Picasso: Drawing Out the Artistic Talents of DB Query Optimizers

Jayant Haritsa
Database Systems Lab
Indian Institute of Science

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
```

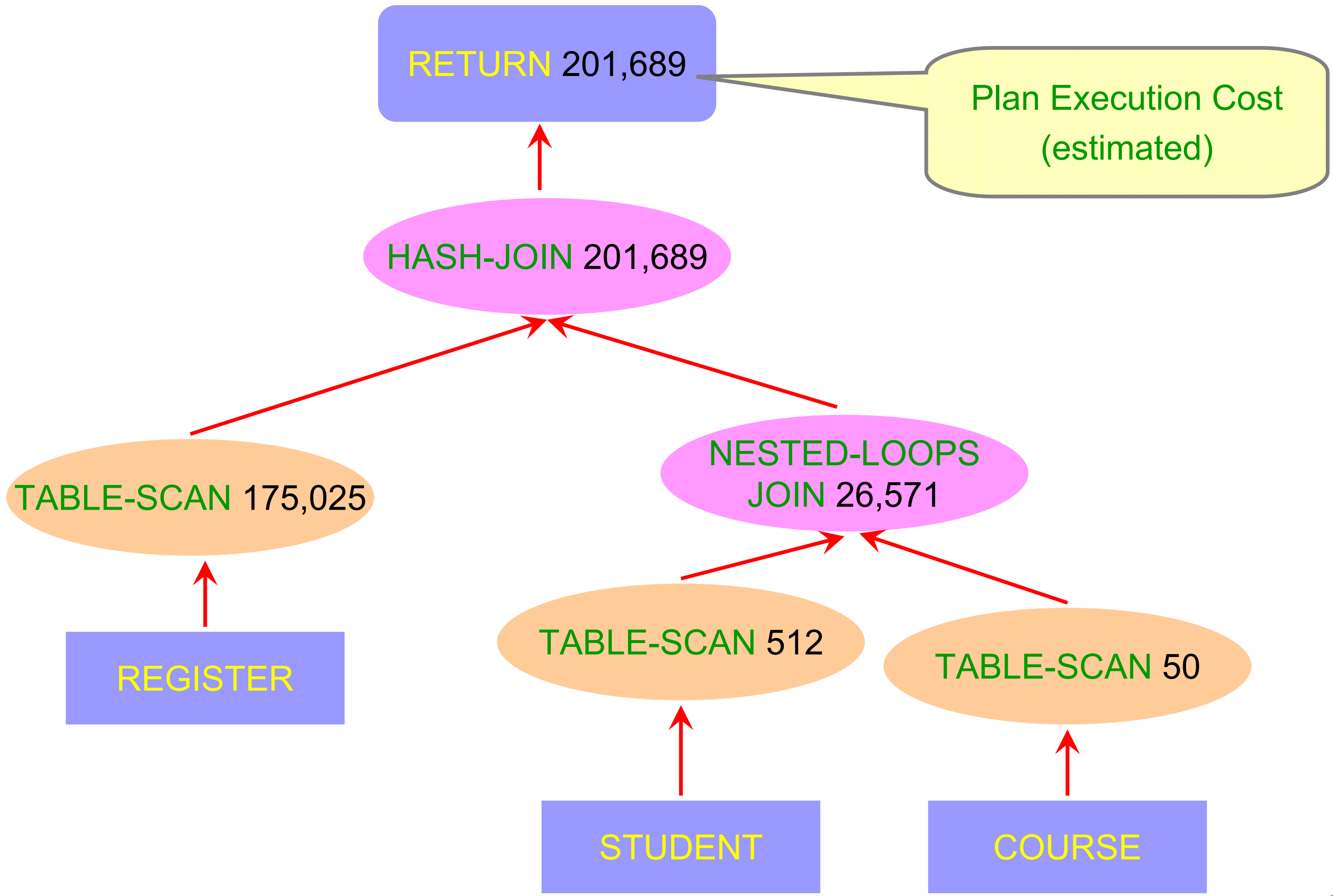
Unspecified:

join order [((S ⋈ R) ⋈ C) **or** ((R ⋈ C) ⋈ S) ?]

join techniques [Nested-Loops **or** Sort-Merge **or** Hash ?]

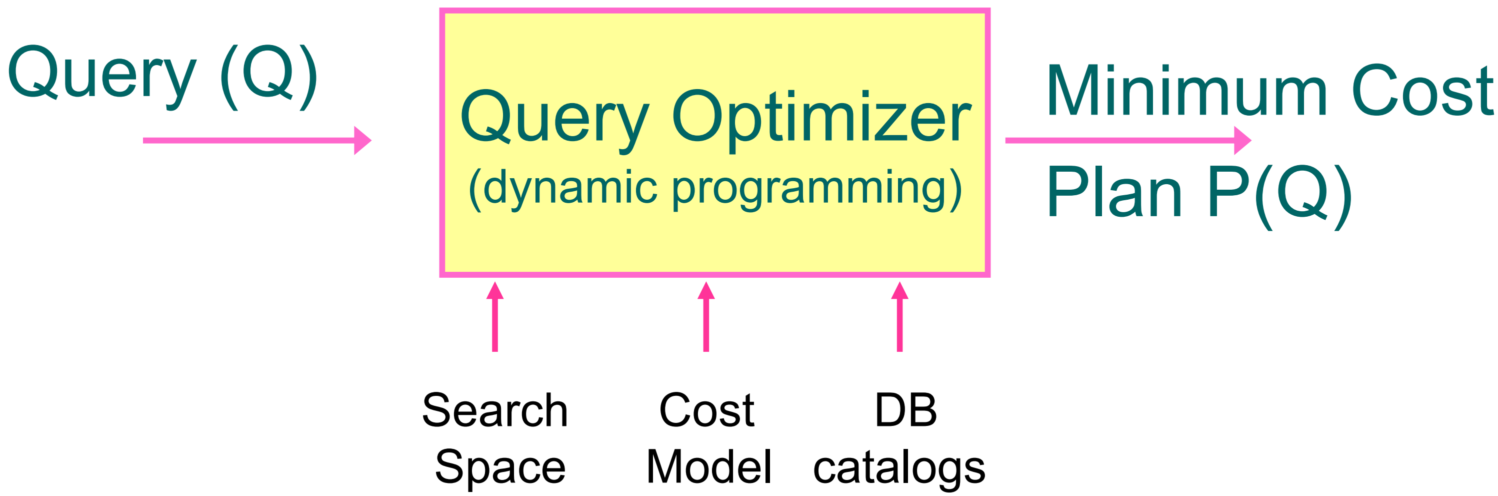
- DBMS query optimizer identifies efficient execution strategy: "query execution plan"

Sample Execution Plan



Query Plan Selection

- Core technique



- Computationally expensive since exhaustive search of exponential search space
(e.g. *n*-way table join ⇒ *n!* permutations)

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 (exponential) search space

Relation Selectivity

- An optimizer's choice of execution plan for a query is dependent on a large number of factors (database, system, query, etc.)

- 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

Query Template [Q7 of TPC-H]

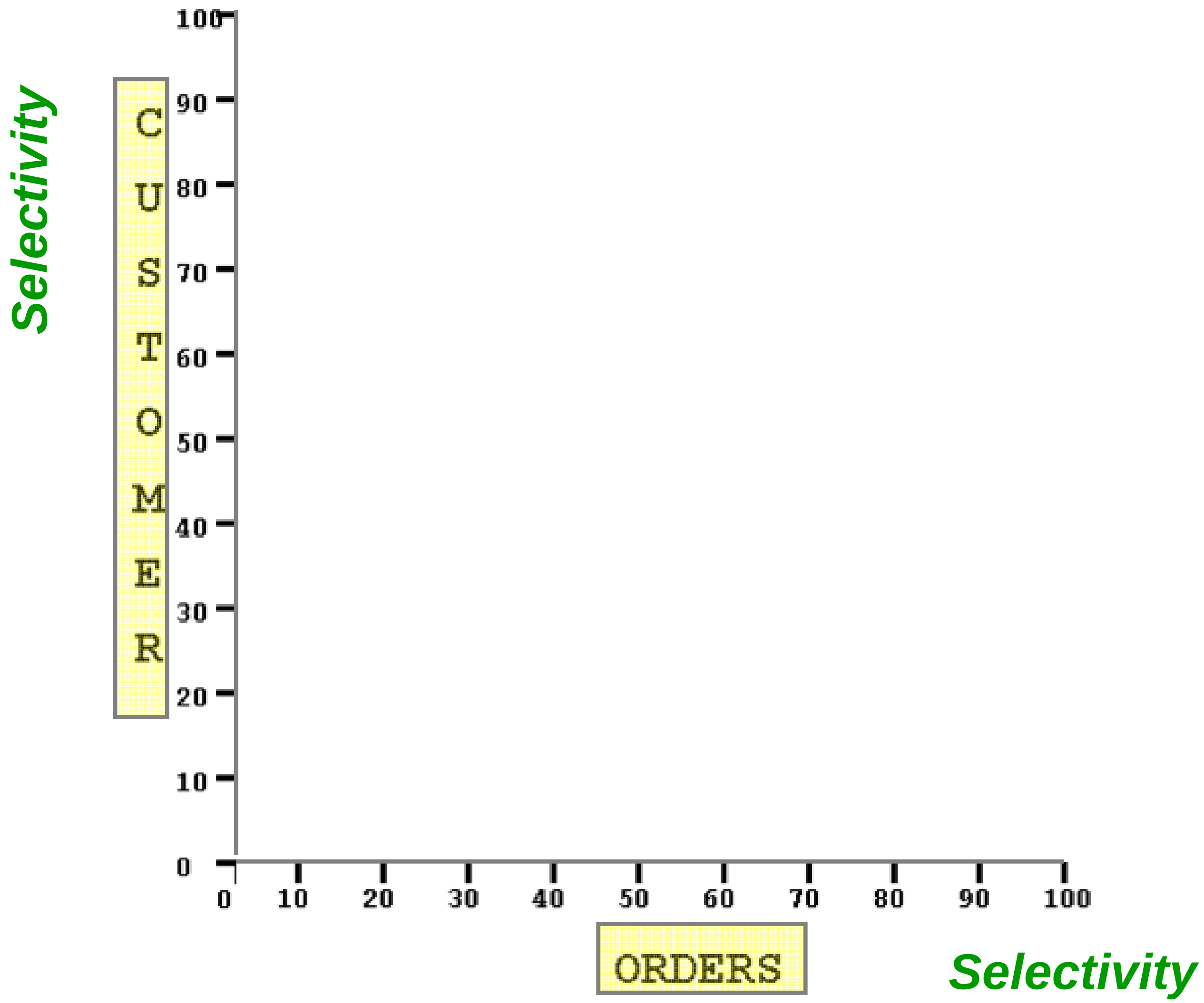
Determines the values of goods shipped between nations in a time period

