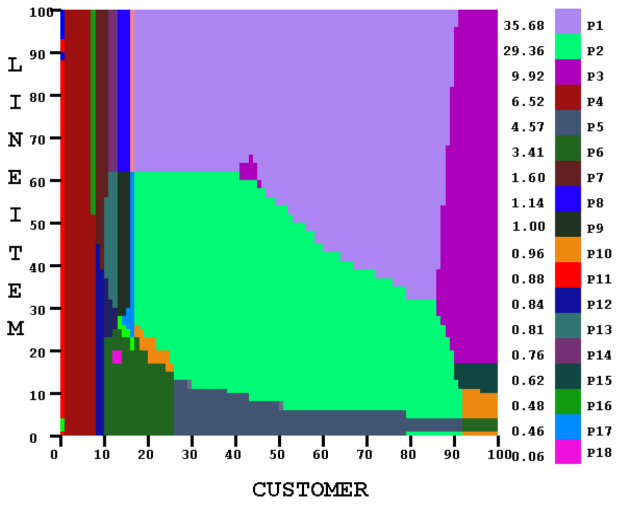


Figure 5: **Duplicates and Islands (Query 10, OptA)**

Databases	# Duplicates		# Islands	
	Original	Reduced	Original	Reduced
OptA	130	13	38	3
OptB	80	15	1	0
OptC	55	7	8	3

Table 3: Duplicates and Islands

In general, the reason for the occurrence of such duplicates and islands is that two or more competing plans have fairly close costs in that area. So, the optimizer due to its extremely fine grained plan choices, obtains plan diagrams with these features. This is confirmed from Table 3 (Reduced columns), where after application of the plan reduction algorithm, a significant decrease is observed in the number of islands and duplicates. This also means that PQO, which, as mentioned in the previous section, appears ill-suited to directly capture the complexities of modern optimizers, may turn out to be a more viable proposition in the space of reduced plan diagrams.

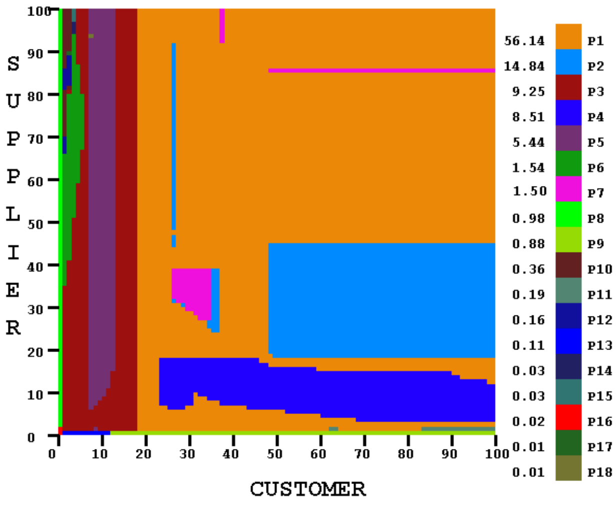


Figure 6: **Duplicates and Islands (Query 5, OptC)**

5.2 Plan Switch Points

In several plan diagrams, we find lines parallel to the axes that run through the *entire* selectivity space, with a plan shift occurring for *all* plans bordering the line, when we move across the line. We will hereafter refer to such lines as "plan switch-points".

In the plan diagram of Figure 7, obtained with Q9 on OptA, an example switch-point appears at approximately 30% selectivity of the SUPPLIER relation. Here, we found a *common change* in all plans across the switch-point – the hash-join sequence PARTSUPP ⋈ SUPPLIER ⋈ PART is altered to PARTSUPP ⋈ PART ⋈ SUPPLIER, suggesting an intersection of the cost function of the two sequences at this switch-point.

