# Drawing Out the Artistic Talents of Database Query Optimizers

Database Systems Laboratory
Indian Institute of Science

Jayant Haritsa
Database Systems Lab
Indian Institute of Science
Bangalore, India

# Query Execution Plans

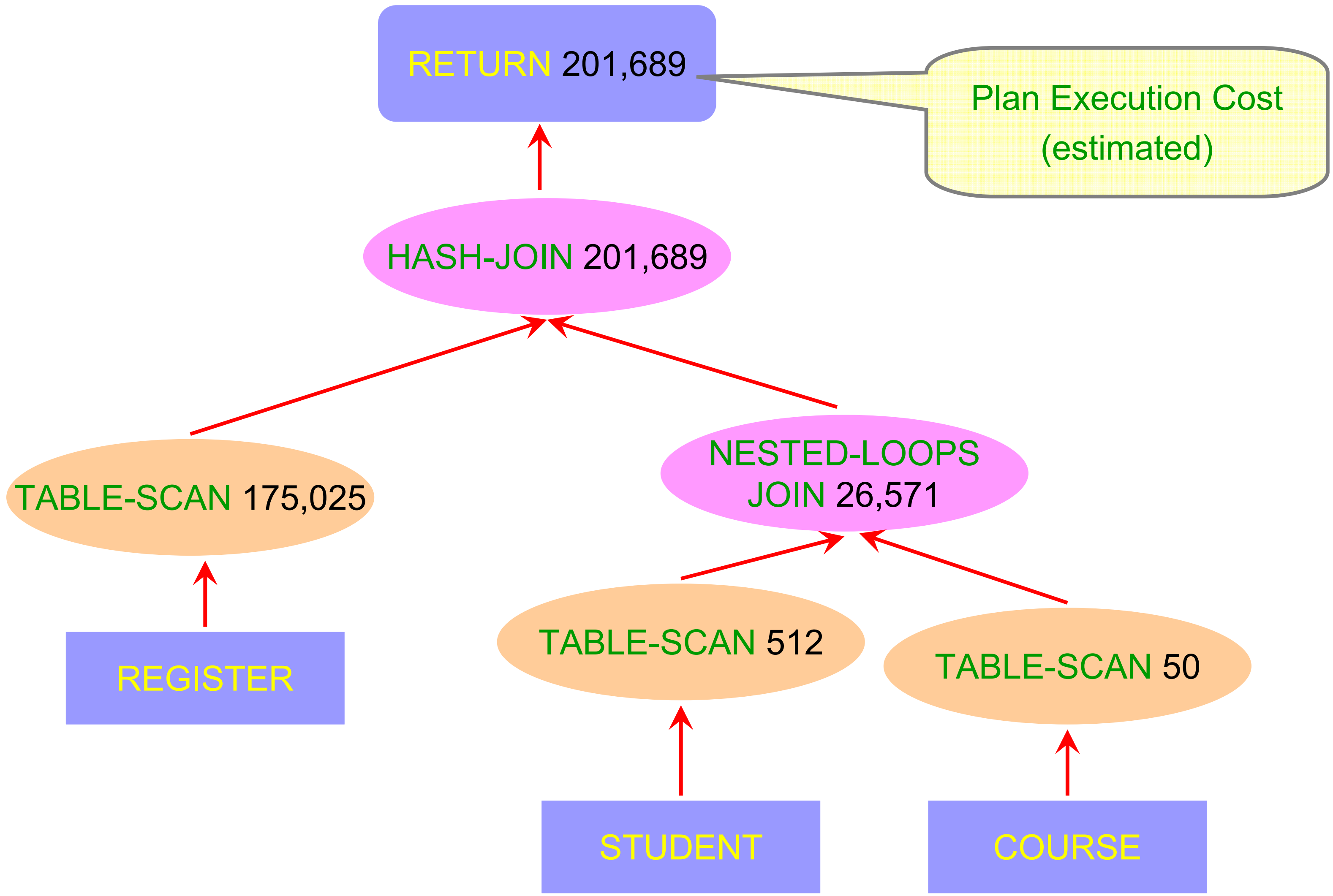
- SQL, the standard database query interface, is a declarative language
  - Specifies only what is wanted, but not how the query should be evaluated (i.e. ends, not means)
  - Example:

```
select   StudentName, CourseName
from     STUDENT,  COURSE,  REGISTER
where    STUDENT.RollNo = REGISTER.RollNo   and
         REGISTER.CourseNo = COURSE.CourseNo
```

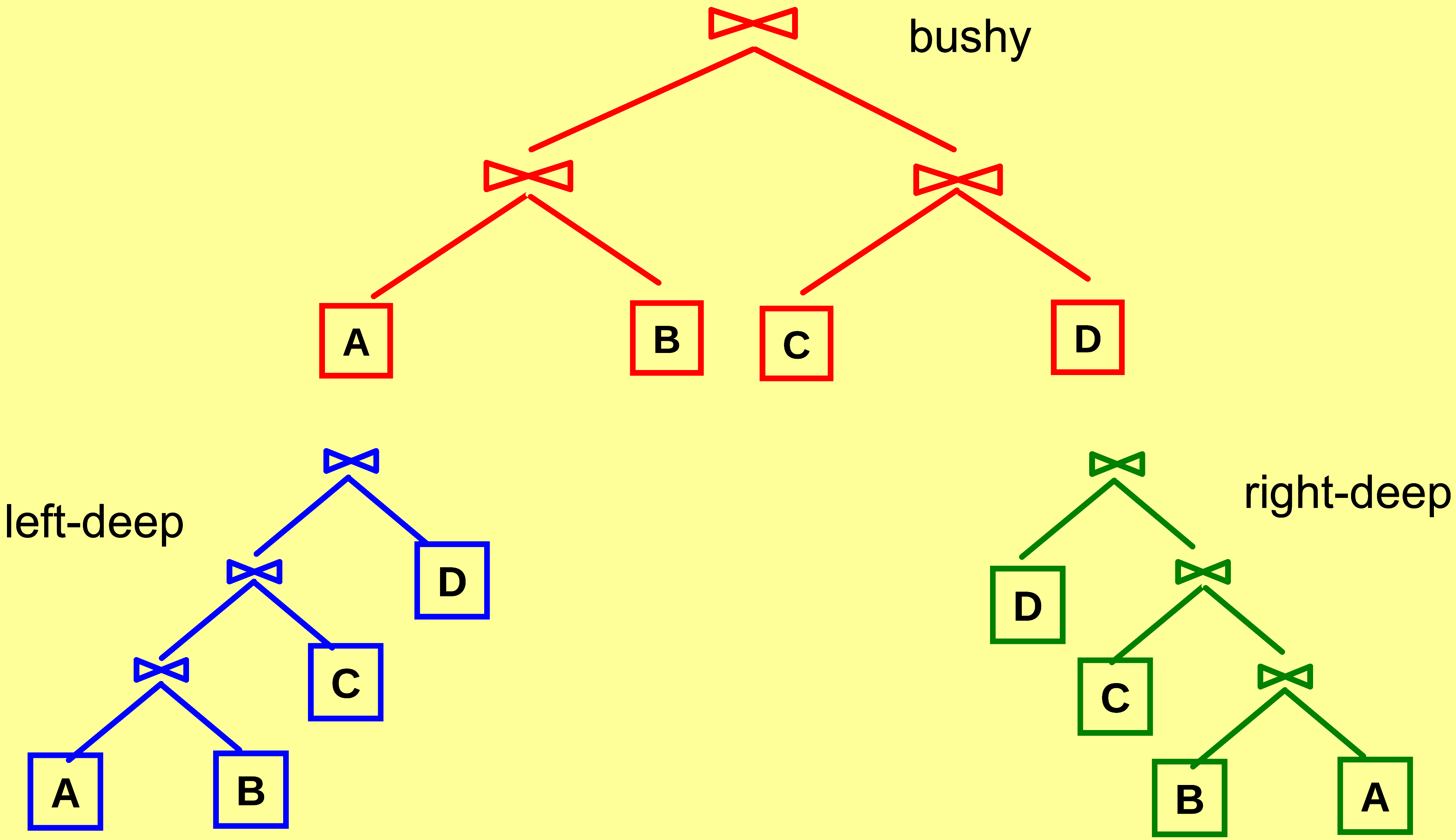
join order [ ((S join R) join C) or ((R join C) join S)  ? ]  and
join techniques  [ Nested-Loops, Sort-Merge, Hash ? ]
are left unspecified

- DBMS query optimizer identifies efficient execution strategy: "query execution plan"

# Sample Execution Plan



# Query Plan Selection



# Need for careful plan selection

- Cost difference between best plan choice and a random choice can be enormous (orders of magnitude)
- Only a small percentage of really good plans over the search space

# Relation Selectivity

- An optimizer's choice of execution plan for a query is dependent on a large number of factors. But, for a given database and system configuration, the plan choice is primarily a function of the selectivities of the base relations participating in the query
  - selectivity is the estimated percentage of rows of a relation used in producing the query result

# Plan and Cost Diagrams

- A plan diagram is a pictorial enumeration of the plan choices of a database query optimizer over the relational selectivity space
- A cost diagram is a visualization of the associated (estimated) plan execution costs over the same relational selectivity space

# Example Query [Q7 of TPC-H]

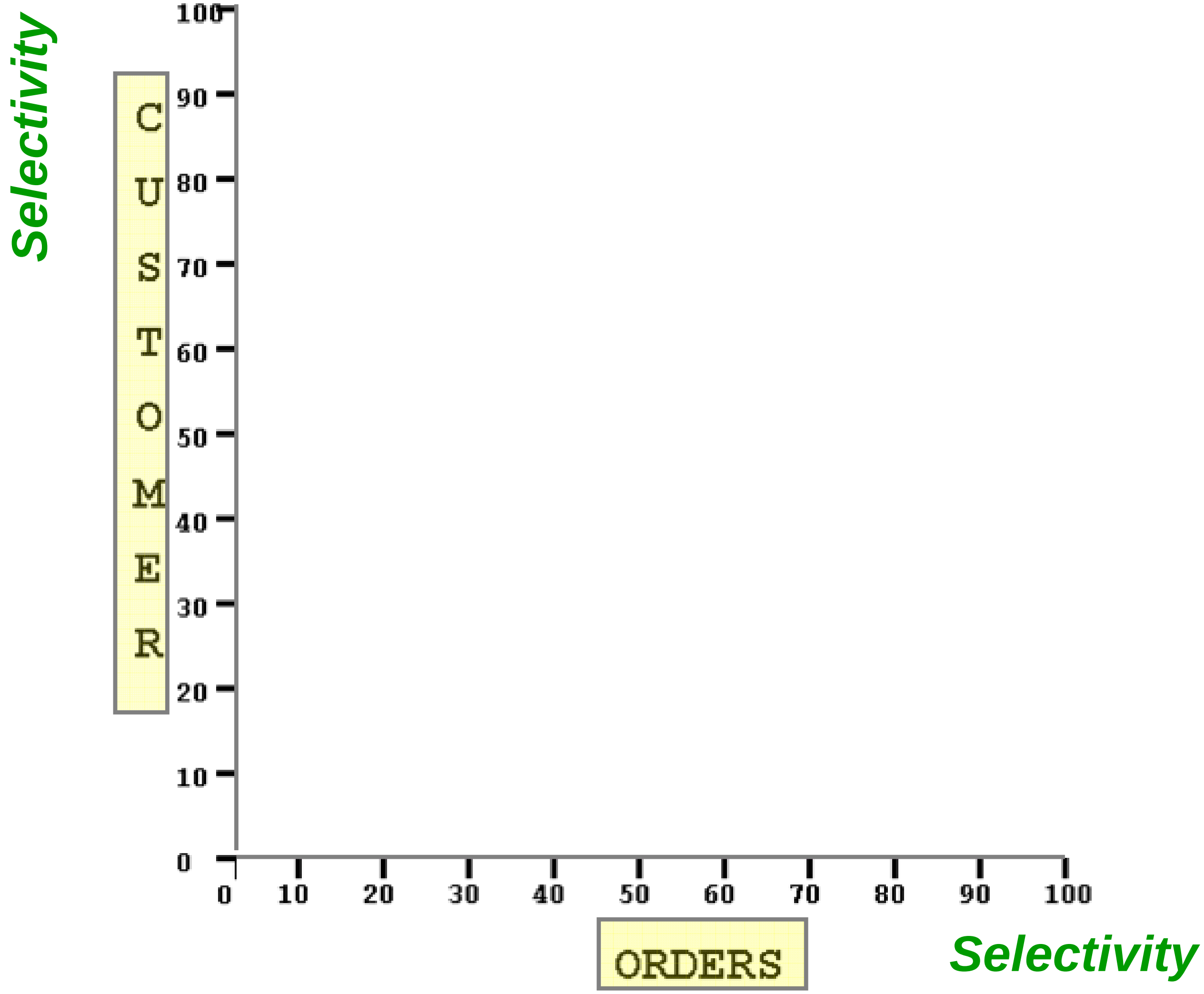
*Determines the values of goods shipped between nations in a time period*

```
select
   supp_nation, cust_nation, l_year, sum(volume) as revenue
from
(select n1.n_name as supp_nation,  n2.n_name as cust_nation,
          extract(year from l_shipdate) as l_year,
          l_extendedprice * (1 - l_discount) as volume
 from     supplier, lineitem, orders, customer, nation n1, nation n2
 wher                   okey and o_orderkey = l
                        key and s_nationkey = r
                        n2.n_nationkey and
        ((n1.n_name =    RANCE' and n2.n_name = 'GE    NY') or
         (n1.n_name = 'G   RMANY' and n2.n_name = '   ANCE')) and
         l_shipdate between   ate '1995-01-01' and da  e '1996-12-31'
          and  o_totalprice < C1  and  c_acctbal < C2   )  as shipping
  group by supp_nation, cust_nation, l_year
  order by supp_nation, cust_nation, l_year
```