```
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

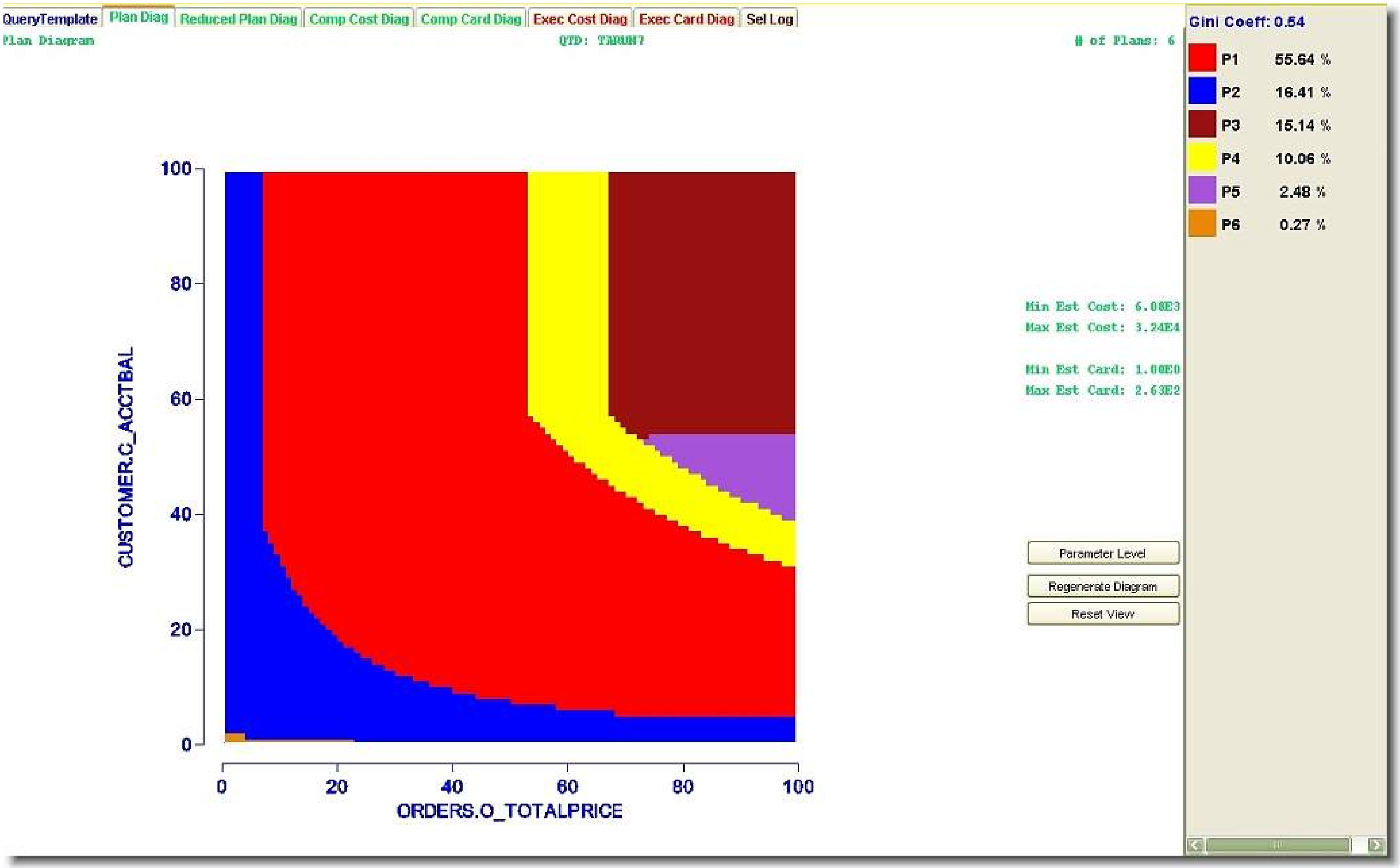
Relational Selectivity Space



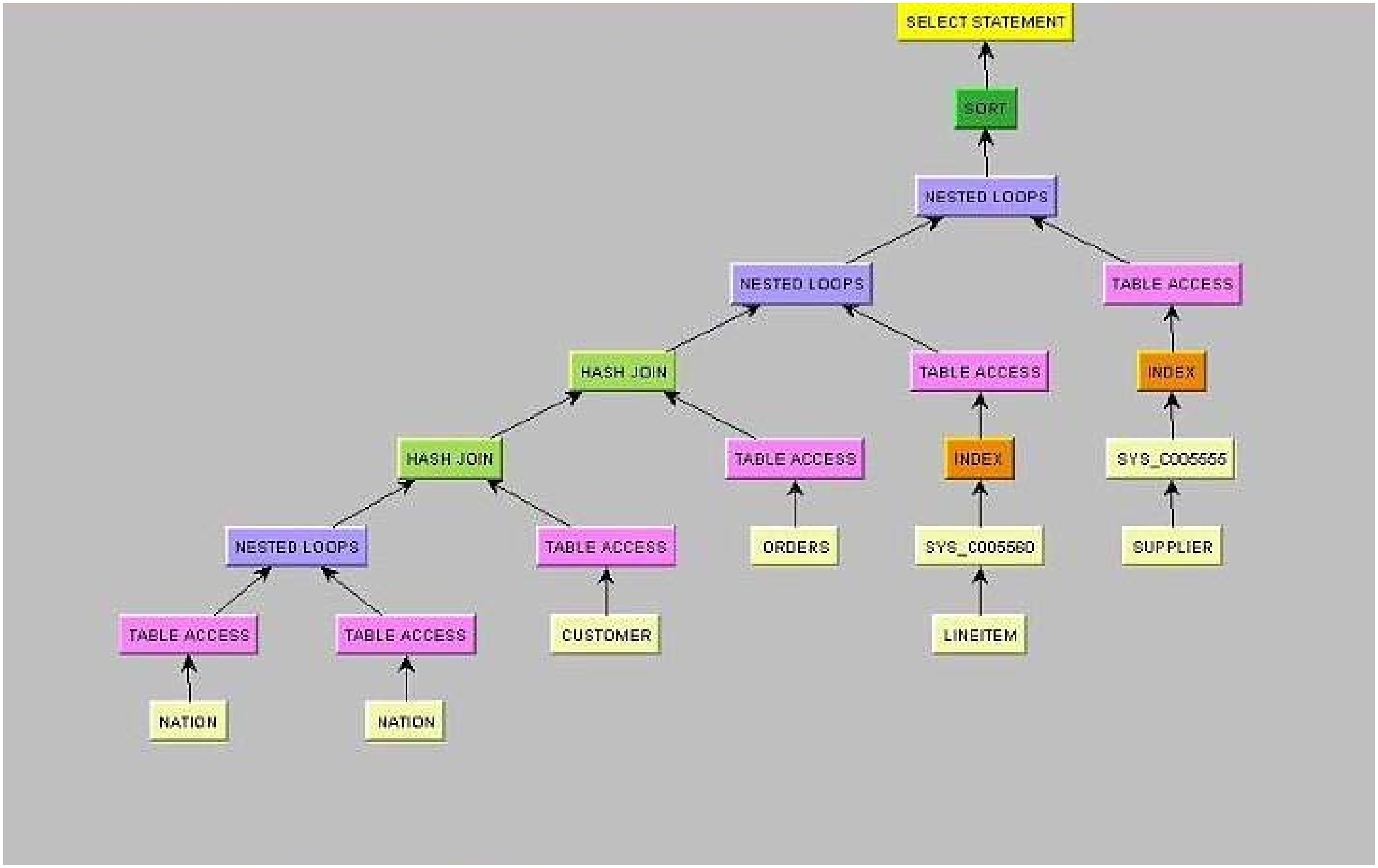