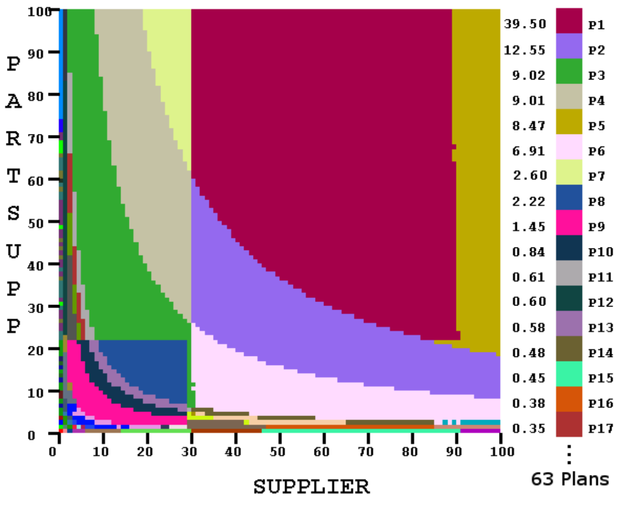


Figure 7: **Plan Switch-Point (Query 9, OptA)**

For the same Q9 query, an even more interesting switch-point example is obtained with OptB, shown in Figure 8. Here we observe, between 10% and 35% on the SUPPLIER axis, *six plans* simultaneously changing with rapid alternations to produce a "Venetian blinds" effect. Specifically, the optimizer changes from P6 to P1, P16 to P4, P25 to P23, P7 to P18, P8 to P9, and P42 to P47, from one vertical strip to the next.

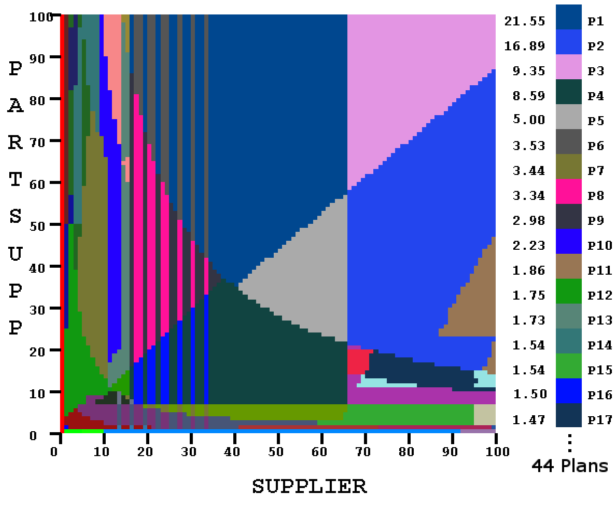


Figure 8: **Venetian Blinds Pattern (Query 9, OptB)**

The reason for this behavior is that the optimizer alternates between a *left-deep* hash join and a *right-deep* hash join across the NATION, SUPPLIER and LINEITEM relations. Both variations have almost equal estimated cost, and their cost-models are perhaps discretized in a step-function manner, resulting in the observed blinds.

5.3 Footprint Pattern

A curious pattern, similar to footprints on the beach, shows up in Figure 9, obtained with Q7 on the OptA optimizer, where we see plan P7 exhibiting a thin (cadet-blue) broken curved pattern in the middle of plan P2's (orange) region. The reason for this behavior is that both plans are of roughly equal cost, with the difference being that in plan P2, the SUPPLIER relation participates in a sort-merge-join at the top of the plan tree, whereas in P7, the hash-join operator is used instead at the same location. This is confirmed in the corresponding reduced plan diagram where the footprints disappear.

5.4 Speckle Pattern

Operating Picasso with Q17 on OptA results in Figure 10. We see here that the entire plan diagram is divided into just two plans, P1 and P2, occupying nearly equal areas, but that plan P1 (bright green) also appears as speckles sprinkled in P2's (red) area.

The only difference between the two plans is that an additional SORT operation is present in P1 on the PART relation. However, the cost of this sort is very low, and therefore we find intermixing of plans due to the close and perhaps discretized cost models.