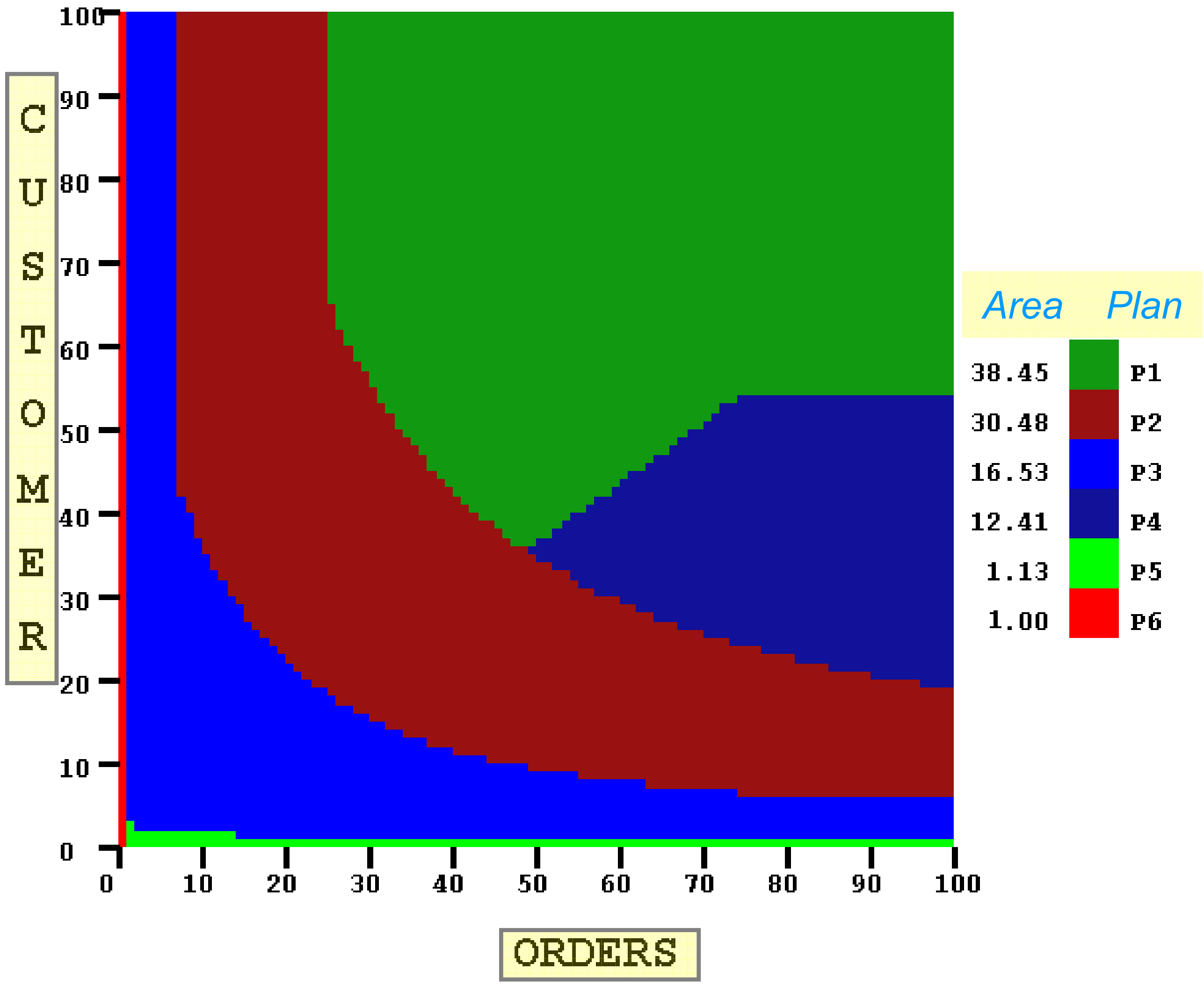
Value determines selectivity of ORDERS relation

Value determines selectivity of CUSTOMER relation

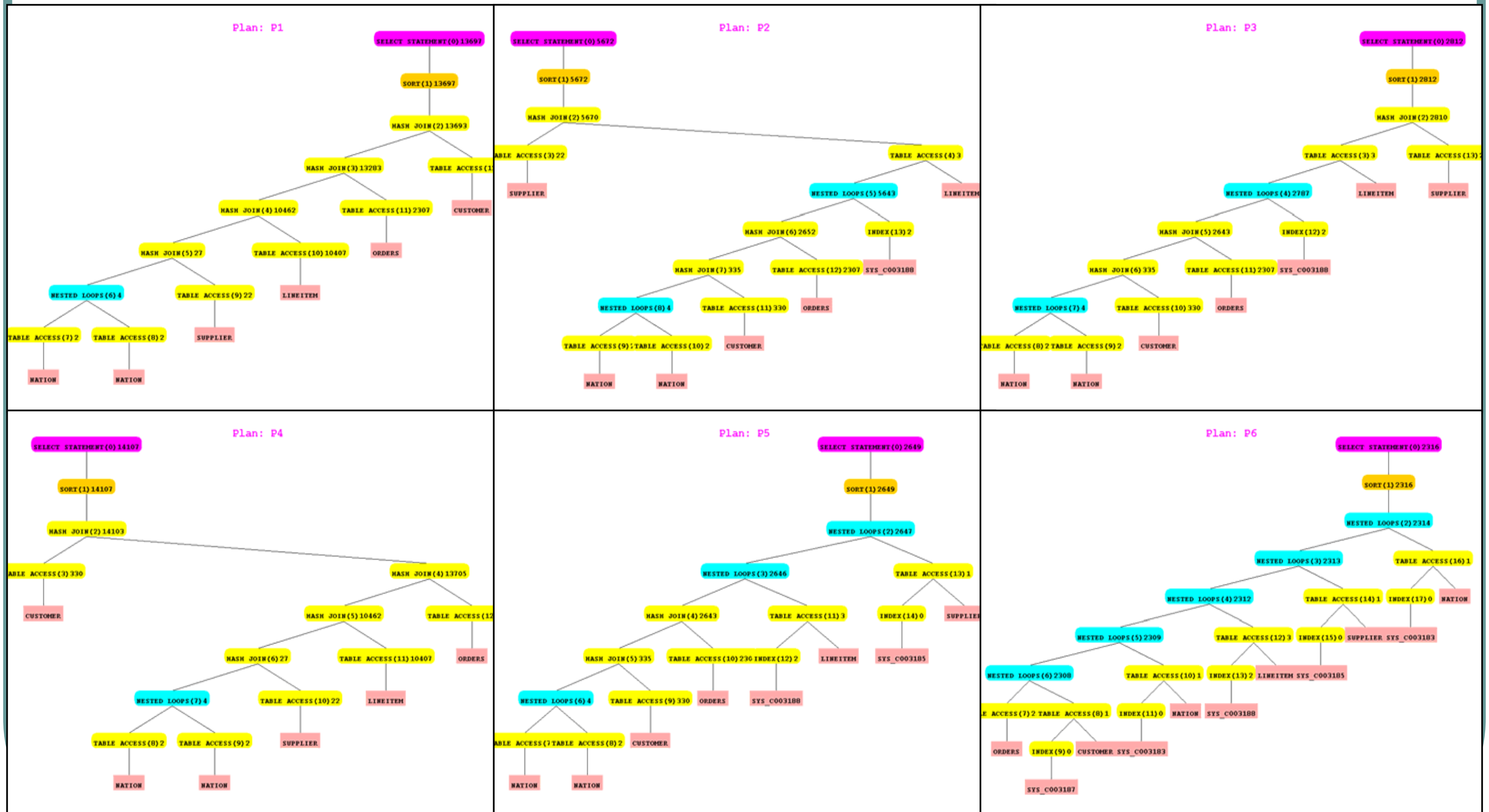
# Plan Diagram for Example Query



# Plan Diagram for Example Query

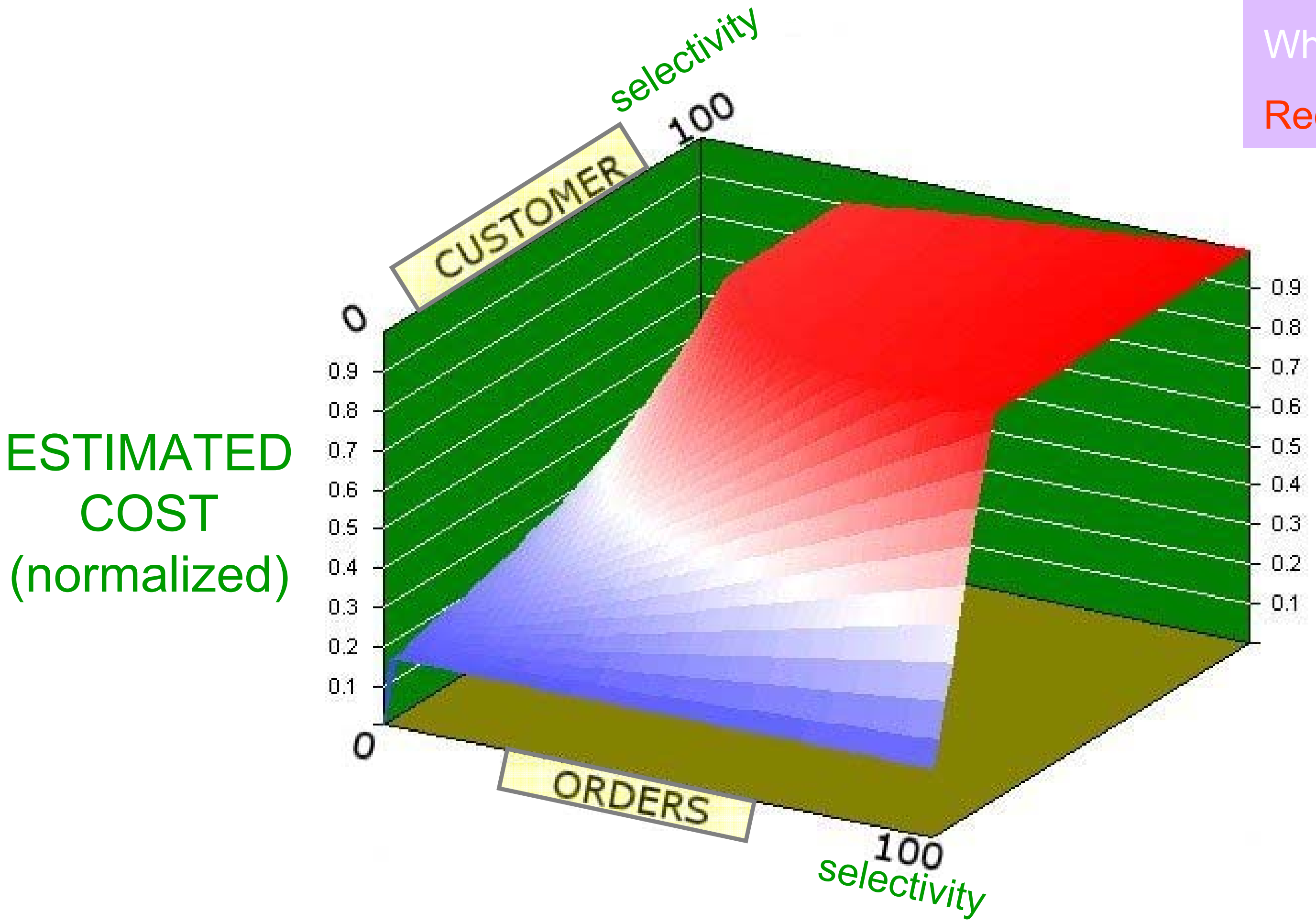


# Specific Plan Choices



# Cost Diagram for Example Query

| Blue: | Low cost |
| --- | --- |
| White: | Medium cost |
| Red: | High cost |



# PICASSO

# Picasso

- A Java tool that, given a query template, automatically generates plan and cost diagrams
  - Fires queries at user-specified granularity (default 100x100 grid)
  - Currently produces 2-D plan diagrams and 3-D cost diagrams
- Using the tool, enumerated the plan/cost diagrams produced by industrial-strength query optimizers on TPC-H-based queries
  - IBM DB2 v8, Oracle 9i, Microsoft SQL Server 2000
    - Oracle 10g and SQL Server 2005 soon
    - Ports of Picasso to Sybase and PostgreSQL ongoing