Plan, Cost and Card Diagrams

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

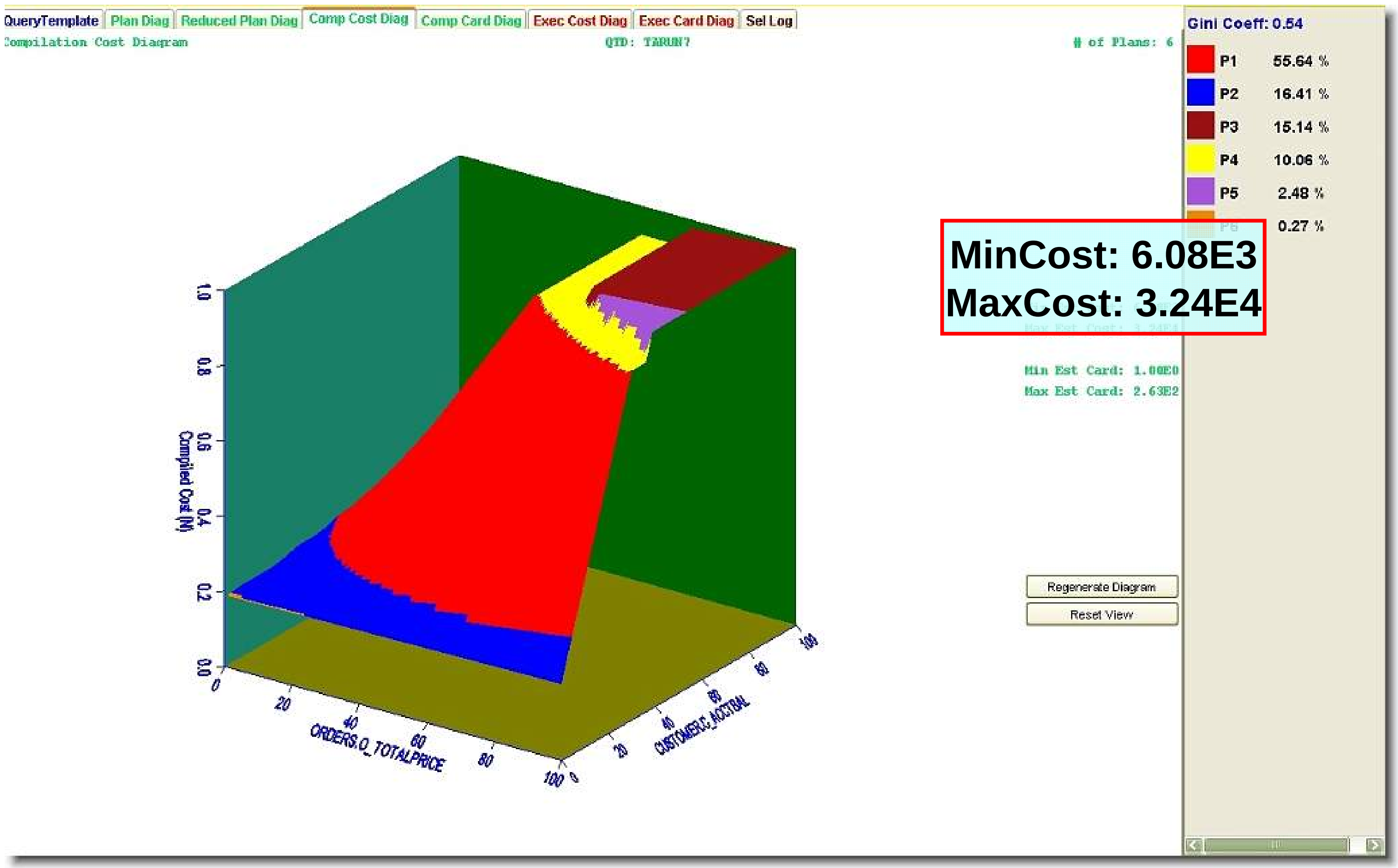
Sample Plan Diagram
[QT7,OptB]