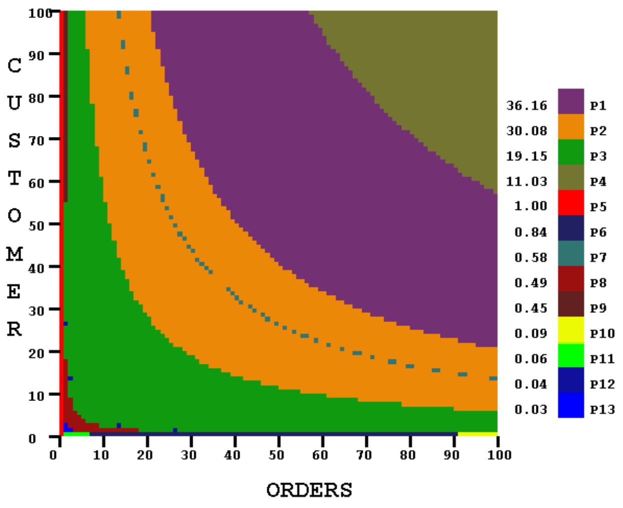


Figure 9: **Footprint Pattern (Query 7, OptA)**

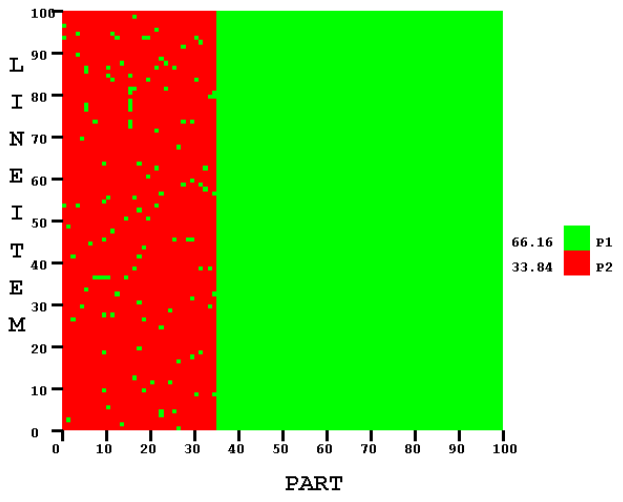


Figure 10: **Speckle Pattern (Query 17, OptA)**

5.5 Non-Monotonic Cost Behavior

The example switch-points shown earlier, were all *cost-based* switch-points, where plans were switched to derive lower execution costs. Yet another example of such a switch-point is seen in Figure 11(a), obtained with query Q2 on OptA, at 97% selectivity of the PART relation. Here, the common change in all plans across the switch-point is that the hash-join between relations PART and PARTSUPP is replaced by a sort-merge-join.

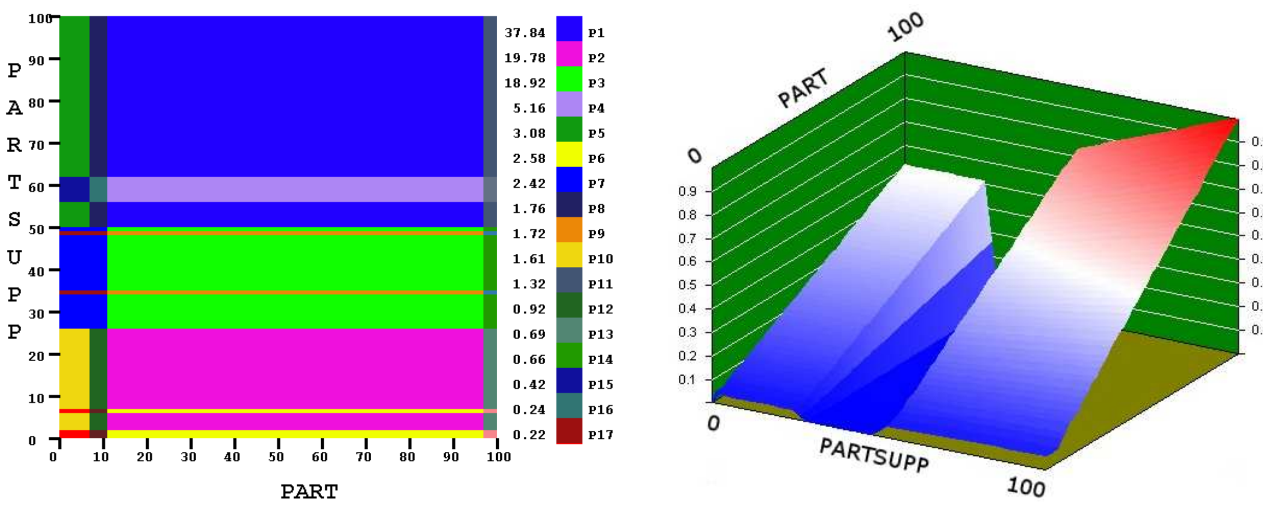
But, in the same picture, there are switch-points occurring at 26% and 50% in the PARTSUPP selectivity range, that result in a counter-intuitive *non-monotonic* cost behavior, as shown in the corresponding cost diagram of Figure 11(b). Here, we see that although the result cardinalities are monotonically increasing, the estimated costs for producing these results show a marked non-monotonic step-down behavior in the middle section. Specifically, at the 26% switch-point, an additional 'sort' operator (on `ps_supplycost`) is added, which substantially decreases the overall cost – for example, in moving from plan P2 to P3 at 50% PART selectivity, the estimated cost decreases by a factor of 50! Conversely, in moving from P3 to P1 at the 50% switch-point, the cost of the optimal plan jumps up by a factor of 70 at 50% PART selectivity.