# Picasso Output

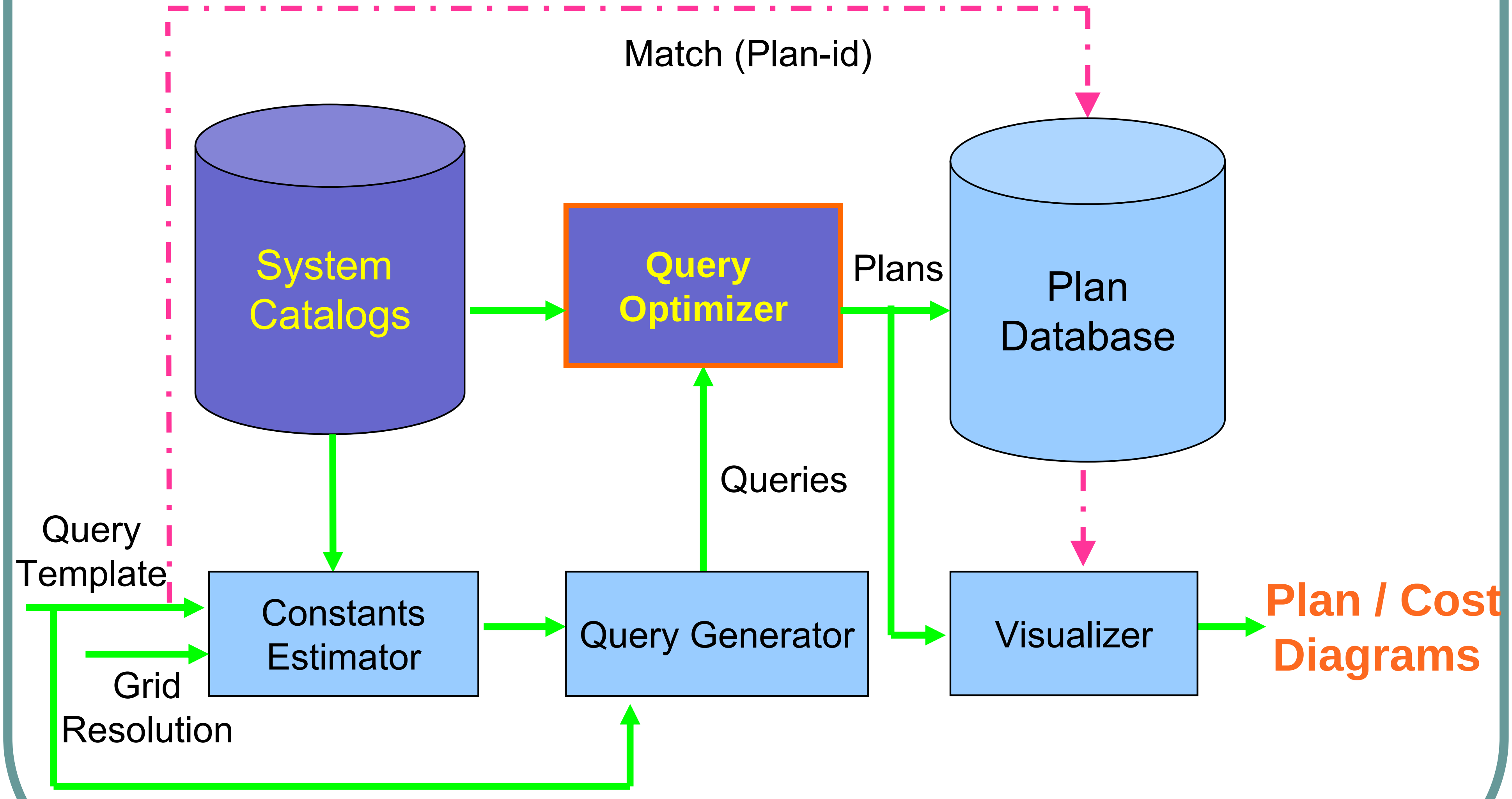
- Plan diagrams often similar to cubist paintings

  [ Pablo Picasso – founder of cubist genre ]

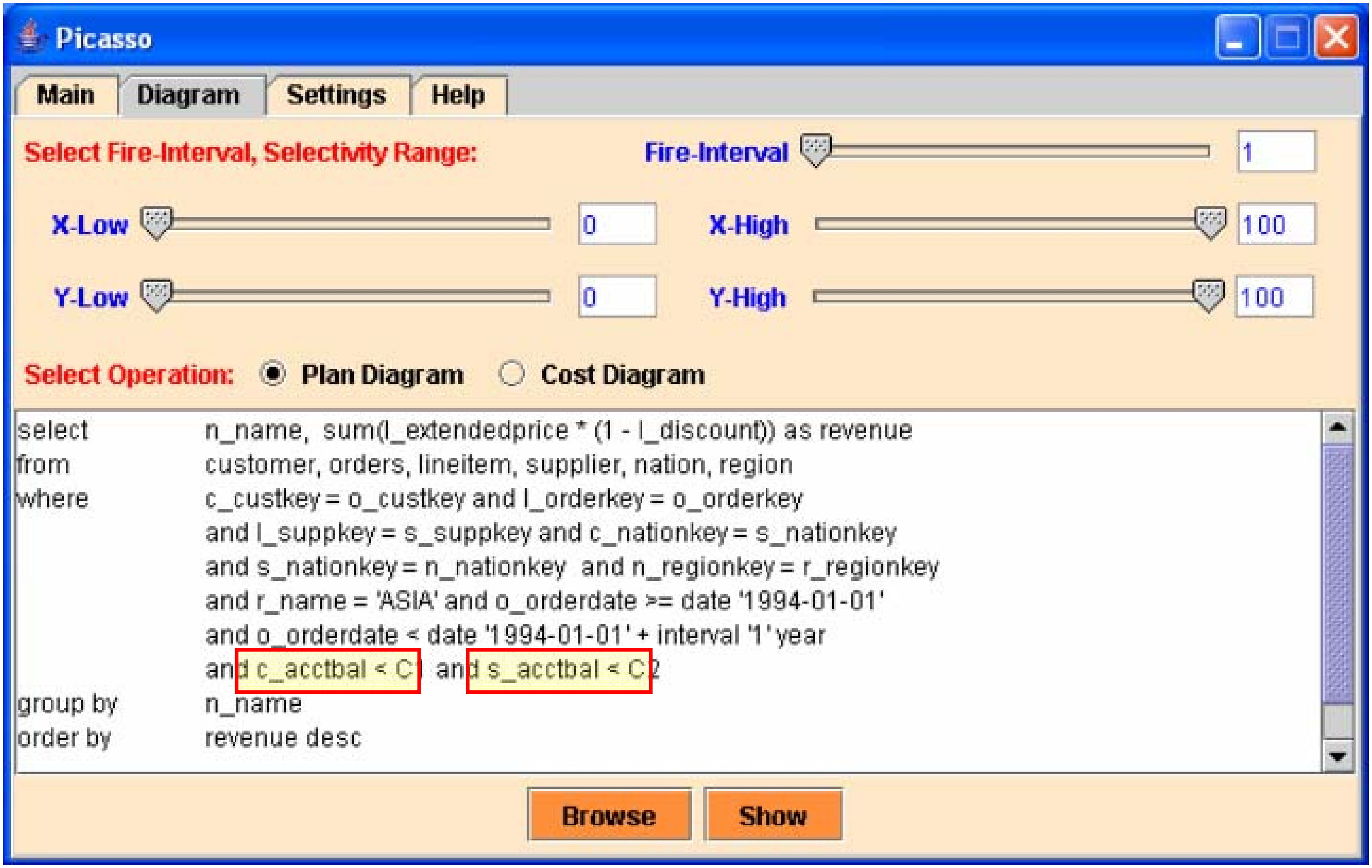
*Woman with a guitar*

Georges Braque, 1913

# Picasso Architecture



# Picasso GUI


Picasso
Main
Diagram
Settings
Help
Select Fire-Interval, Selectivity Range:
Fire-Interval
1
X-Low
0
X-High
100
Y-Low
0
Y-High
100
Select Operation:
Plan Diagram
Cost Diagram
select n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
from customer, orders, lineitem, supplier, nation, region
where c_custkey = o_custkey and l_orderkey = o_orderkey
and l_suppkey = s_suppkey and c_nationkey = s_nationkey
and s_nationkey = n_nationkey and n_regionkey = r_regionkey
and r_name = 'ASIA' and o_orderdate >= date '1994-01-01'
and o_orderdate < date '1994-01-01' + interval '1' year
and c_acctbal < C1 and s_acctbal < C2
group by n_name
order by revenue desc
Browse
Show

# Testbed Environment

- Database
  - TPC-H database (1 GB scale) representing a manufacturing environment, featuring the following relations:
- Query Set
  - Queries based on TPC-H benchmark [Q1 through Q22]
  - Uniform 100x100 grid (10000 queries) [0.5%, 0.5%] to [99.5%, 99.5%]
- Relational Engines
  - Default installations (with all optimization features on)
  - Stats on all columns; no extra indices
- Computational Platform
  - Pentium-IV 2.4 GHz, 1GB RAM, Windows XP Professional

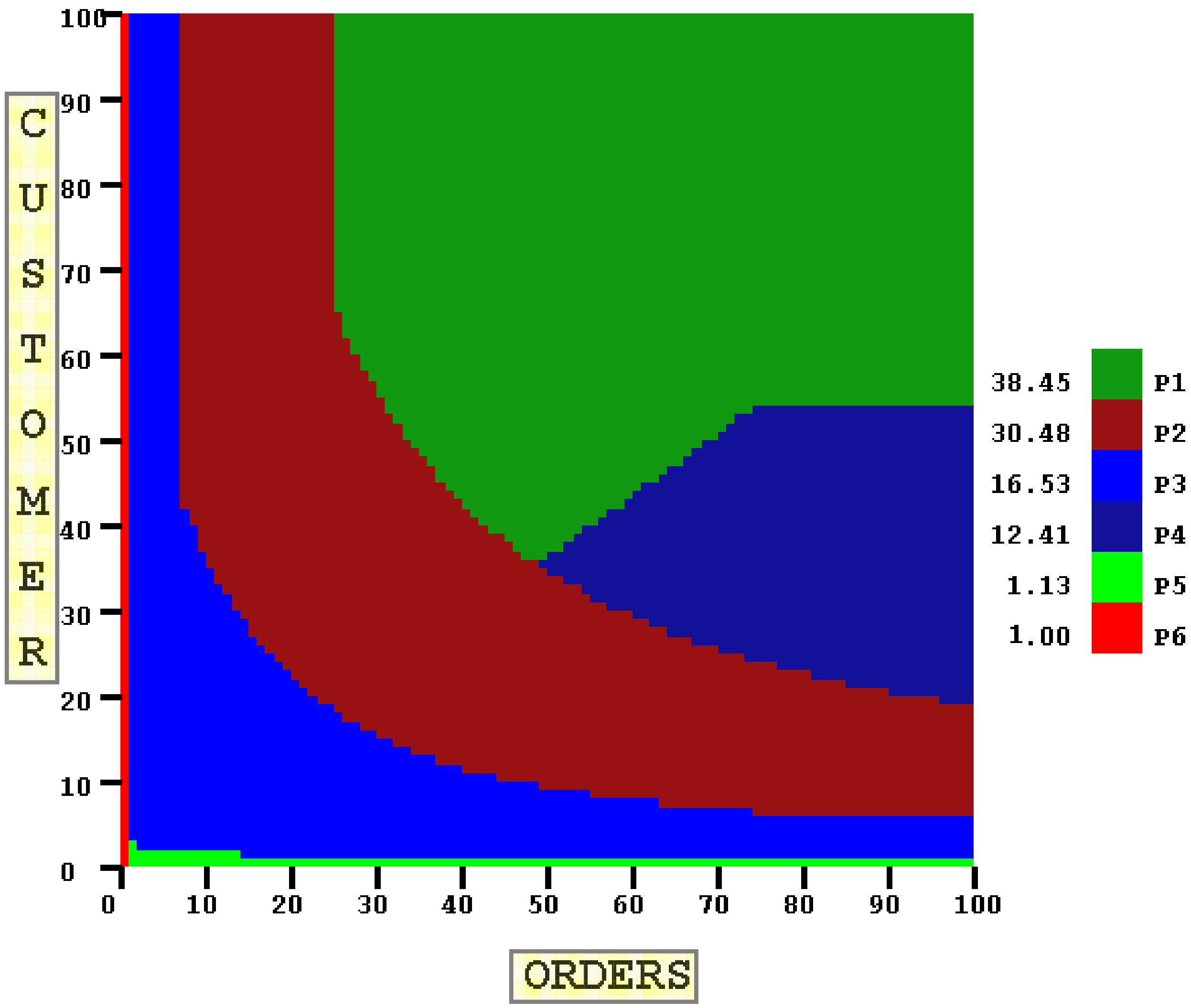
| Relation | Cardinality |
|---|---|
| REGION | 5 |
| NATION | 25 |
| SUPPLIER | 10000 |
| CUSTOMER | 150000 |
| PART | 200000 |
| PARTSUPP | 800000 |
| ORDERS | 1500000 |
| LINEITEM | 6001215 |