Plan P5

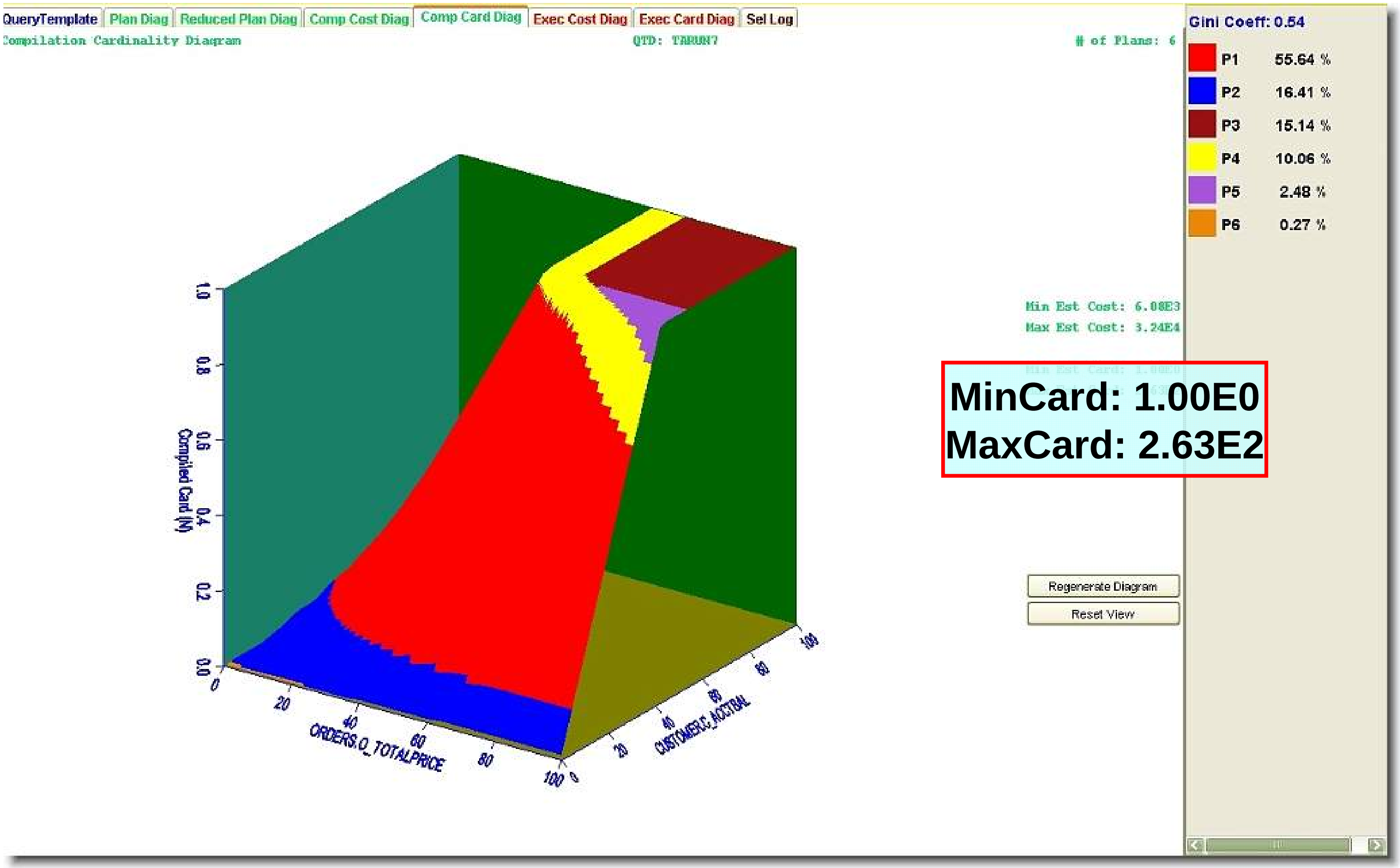

SELECT STATEMENT
SORT
NESTED LOOPS
NESTED LOOPS
TABLE ACCESS
HASH JOIN
TABLE ACCESS
INDEX
HASH JOIN
TABLE ACCESS
INDEX
SYS_C005555
NESTED LOOPS
TABLE ACCESS
ORDERS
SYS_C005560
SUPPLIER
TABLE ACCESS
TABLE ACCESS
CUSTOMER
LINEITEM
NATION
NATION

Sample Cost Diagram
[QT7,OptB]



Sample Cardinality Diagram

[QT7,OptB]