Step-function upward jumps in the cost with increasing input cardinalities are known to occur – for example, when one of the relations in a join ceases to fit completely within the available memory – however, what is surprising in the above is the step-function cost *decrease* at the 26% switch-point. We conjecture that such disruptive cost behavior may arise either due to the presence of rules in the optimizer, or due to parameterized changes in the search space evaluated by the optimizer.

The above example showed non-monotonic behavior arising out of a plan-switch. However, more surprisingly, we have also encountered situations where a plan shows non-monotonic behavior *internal* to its optimality region. A specific example is shown in Figure 12 obtained for Q21 with OptA. Here, the plans P1, P3, P4 and P6, show a reduction in their estimated costs with increasing input and result cardinalities. An investigation of these plans showed that all of them feature a nested-loops join, whose estimated cost *decreases* with increasing cardinalities of its input relations – this may perhaps indicate an inconsistency in the associated cost model. Further, such instances of non-monotonic behavior were observed with all three optimizers.

6 Related Work

To the best of our knowledge, there has been no prior work on the analysis of plan diagrams with regard to *real-world industrial-strength* query optimizers. However, similar issues have been studied in the parametric query optimization (PQO) literature in the context of simplified self-crafted optimizers. Specifically, in [1, 11, 12], an optimizer modeled along the lines of the original System R optimizer [16] is used, with the search space restricted to left-deep join trees, and the workload comprised of pure SPJ queries with "star" or "linear" join-graphs. The metrics considered include the cardinality and spatial distribution of the set of optimal plans – while [1] considered only single-relation selectivities, [11, 12] evaluated two-dimensional relational selectivity spaces, similar to those considered in this paper. Their results in the 2-D case indicate that for linear queries, the average number of optimal plans is linear in the number of join relations, while for star queries, this number is almost quadratic. Also, the optimal plans are found to be densely packed close to the origin and the selectivity axes. An analysis of plan reduction possibilities in [1], given a plan optimality tolerance threshold, indicates that a larger fraction of plans can be removed with increasing query complexity. In [7, 8], an optimizer modeled along the lines of the Volcano query optimizer [6] is used, and they find the cardinality of the optimal plan set for queries with two, three and four-dimensional relational selectivities. They also present



(a) **Plan Diagram** (b) **Cost Diagram**

Figure 11: **Plan-Switch Non-Monotonic Costs (Query 2, OptA)**

efficient techniques for approximating the optimal plan set. Finally, a complexity analysis of the optimal plan set cardinality is made in [3] for the specific case of linear (affine) cost functions in two parameters.