# RESULTS

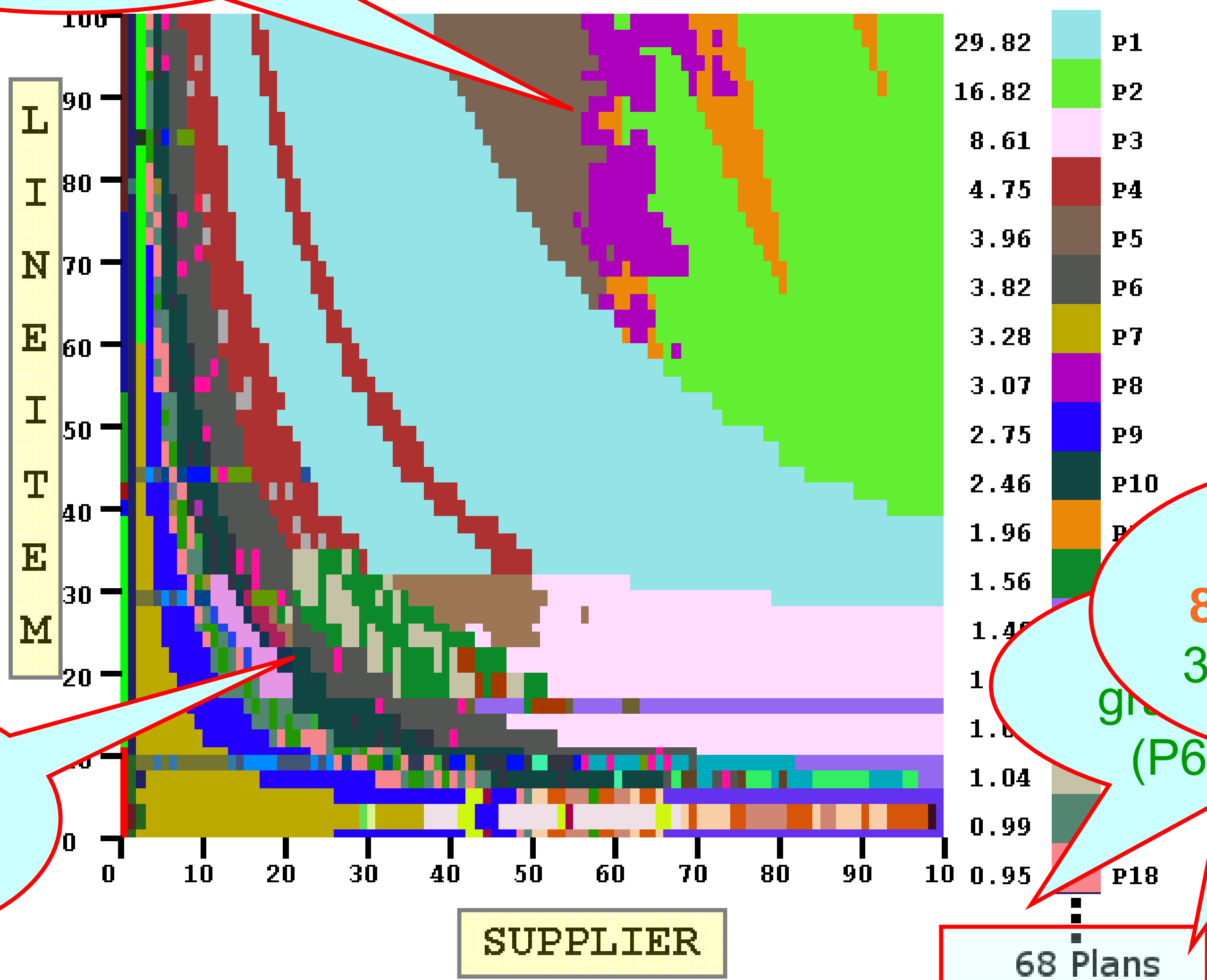
- Optimizers randomly identified as Opt A, Opt B, Opt C
- NOT intended to make comparisons across optimizers
- Black-box testing ⇒ our remarks are speculative

- Full result listing at http://dsl.serc.iisc.ernet.in/projects/PICASSO

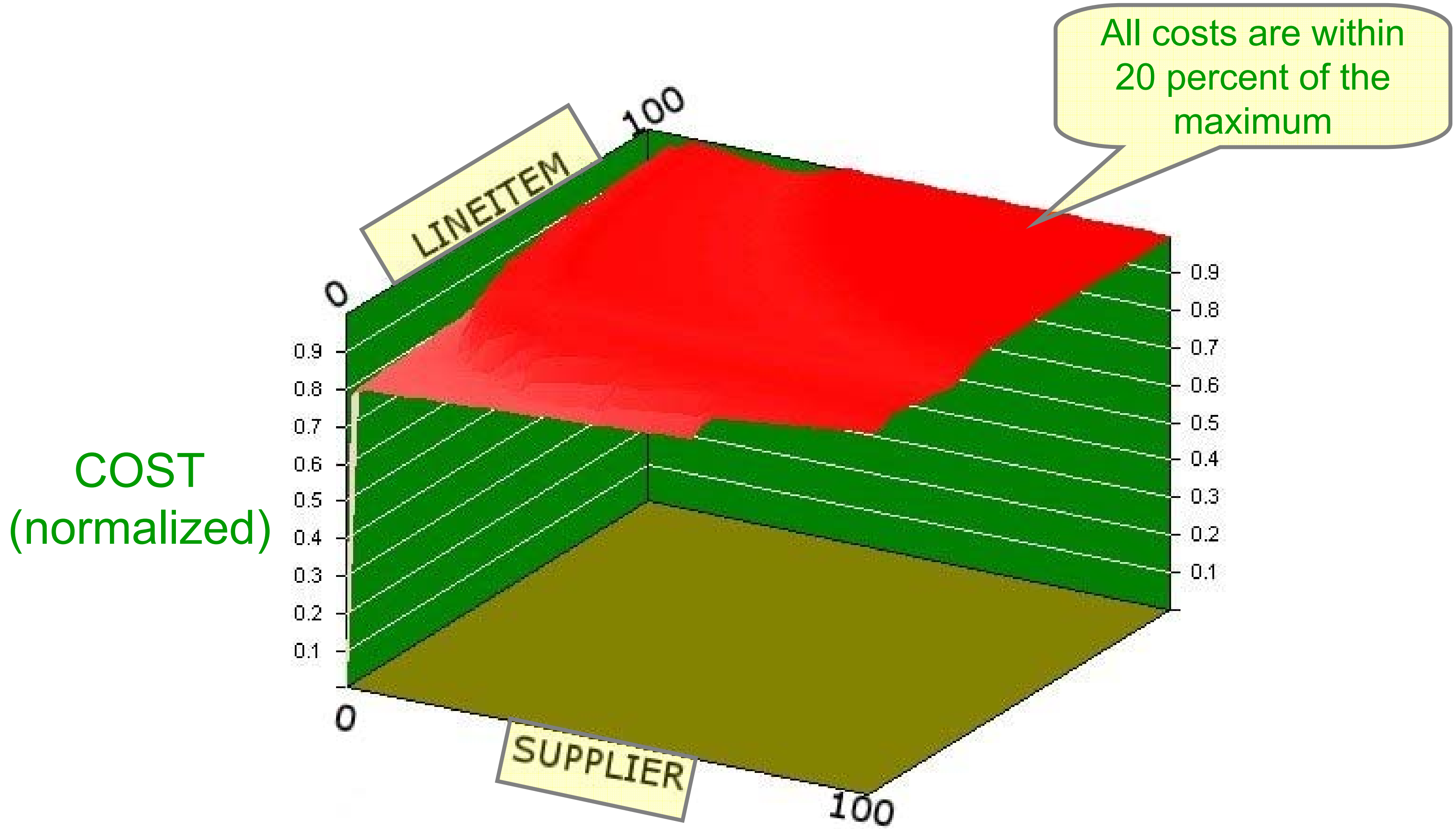
# Smooth Plan Diagram [Q7, Opt B]



# Comp… Plan Diagram [Q8, Opt A*]



# Cost Diagram [Q8, Opt A*]



# Skew in Plan Space Coverage

**80-20 Rule**
**Gini skew index > 0.7**

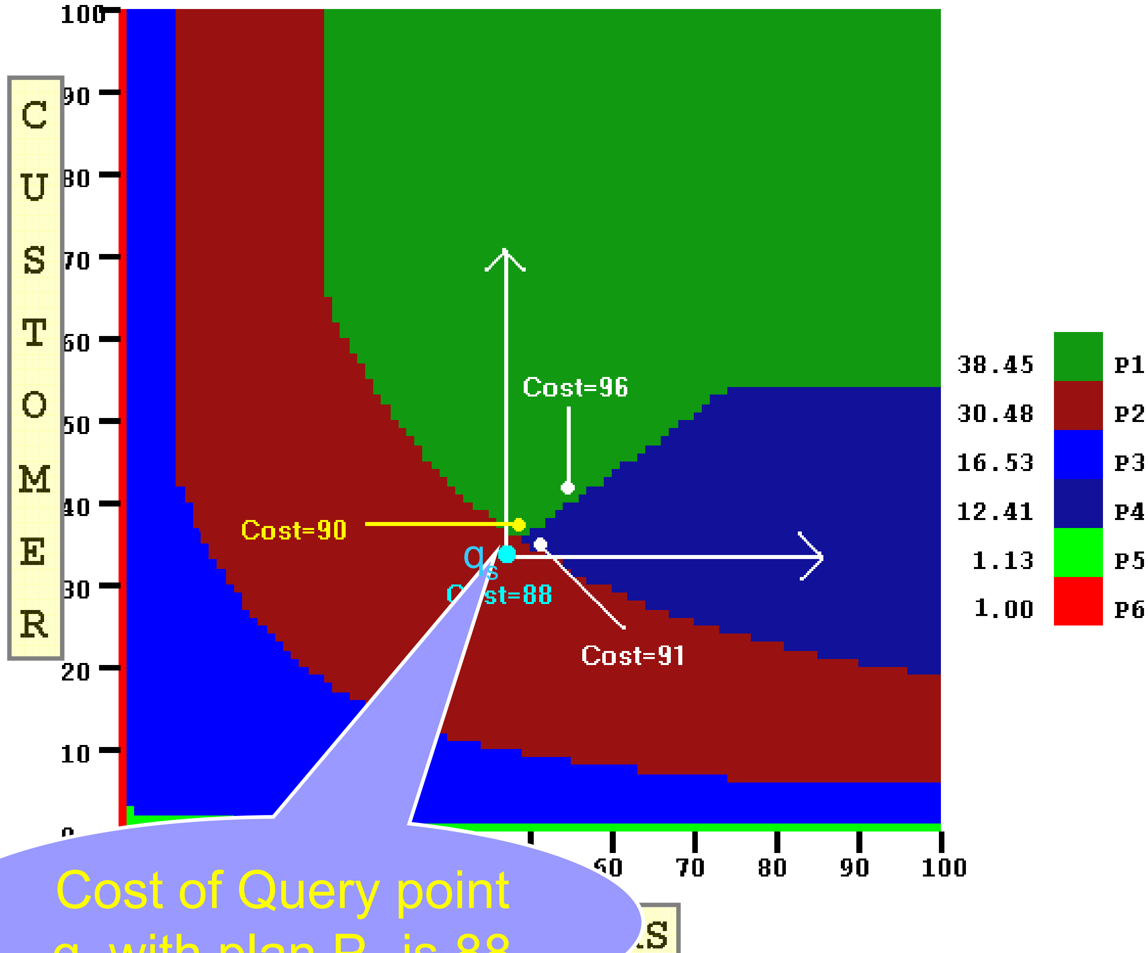
| TPC-H Query | Opt A Plan Cardinality | Opt A 80% Coverage | Opt A Gini Index | Opt B Plan Cardinality | Opt B 80% Coverage | Opt B Gini Index | Opt C Plan Cardinality | Opt C 80% Coverage | Opt C Gini Index |
|---|---|---|---|---|---|---|---|---|---|
| 2 | | | | | | | | | |
| 5 | | | | | | | | | |
| 7 | | | | | | | | | |
| **8** | **31** | | | **25** | | | **38** | | |
| **9** | **63** | | | **44** | | | **41** | | |
| 10 | | | | | | | | | |
| **18** | **5** | | | **13** | | | **5** | | |
| 21 | | | | | | | | | |
| **Avg(dense)** | | | | | | | | | |

Dense ⇒ Plan Cardinality ≥ 10

# Remarks

- Modern optimizers tend to make extremely fine-grained and skewed choices
  - even these stats are conservative (100x100 grid) !
- Is this an over-kill, perhaps not merited by the coarseness of the underlying cost space – i.e. are optimizers "doing too good a job" ?
- Is it feasible to reduce the plan diagram complexity without materially affecting the plan quality?