PICASSO

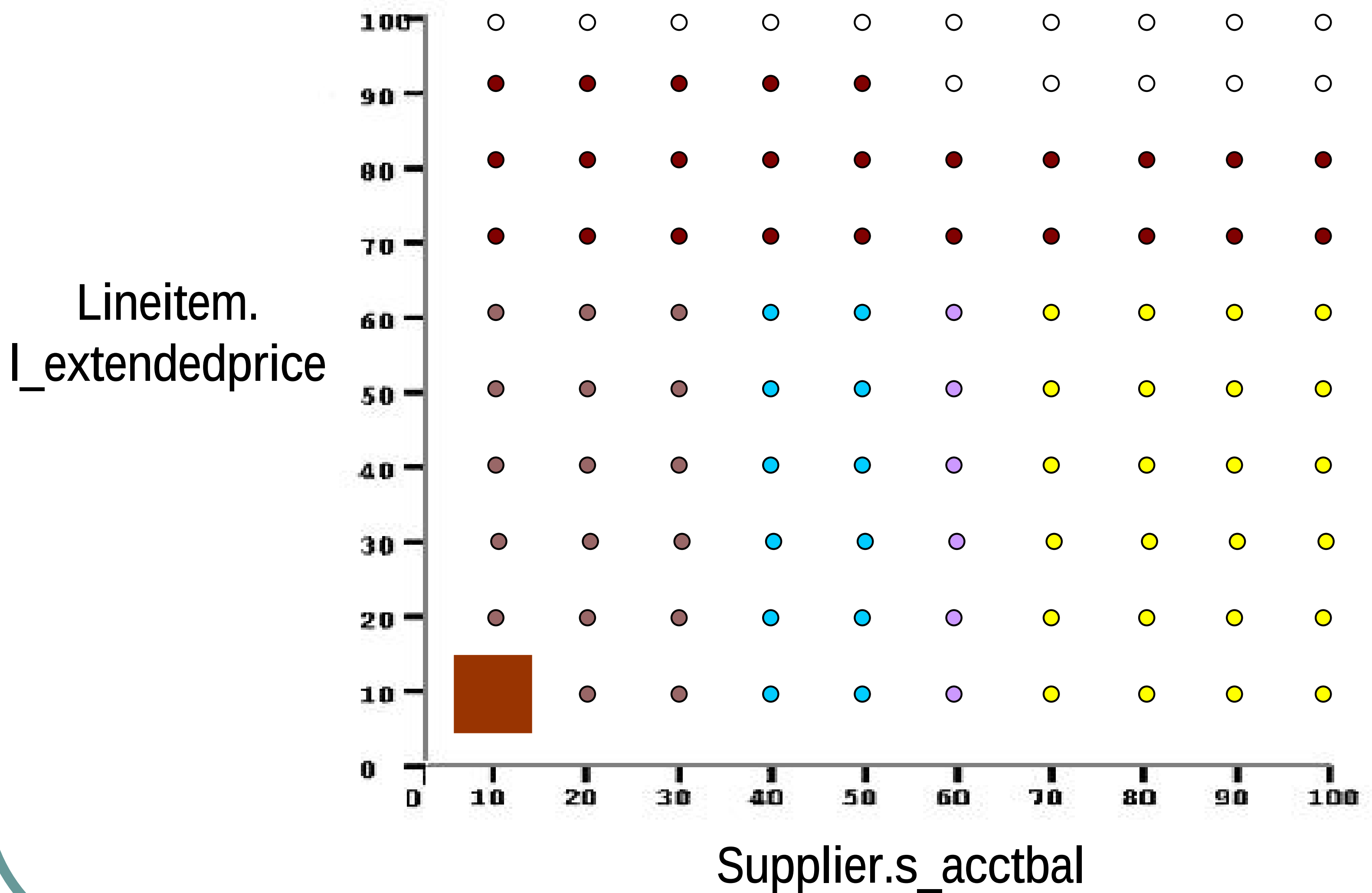
Overview

Picasso is a Java tool that, given an *n*-dimensional SQL query template and a choice of database engine, automatically generates plan, cost and card diagrams

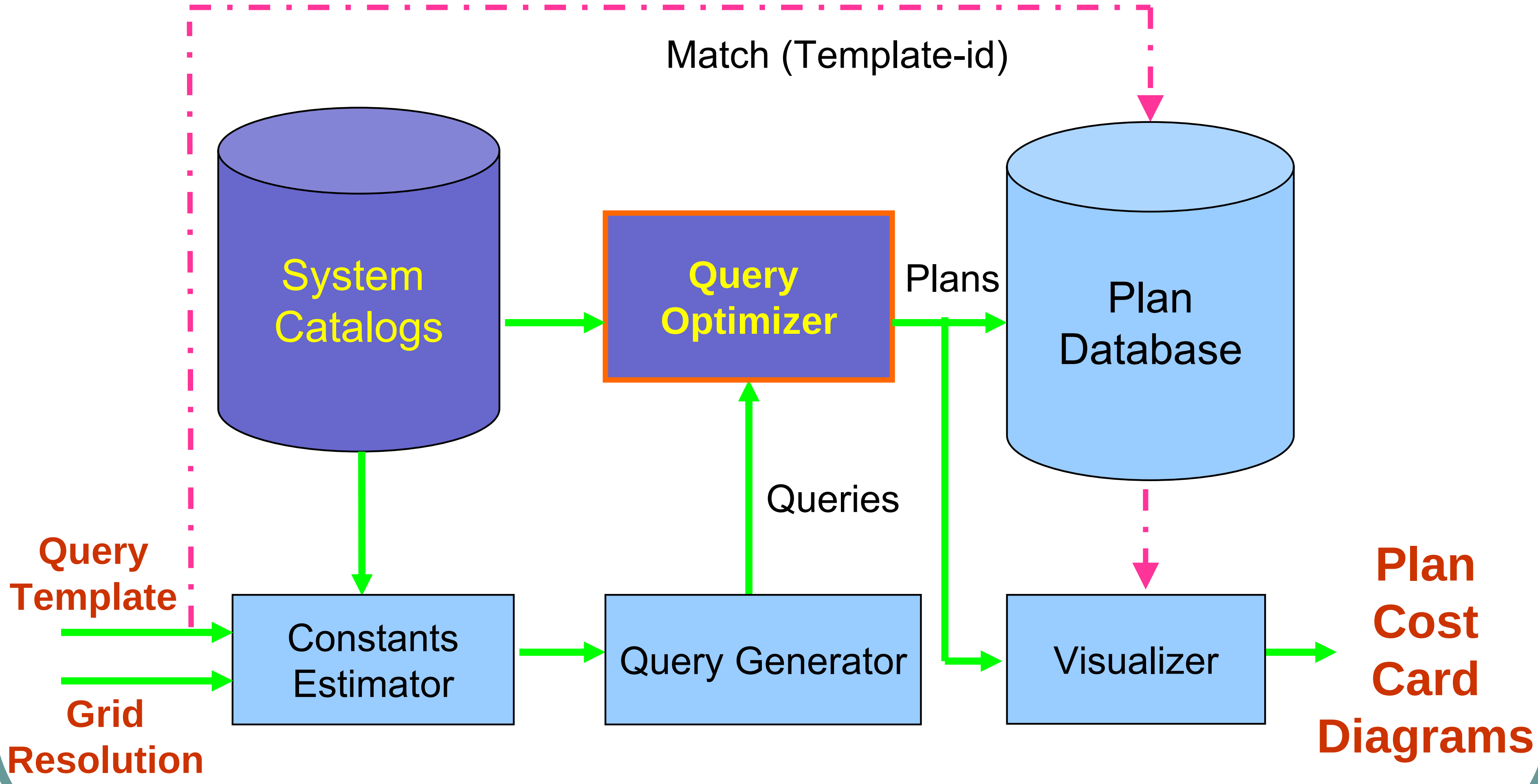
- Fires queries at user-specified granularity (10, 30, 100, 300, 1000 queries per dimension)
- Visualization: 2-D plan diagrams (slices if n > 2)
  3-D cost and card diagrams

Also:
Plan-trees, Plan differences
Foreign Plans
Abstract-plan diagrams
Execution cost/card diagrams

Diagram Generation Process



Picasso Architecture



PICASSO OUTPUT

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

Testbed Environment

- Databases
  - TPC-H database (1 GB)
  - TPC-DS database (100 GB)
- Query Sets
  - 2-D, 3-D, 4-D Query templates based on TPC-H benchmark [Q1 ~ Q22] and TPC-DS benchmark [Q1 ~ Q99]
  - Default uniform 100x100 grid (10000 queries) [0.5%, 0.5%] to [99.5%, 99.5%]

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

- Relational Engines
  - Default installations (with all optimization features on)
  - Stats on all columns; no extra indices
- Computational Platforms
  - PIV 2.4 GHz, 2GB RAM, Windows XP Pro
  - Sun Opteron 4GHz, 4GB RAM, Windows XP Pro