While the above efforts do provide important insights, the results presented in this paper indicate that plan diagrams with sophisticated real-world optimizers and queries show much more variability with regard to both plan set cardinalities and spatial distributions, as compared to those anticipated from the PQO literature. For example, as mentioned earlier, we find that plan densities can be high even in regions far from the plan diagram axes, and that the optimality region geometries can have extremely irregular boundaries.

There has also been work on characterizing the sensitivity of query optimization to storage access cost parameters [13], but this work focuses on the robustness of optimal plan choices to inaccuracies in the optimizer input parameters, and when suboptimal choices are made, the impact of these errors. So, the focus is on plan *quality*, not quantity or spatial distribution. Further, their analysis shows that when all tables and indexes are on a single device (as in our case), the optimizer proved relatively insensitive to inaccurate resource costs in terms of plan choices – however, we find strong sensitivity with regard to *selectivity values*. Further, many of the queries for which they did find some degree of sensitivity also feature in our list of “dense” queries.

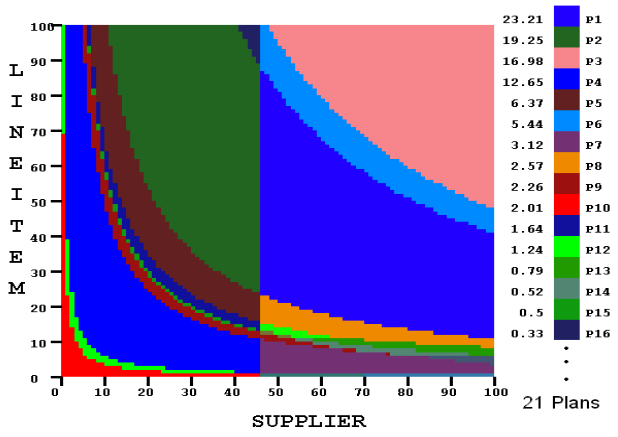
Cost-based attempts to reduce the optimizer’s search space include a “pilot-pass” approach [14], where a complete plan is initially computed and the cost of this plan is used to prune the subsequent dynamic programming enumeration by removing all subplans whose costs exceed that of the complete plan. But, it has been reported [9] that such pruning has only marginal impact in real-world environments. Finally, a preliminary study of a sampling-based approach to find acceptable quality plans, evaluated on a commercial optimizer, is discussed in [19], but its impact on the *optimal* plan set cardinality is an open issue.

7 Conclusions

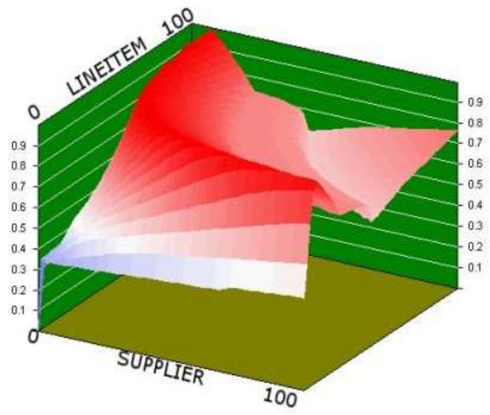
In this paper, we have attempted to analyze the behavior of (1-D and 2-D) plan and cost diagrams produced by modern optimizers on queries based on the TPC-H benchmark. Our study shows that many of the queries result in highly intricate diagrams, with several tens of plans covering the space. Further, there is heavy skew in the relative coverage of the plans, with 80 percent of the space typically covered by 20 percent or less of the plans. We showed that through a process of plan reduction where the query points associated with a small-sized plan are swallowed by a larger plan, it is possible to significantly bring down the cardinality of the plan diagram, without materially affecting the query cost.

We also demonstrated that a variety of complex and intricate patterns are produced in the diagrams, which may be an overkill given the coarseness of the underlying cost space. These patterns also indicate that the basic assumptions of parametric query optimization literature do not hold in practice. However, with reduced plan diagrams, the gap between theory and practice is considerably narrowed.

Not being privy to the internals of optimizers, our work is perforce speculative in nature. However, we hope that it may serve as a stimulus to the database research community to investigate mechanisms for pruning the plan search space so as to directly generate reduced plan diagrams, and thereby perhaps achieve substantial savings in the significant overheads normally associated with the query optimization process.