# Cost Domination Principle




Cost of Query point $q_s$ with plan $P_2$ is 88

Cost of executing any "foreign" query point in the first quadrant of $q_s$ is an upper bound on the cost of executing the foreign plan at $q_s$

Cost of executing $q_s$ with foreign plan $P_4$ or $P_1$ is less than or equal to 91 or 90, respectively.

# Formal Definition

- Dominating Point

  Given a pair of distinct poi
  2-D selectivity space, we s
  $x_2 \geq x_1$, $y_2 \geq y_1$ and result ca

> Intuition:
> more input + more output
> ⇒ more work ⇒ more cost

- Cost Domination Principle

  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, and $q_2 \succ q_1$, the cost of executing query $q_1$ with plan $P_2$ is upper-bounded by the cost of executing $q_2$ with $P_2$

# Caution on CDP:

- Sometimes not followed by commercial optimizers (as we shall see later)
- Also a few genuine cases where the principle does not hold

# Plan Cardinality Reduction

1. Order the plans in
list, checking for po

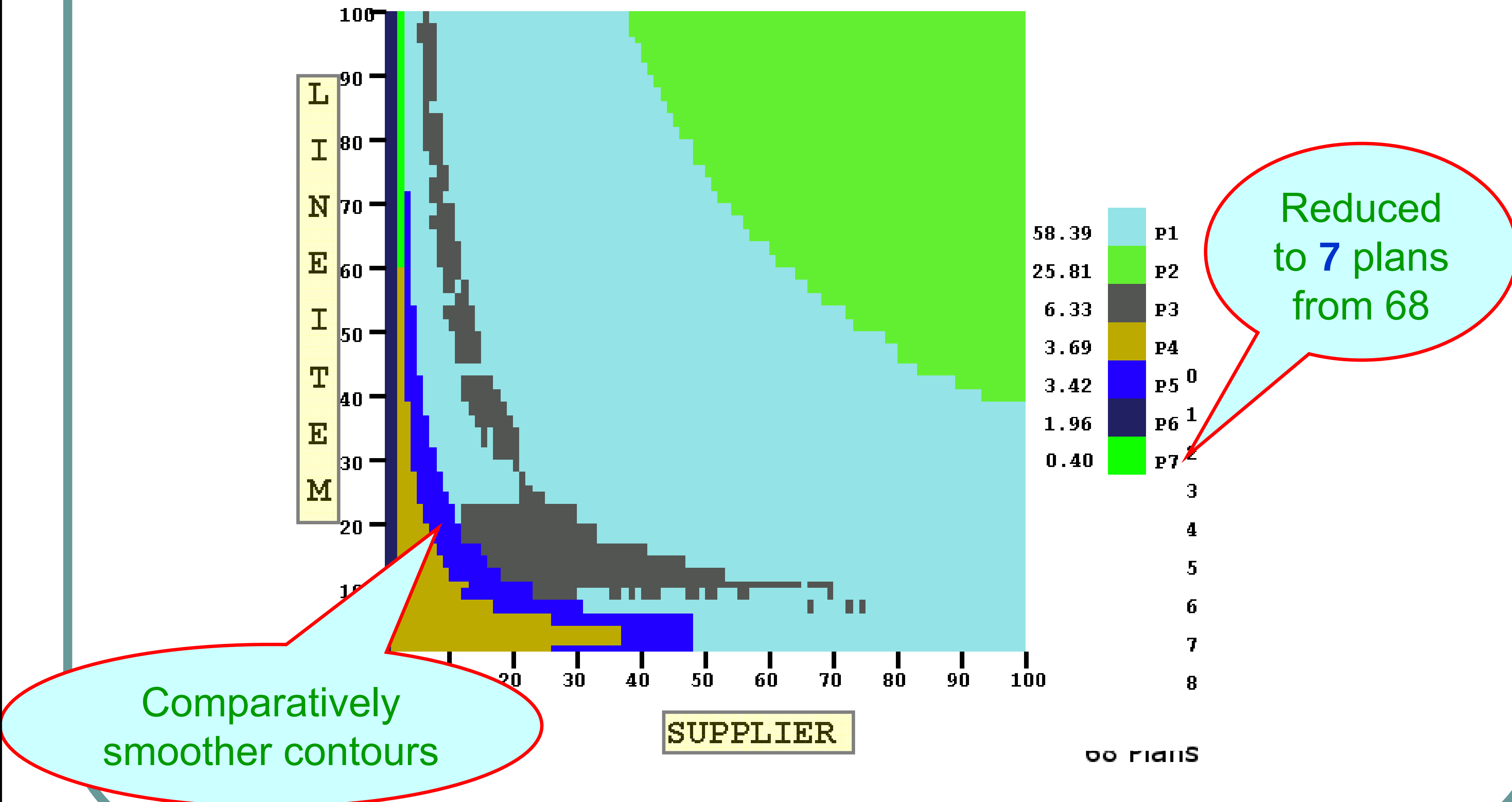
2. Given plan p, for e
replacements by "f
first quadrant relat

Guarantee: *No query point in the original space has its (estimated) cost increased, post-swallowing, by more than λ percent*

3. For the foreign points that are within λ (e.g. λ=10%) cost degradation threshold, choose the point with lowest cost as potential replacement.

4. An entire plan is "swallowed" only if all its query points are replaceable by single plan or group of plans.

# Complex Plan Diagram [Q8] / Reduced Plan Diagram [λ=10%*]



# Plan Cardinality Reduction by Swallowing
# (λ =10%)

**Average Cost Increase < 2%**

| TPC-H Query | Opt A % Card Decrease | Opt A Avg Cost Increase | Opt A Max Cost Increase | Opt B % Card Decrease | Opt B Avg Cost Increase | Opt B Max Cost Increase | Opt C % Card Decrease | Opt C Avg Cost Increase | Opt C Max Cost Increase |
|---|---|---|---|---|---|---|---|---|---|
| 2 | | | | | | | | | |
| 5 | | | | | | | | | |
| 7 | | | | | | | | | |
| 8 | 87.6 | 0.4 | 9.4 | 84.0 | 0.9 | 9.1 | 86.8 | 1.2 | 8.4 |
| 9 | | | | | | | | | |
| 10 | | | | | | | | | |
| 18 | | | | | | | | | |
| 21 | | | | | | | | | |
| Avg(dense) | | | | | | | | | |

# Note:

- A 10% threshold is *well within* the confidence intervals of the cost estimates of modern optimizers
- The average and maximum degradation values are *upper bounds* – the actual costs may be even lower in practice
- Plan Cardinality Reduction ≠ Change in Optimization Levels

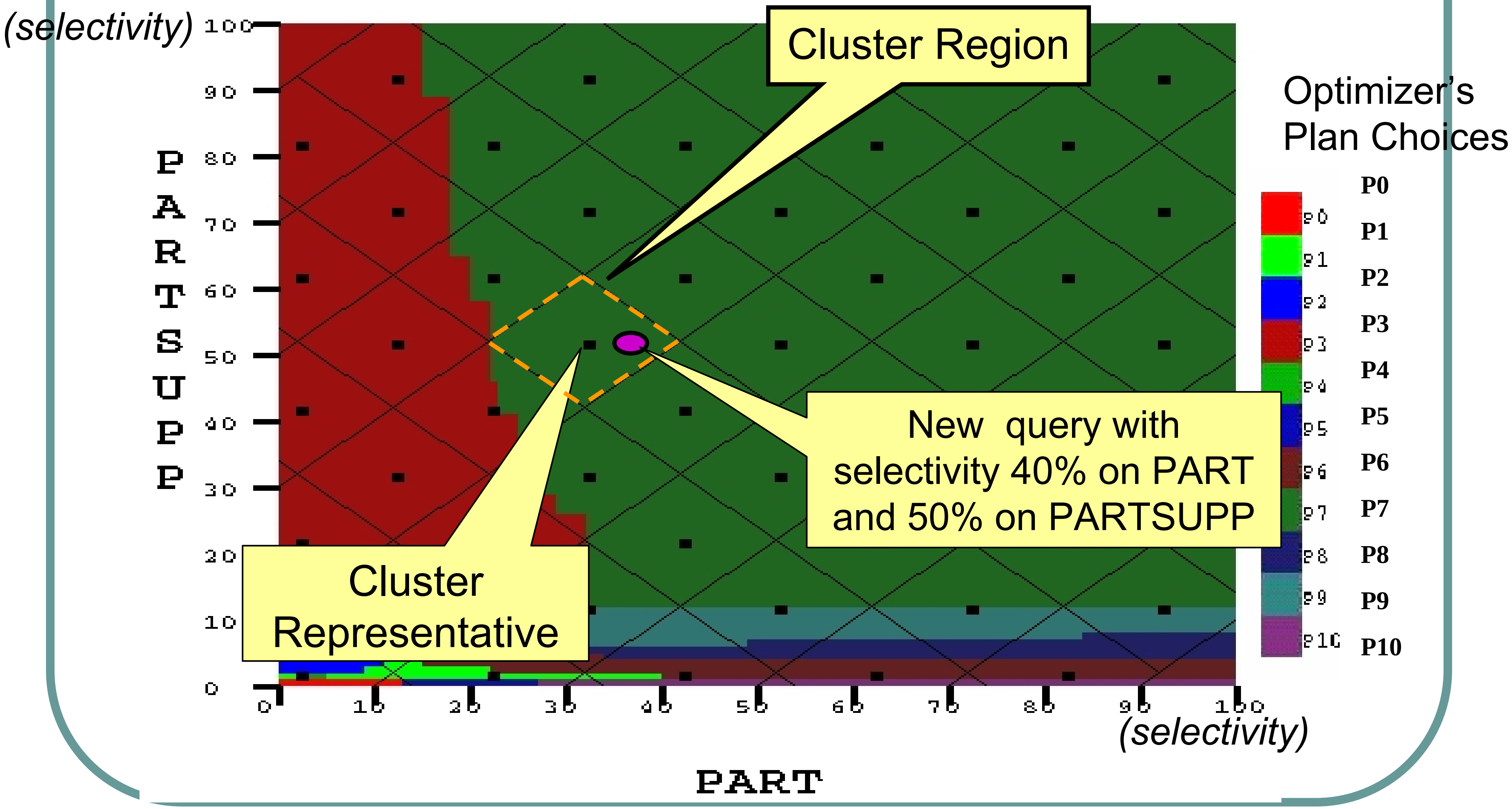
# Remarks

- "Two-thirds of the plans in a dense plan diagram are liable to be eliminated through plan swallowing, without materially affecting query processing quality."
- Would it be possible to simplify current optimizers to produce only reduced plan diagrams, perhaps leading to a lowering of the high computational overheads associated with query optimization?
  - Open research question …

# Indirect Reduction Approach

- Notion of reduction fits in perfectly with our earlier PLASTIC [VLDB 2002] approach of plan recycling based on query clustering, since cluster regions *inherently coarsen* the plan diagram granularity.