The Picasso Connection

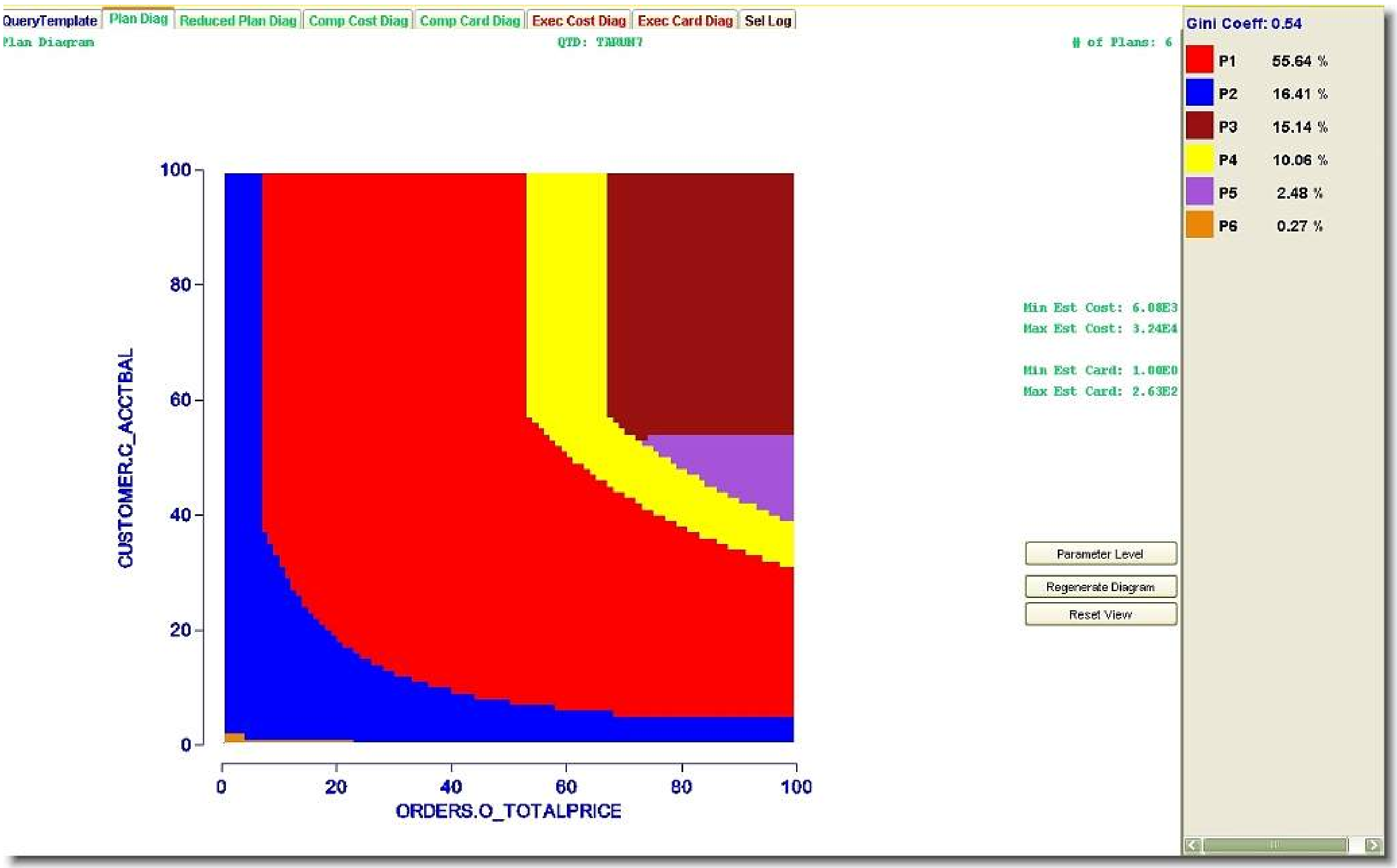
Woman with a guitar

Georges Braque, 1913

Plan diagrams are often similar to cubist paintings !

[Pablo Picasso – founder of cubist genre]

Smooth Plan Diagram
[QT7,OptB]

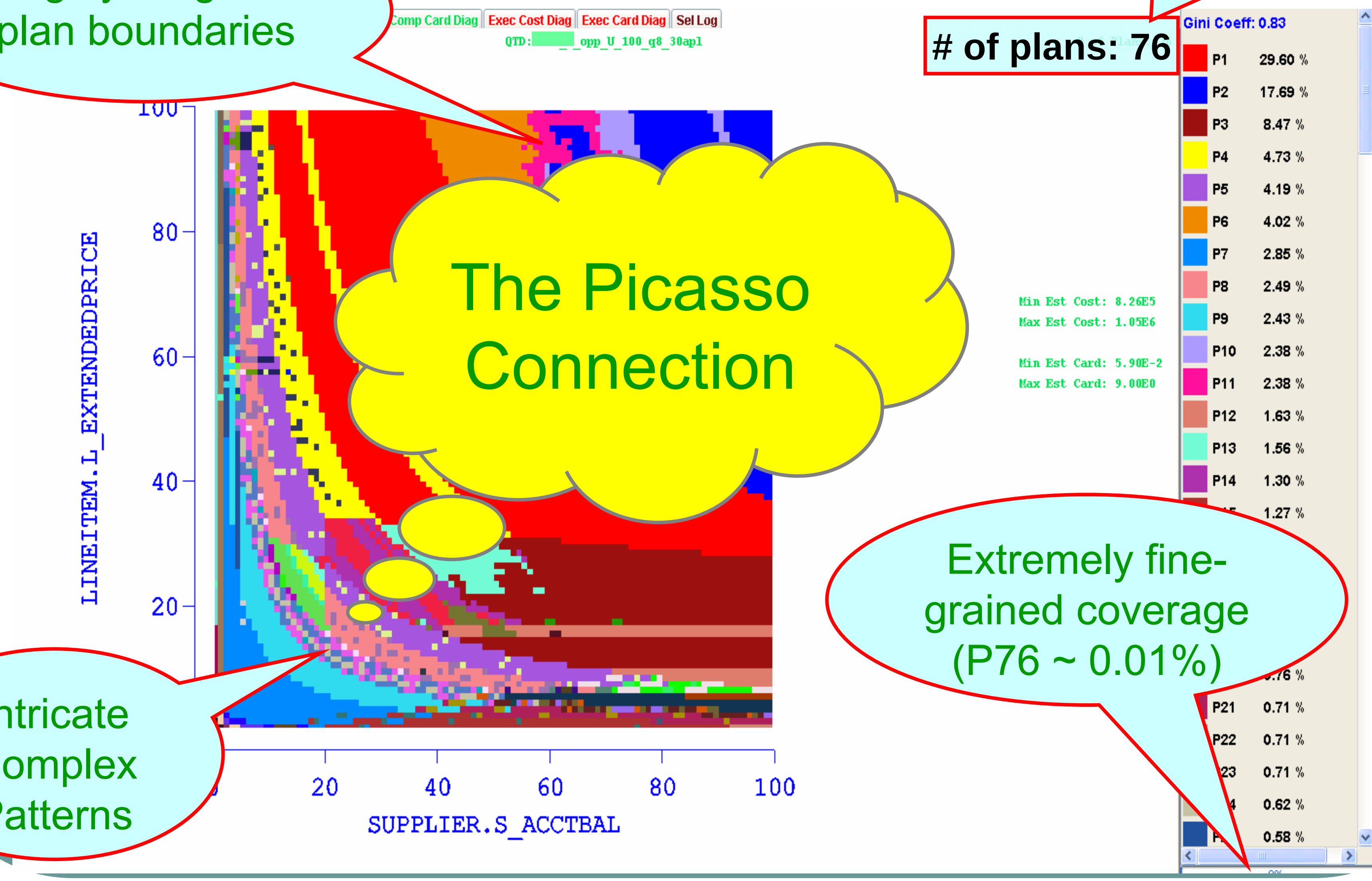


Complex Plan Diagram

[QT8.OptA*]

Highly irregular plan boundaries

Increases to **90 plans** with 300x300 grid !