(a) **Plan Diagram**



(b) **Cost Diagram**

Figure 12: **Intra-plan Non-Monotonic Costs (Query 21, OptA)**

In the future, we would like to conduct a deeper investigation into the kinds of queries that result in dense plan diagrams, such as, for example, the presence of dynamic base relations. Also, a major limitation of our current work is its restriction to 1-D and 2-D plan diagrams – in practice, there may be many more schema and system dimensions affecting plan choices. Therefore, we intend to investigate higher dimensional plan diagrams in our future research.

Acknowledgments. This work was supported in part by a Swarnajayanti Fellowship from the Dept. of Science & Technology, Govt. of India.

References

[1] A. Betawadkar, "Query Optimization with One Parameter", *Master's Thesis, Dept. of Computer Science & Engineering, IIT Kanpur*, February 1999.

[2] R. Cole and G. Graefe, "Optimization of Dynamic Query Evaluation Plans", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 1994.

[3] S. Ganguly, "Design and Analysis of Parametric Query Optimization Algorithms", *Proc. of 24th Intl. Conf. on Very Large Data Bases (VLDB)*, August 1998.

[4] S. Ganguly and R. Krishnamurthy, "Parametric Query Optimization for Distributed Databases based on Load Conditions", *Proc. of COMAD Intl. Conf. on Management of Data*, December 1994.

[5] A. Ghosh, J. Parikh, V. Sengar and J. Haritsa, "Plan Selection based on Query Clustering", *Proc. of 28th Intl. Conf. on Very Large Data Bases (VLDB)*, August 2002.

[6] G. Graefe and W. McKenna, "The Volcano optimizer generator: Extensibility and efficient search", *Proc. of 9th IEEE Intl. Conf. on Data Engineering (ICDE)*, April 1993.

[7] 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.

[8] 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.

[9] I. Ilyas, J. Rao, G. Lohman, D. Gao and E. Lin, "Estimating Compilation Time of a Query Optimizer", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 2003.

[10] Y. Ioannidis, R. Ng, K. Shim, and T. Sellis, "Parametric Query Optimization", *Proc. of 18th Intl. Conf. on Very Large Data Bases (VLDB)*, August 1992.

[11] V. Prasad, "Parametric Query Optimization: A Geometric Approach", *Master's Thesis, Dept. of Computer Science & Engineering, IIT Kanpur*, April 1999.

[12] S. Rao, "Parametric Query Optimization: A Non-Geometric Approach", *Master's Thesis, Dept. of Computer Science & Engineering, IIT Kanpur*, March 1999.

[13] F. Reiss and T. Kanungo, "A Characterization of the Sensitivity of Query Optimization to Storage Access Cost Parameters", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 2003.

[14] A. Rosenthal, U. Dayal and D. Reiner, "Speeding a query optimizer: the pilot pass approach", *Unpublished Manuscript*, Computer Corporation of America, 1990.

[15] P. Sarda and J. Haritsa, "Green Query Optimization: Taming Query Optimization Overheads through Plan Recycling", *Proc. of 30th Intl. Conf. on Very Large Data Bases (VLDB)*, September 2004.

[16] P. Selinger, M. Astrahan, D. Chamberlin, R. Lorie and T. Price, "Access Path Selection in a Relational Database System", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 1979.

[17] V. Sengar and J. Haritsa, "PLASTIC: Reducing Query Optimization Overheads through Plan Recycling", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, June 2003.

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

[19] F. Waas and C. Galindo-Legaria, "Counting, enumerating, and sampling of execution plans in a cost-based query optimizer", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 2000.

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

[21] *http://www.orchardhouse.vtrading.co.uk/Plot3D.htm*

[22] *http://en.wikipedia.org/wiki/Gini_coefficient*

[23] *http://www.artlex.com/ArtLex/c/cubism.html*

[24] *http://dsl.serc.iisc.ernet.in/projects/PLASTIC*

[25] *http://dsl.serc.iisc.ernet.in/projects/PICASSO*