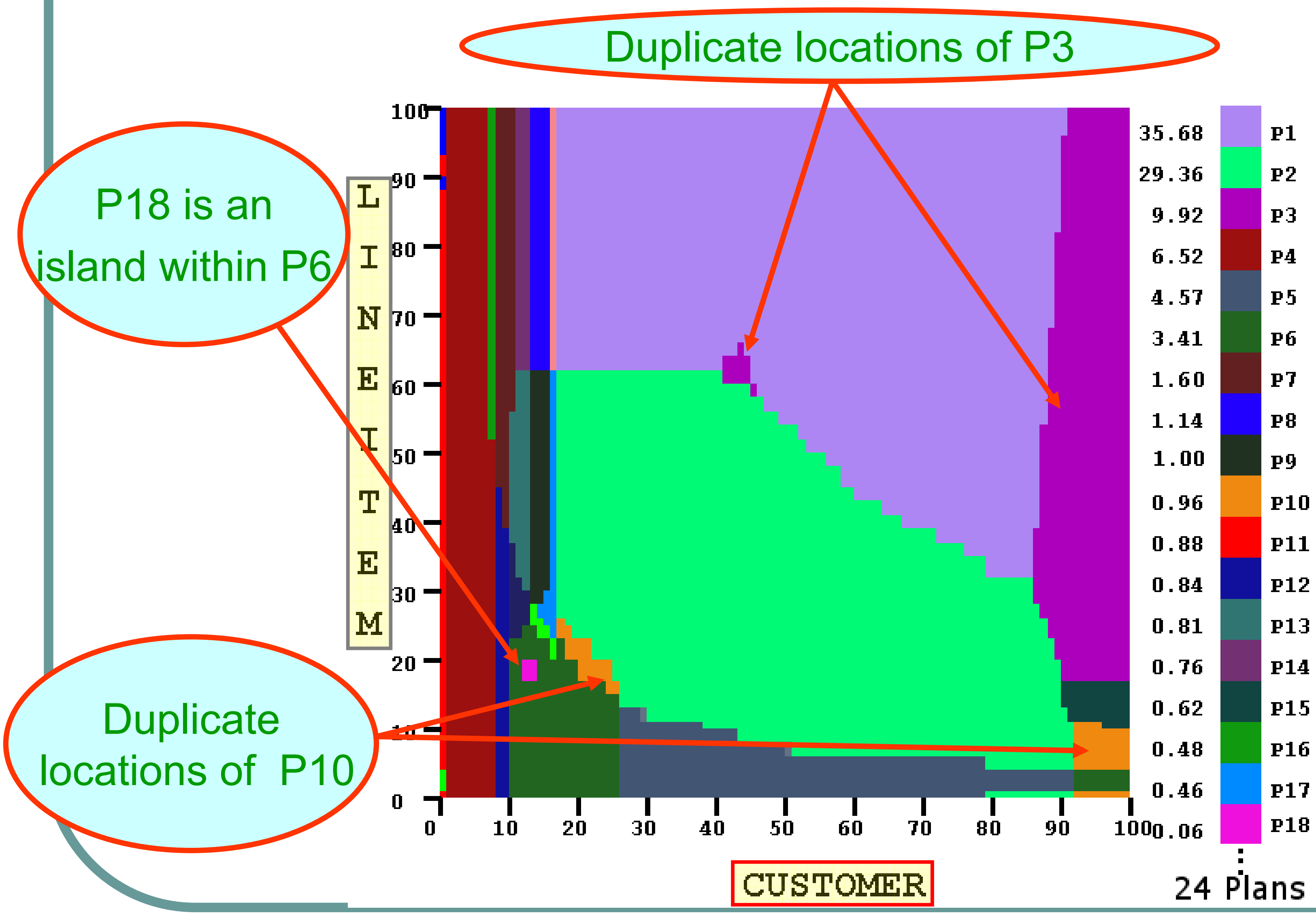
# PLASTIC Cluster Diagram



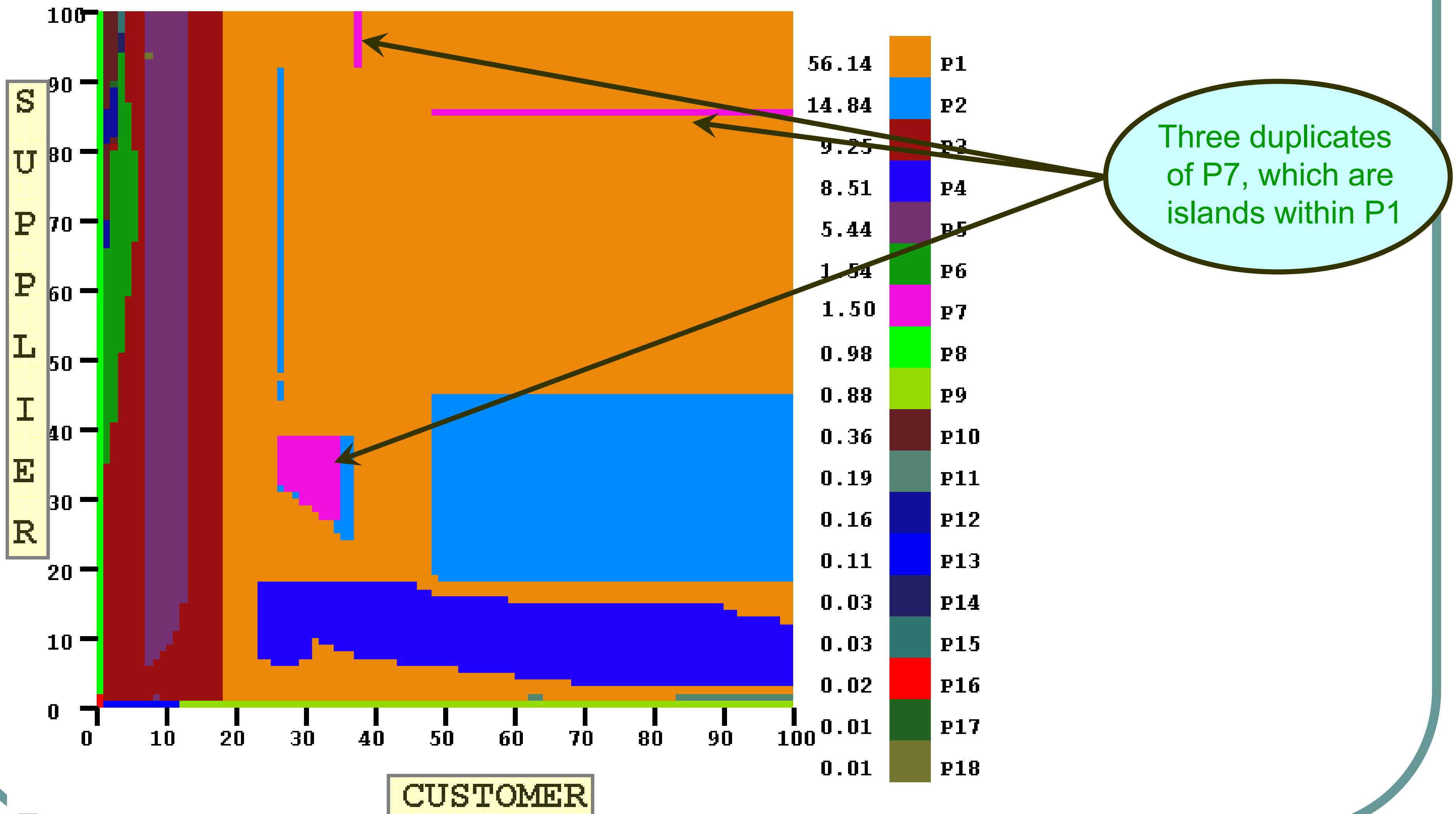
# Pattern Gallery

- Duplicates and Islands
- Plan Switch Points
- Footprint Pattern
- Speckle Pattern

# Duplicates and Islands [Q10, Opt A]



# Duplicates and Islands [Q5, Opt C]

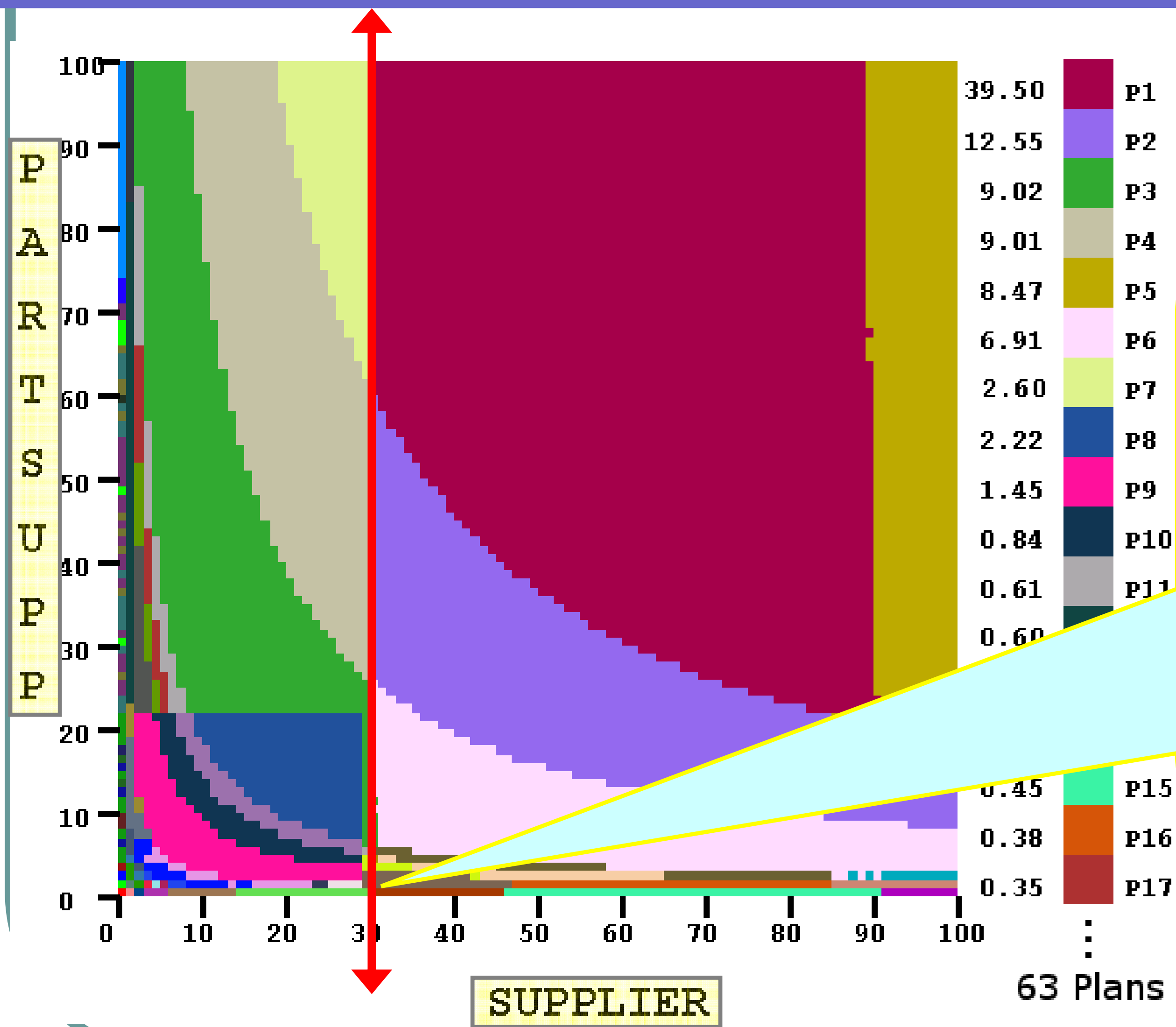


# Duplicates and Islands Removal

| Databases | # Duplicates Original | # Duplicates Reduced [λ=10%] | # Islands Original | # Islands Reduced [λ=10%] |
|---|---|---|---|---|
| Opt A | 130 | 13 | 38 | 3 |
| Opt B | 80 | 15 | 1 | 0 |
| Opt C | 55 | 7 | 8 | 3 |

- With Plan Reduction by Swallowing, significant decrease in duplicates and islands

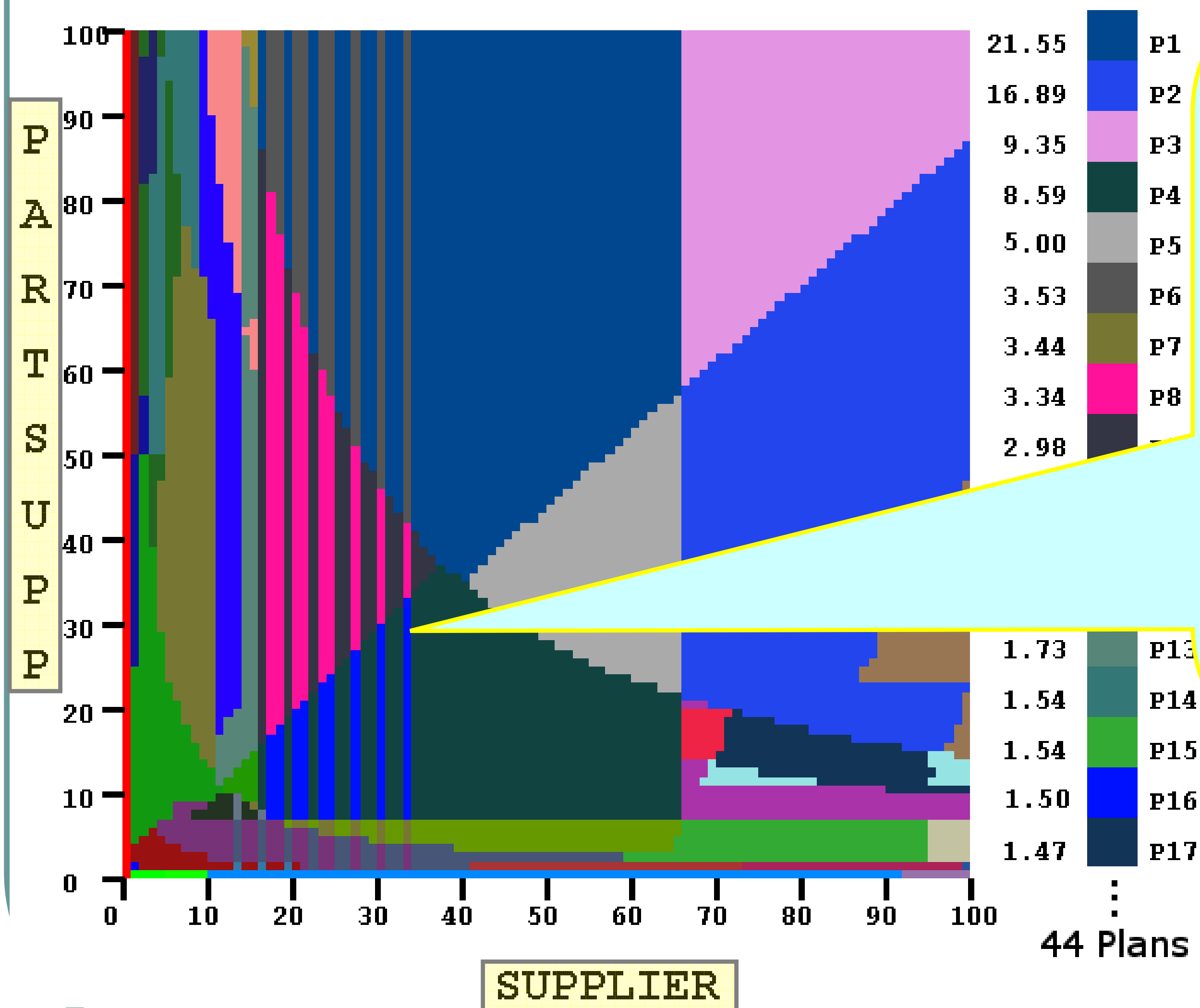
# Plan Switch Points [Q9, Opt A]




Plan Switch Point:
line parallel to axis with a plan shift for all plans bordering the line.