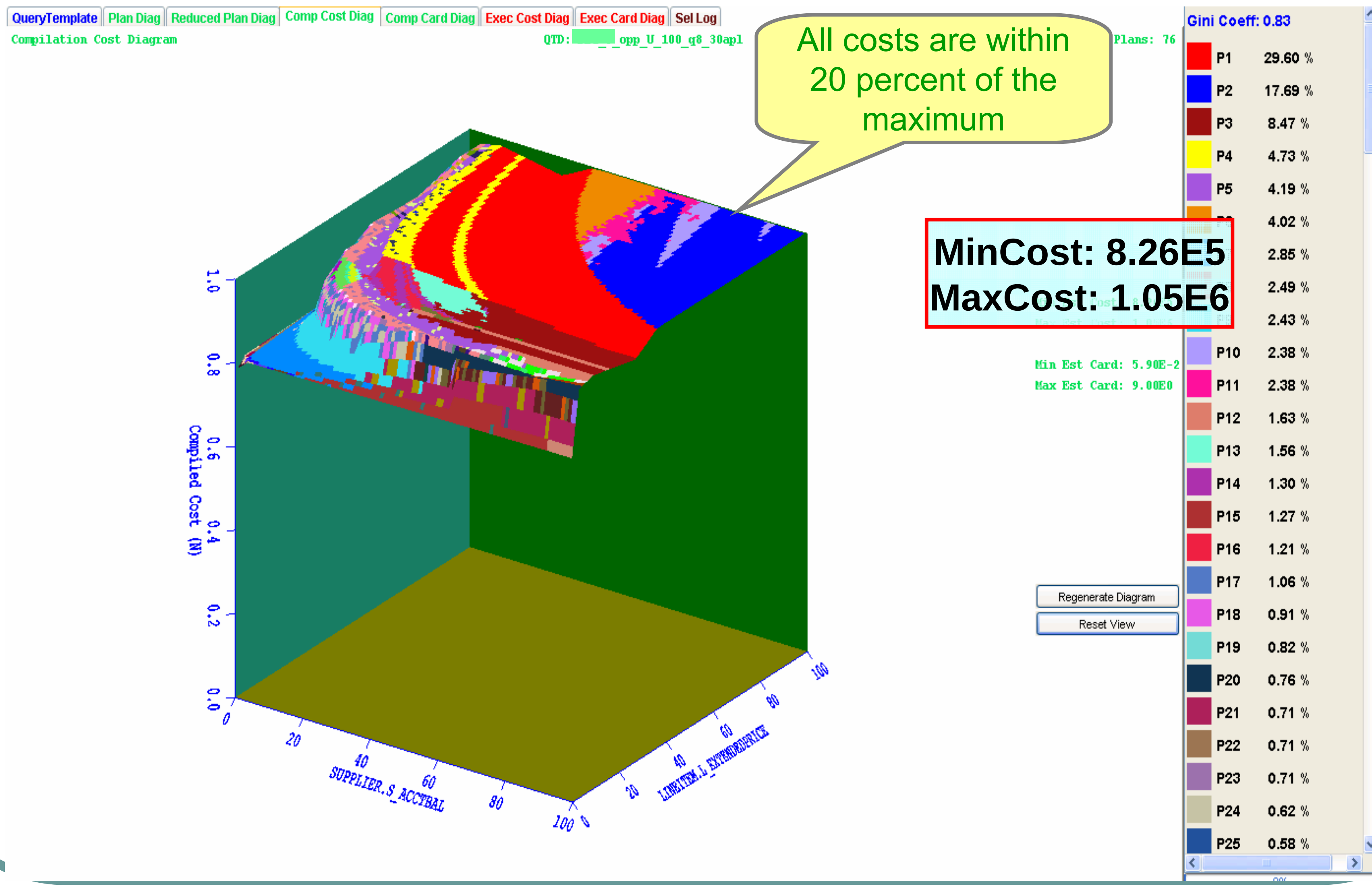
The Picasso Connection

Intricate Complex Patterns

Extremely fine-grained coverage (P76 ~ 0.01%)

Cost Diagram

[QT8, Opt A*]



Remarks

- Modern optimizers tend to make extremely fine-grained and skewed choices
- 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?

PLAN DIAGRAM REDUCTION

Problem Statement

Can the plan diagram be recolored with a smaller set of colors (i.e. some plans are "swallowed" by others), such that

Guarantee:

No query point in the original diagram has its estimated cost increased, post-swallowing, by more than λ percent (user-defined)

Analogy:

(with due apologies to Sri Lankans in the audience)

Sri Lanka agrees to be annexed by India if it is assured that the cost of living of each Lankan citizen is not increased by more than λ percent

Our Results

- Optimal plan diagram reduction (w.r.t. minimizing the number of plans/colors) is NP-hard
  - through problem-reduction from classical Set Cover