Hash-Join sequence
PARTSUPP▶◀SUPPLIER▶◀PART
is altered to
PARTSUPP▶◀PART▶◀SUPPLIER

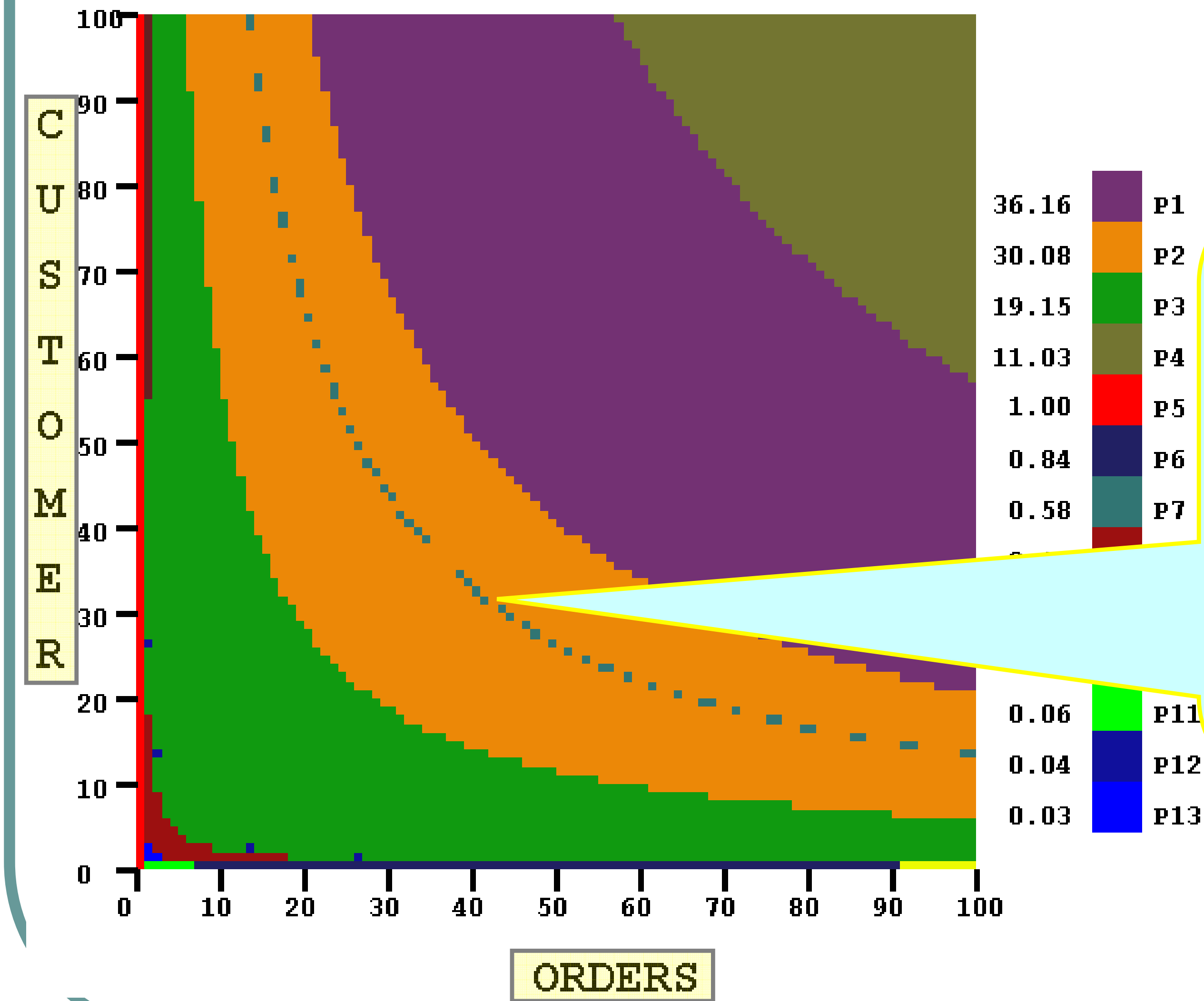
# Venetian Blinds [Q9, Opt B]




Six plans simultaneously change with rapid alternations to produce a "Venetian blinds" effect.

Left-deep hash join across NATION, SUPPLIER and LINEITEM relations gets replaced by a right-deep hash join.

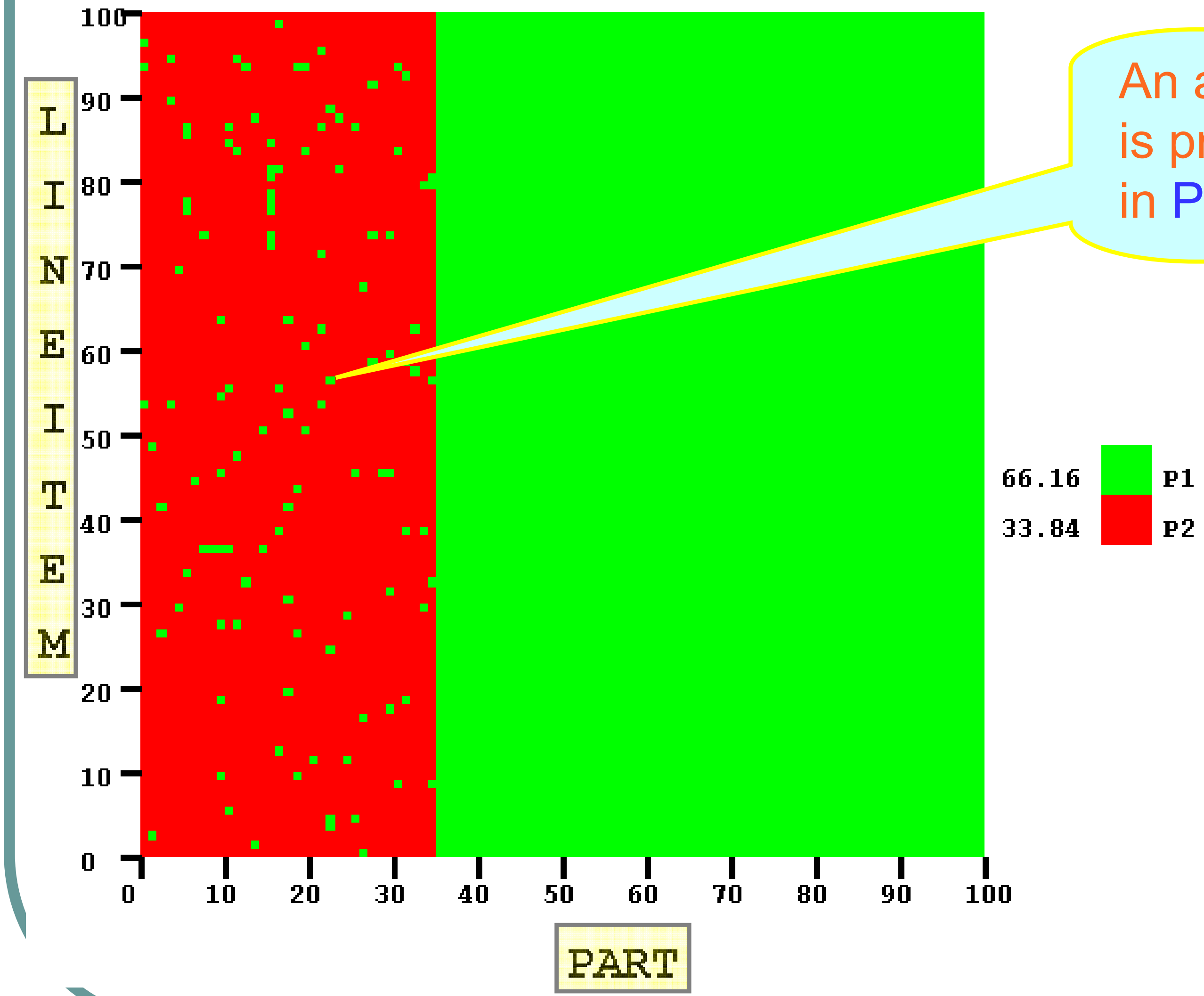
# Footprint Pattern [Q7, Opt A]




P7 is a thin and broken curved pattern in the middle of P2's region.

P2 has sort-merge-join at the top of the plan tree, while P7 uses hash-join

# Speckle Pattern [Q17, Opt A*]




An additional sort operation is present on PART relation in P2, whose cost is very low

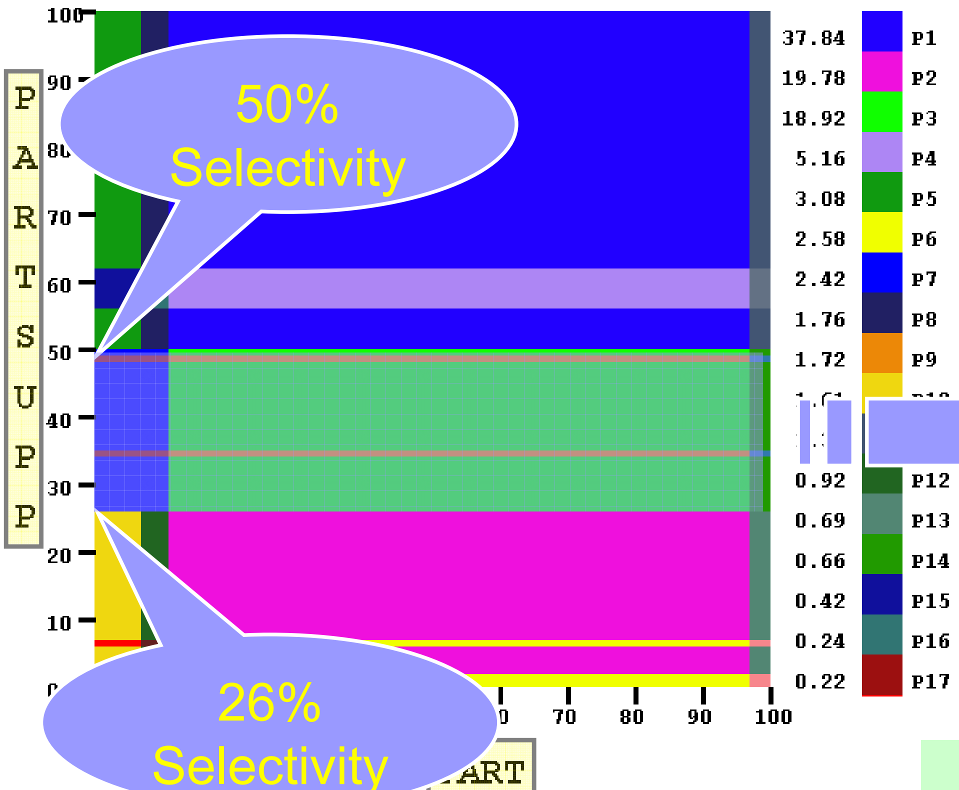
# Non-Monotonic Cost Behavior

- Plan-Switch Non-Monotonic Costs
- Intra-Plan Non-Monotonic Costs

# Plan-Switch Non-Monotonic Costs [Q2, Opt A]

**Presence of Rules?**

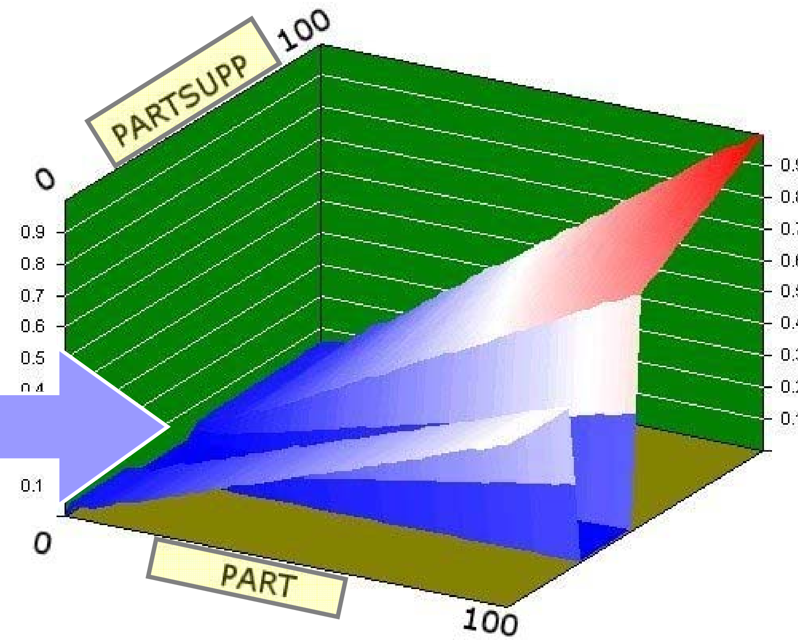
**Parameterized changes in search space?**




50% Selectivity

26% Selectivity

Plan Diagram

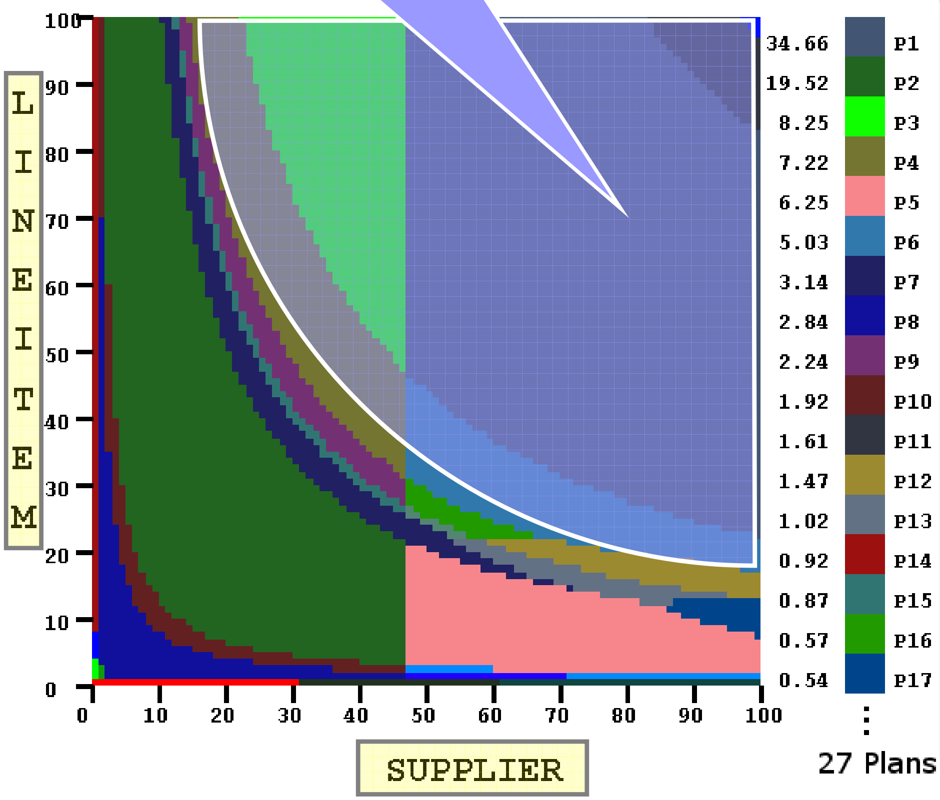


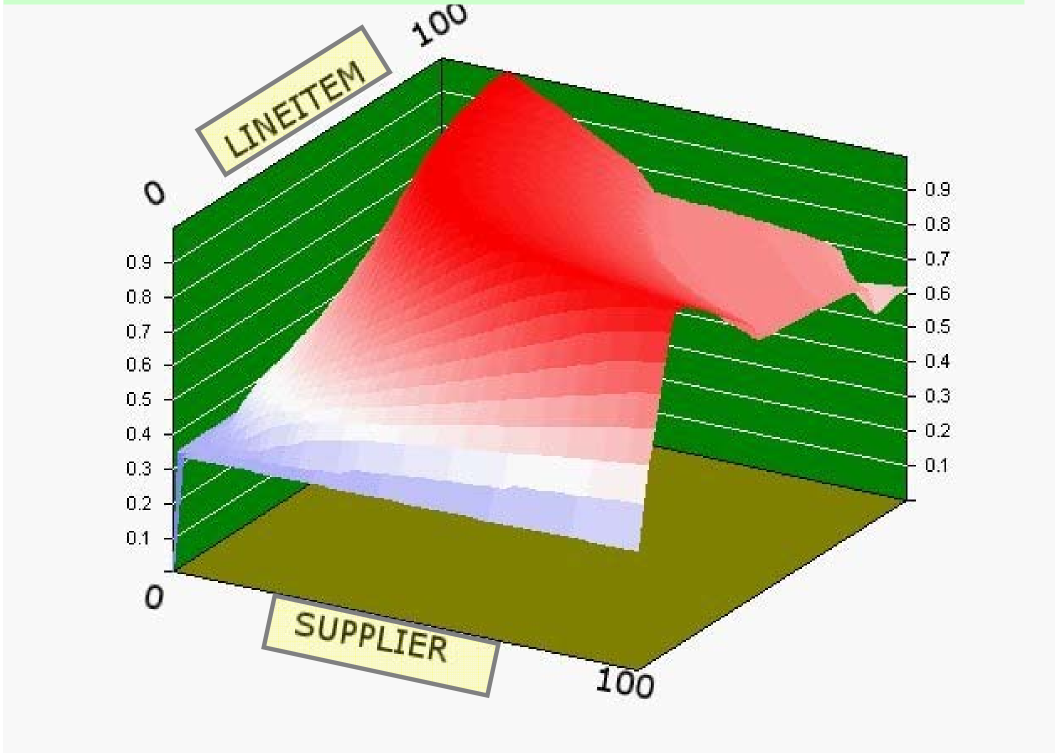

26%: Cost decreases by a factor of *50*
50%: Cost increases by a factor of *70*

# Intra-Plan Non-Monotonic Costs [Q21, Opt A]

Plans P1, P3, P4 and P6

*Nested loops join whose cost decreases with increasing input cardinalities*




Plan Diagram

Cost Diagram

# Remarks

- Optimizers may have become too complex over time, making it difficult to anticipate the interactions and side-effects of their modules
- Well-kept secret by optimizer developers? Perhaps worth having a re-look at optimizer design …

# Relationship to PQO

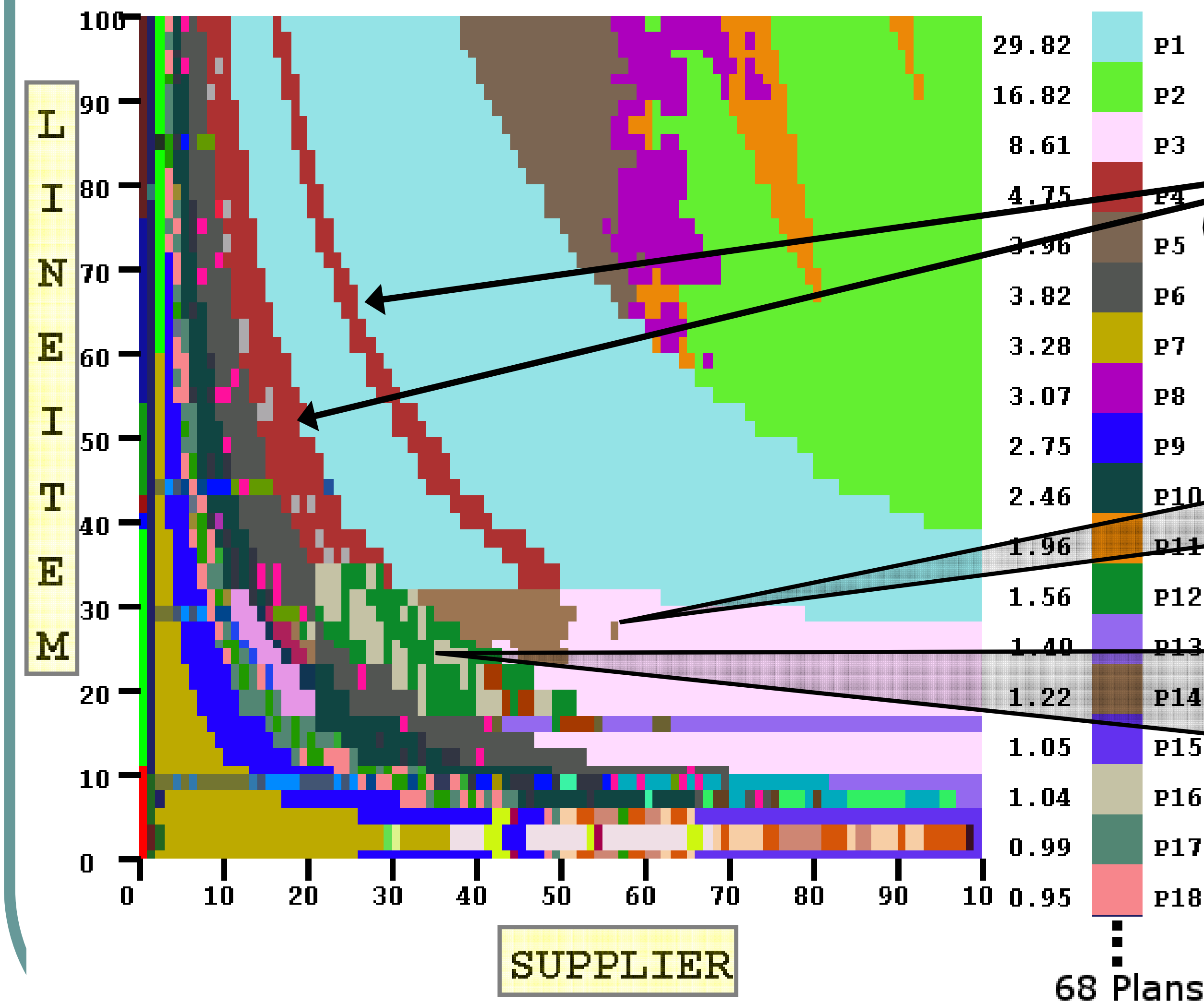
# PQO (Parametric Query Optimization)

- Active research area for last 15 years
  - VLDB 1992, 1998, 2002, 2003
  - IIT Kanpur (Sumit Ganguly), IIT Bombay (Sudarshan)
- Identify the optimal set of plans for the entire relational selectivity space at compile time
- At run time, use actual selectivity values to identify the appropriate plan choice

# PQO Assumptions

- *Plan Convexity:* If a plan is optimal at two points, then it is optimal at all points on the straight line joining them
- *Plan Uniqueness:* An optimal plan appears at only one contiguous region in the entire space
- *Plan Homogeneity:* An optimal plan is optimal within the entire region enclosed by its plan boundaries

# Validity of PQO [Q8, Opt A*]




Plan uniqueness is violated by P4

Plan Homogeneity is violated by P14

Plan Convexity is severely violated by regions covered by P12 (dark green) and P16 (light gray).

# Remarks:

- PQO assumptions do not hold, even approximately, in current optimizers
- But, PQO may be a more viable proposition in the world of reduced plan diagrams due to the removal of most duplicates and islands

# Conclusions

- Conceived and developed the **Picasso tool** for automatically generating plan and cost diagrams
  - optimizer debugger / research platform / teaching aid
- Presented and analyzed representative plan and cost diagrams on popular commercial query optimizers
  - Optimizers make *fine grained* choices
  - Plan optimality regions can have *intricate patterns* and *complex boundaries*
  - Complexity of plan diagrams can be *drastically reduced without materially affecting the query processing quality*
  - *Non-Monotonic* cost behavior exists where increasing input and result cardinalities decrease the estimated cost
  - Basic assumptions of PQO research literature on PQO *do not hold* in practice; hold approximately for reduced plan diagrams

# Recently Added Features of Picasso

- (estimated) Result Cardinality diagrams
- PlanDiff (highlight differences in plans)
- 3-D Integrated plan-cost diagrams
- 3-D Integrated plan-cardinality diagrams
- n-D Query Templates

# Work Involved in Porting

- Porting to a new dbms depends on
  - extent to which the dbms consistently uses tables to store internal data – e.g. plan steps, statistics
  - extent to which it exposes this data to SQL access
  - method used for computing selectivities

# Related Efforts

- Sumit Ganguly had considered many of these issues in set of (unpublished) MTech theses at IIT Kanpur [1999]
  - Home-brewed simple System-R style optimizer
  - Pure SPJ queries with star or linear join-graphs
  - Focus on coming up with theoretical formulas
- Arvind Hulgeri's Phd thesis [2003] at IIT Bombay evaluates cardinality of optimal plan set, and reduced plan sets in context of PQO and Volcano-style optimizer
- In contrast, our evaluation is in the context of "industrial-strength" queries and optimizers
  - we find high plan density even away from the axes
  - highly irregular optimality boundaries

# Take Away

*Query Optimization is truly an "art"* ☺

# Additional Information

- Paper: "Analyzing Plan Diagrams of Database Query Optimizers" [Proc. of VLDB 2005 Conference]
- Project Website
  http://dsl.serc.iisc.ernet.in/projects/PICASSO

# END PICASSO