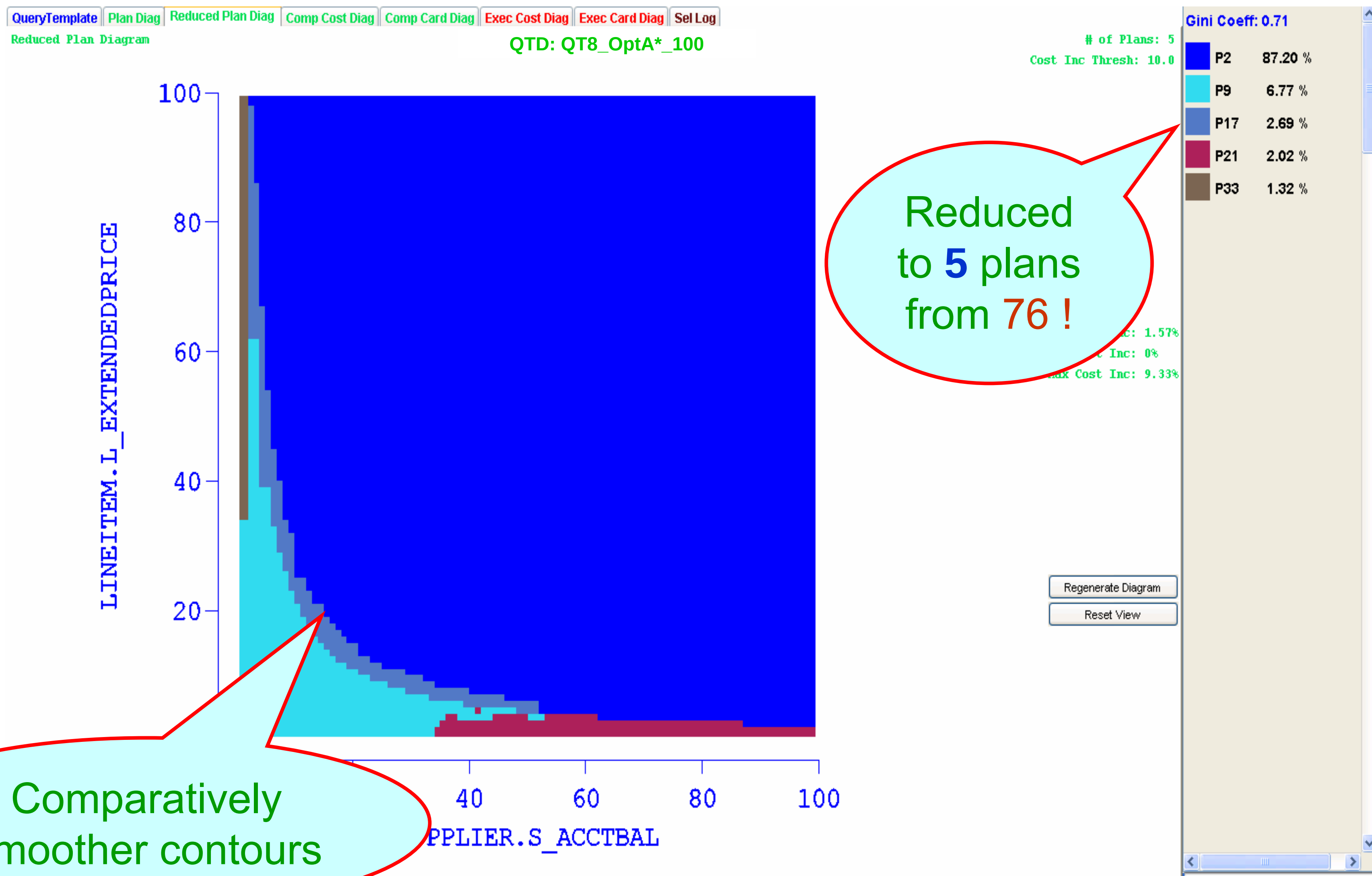
- Designed CostGreedy, a greedy heuristic-based algorithm with following properties:

  [m is number of query points, n is number of plans in diagram]

  - Time complexity is O(mn)
    - linear in number of plans for a given diagram resolution
  - Approximation Factor is O(ln m)
    - bound is both tight and optimal
    - in practice, closely approximates optimal

Reduced Plan Diagram [λ=10%]

[QT8, OptA*, Res=100]



Reduced to **5** plans from 76 !

Comparatively smoother contours

Anorexic Reduction

Extensive empirical evaluation with a spectrum of multi-dimensional TPCH-based query templates indicates that

"With a cost-increase-threshold of just 20%, virtually all complex plan diagrams
[irrespective of query templates, data distribution, query distribution, system configurations, etc.]

reduce to "anorexic levels" (~10 or less plans)!

Applications of Plan Diagram Reduction

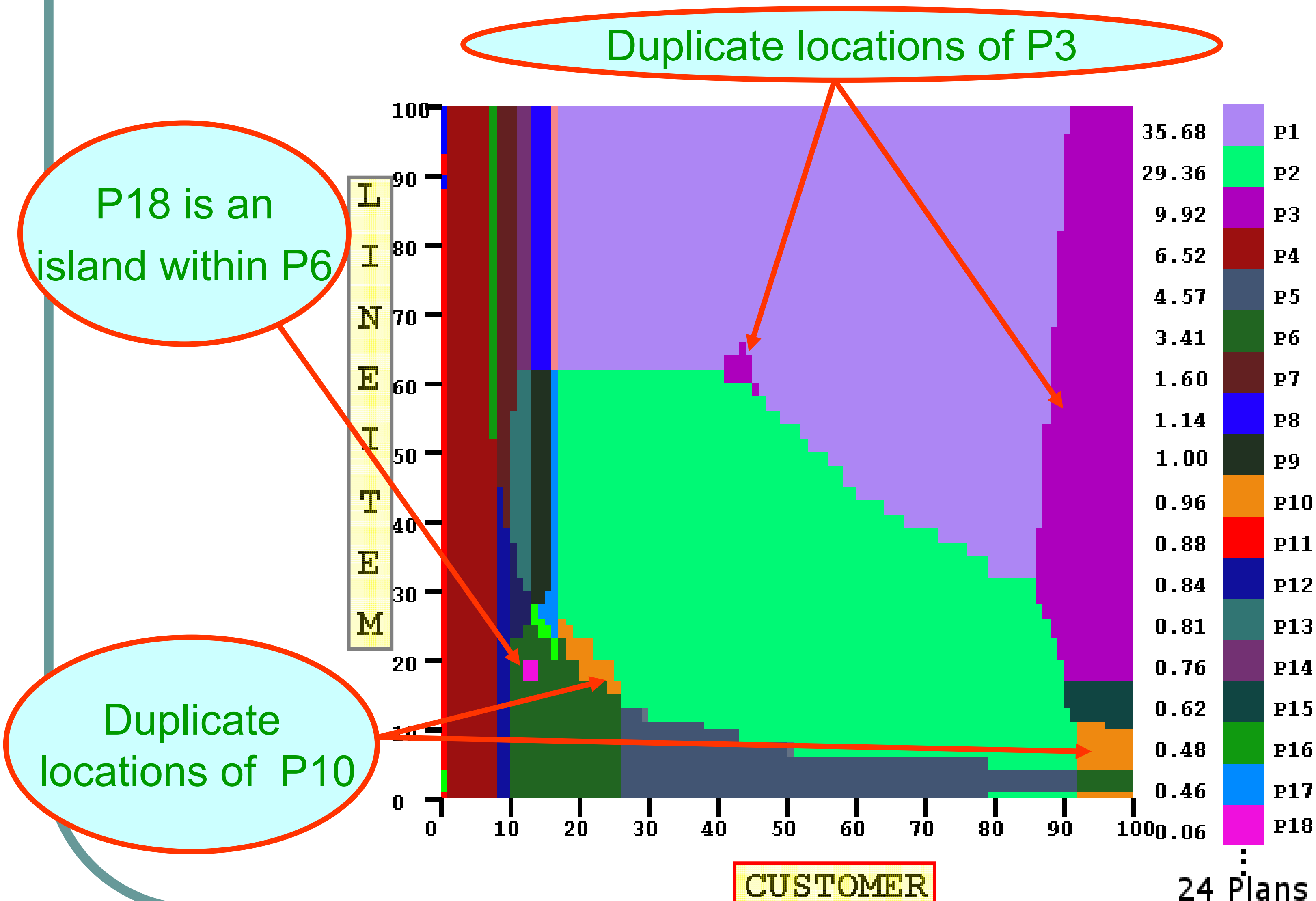
- Quantifies redundancy in plan search space
- Provides better candidates for plan-cacheing
- Enhances viability of Parametric Query Optimization (PQO) techniques
- Improves efficiency/quality of Least-Expected-Cost (LEC) plans
- Minimizes overheads of multi-plan (e.g. Adaptive Query Processing) approaches
- Identifies selectivity-error resistant plan choices
  - retained plans are robust choices over larger selectivity parameter space

Picasso Art Gallery

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

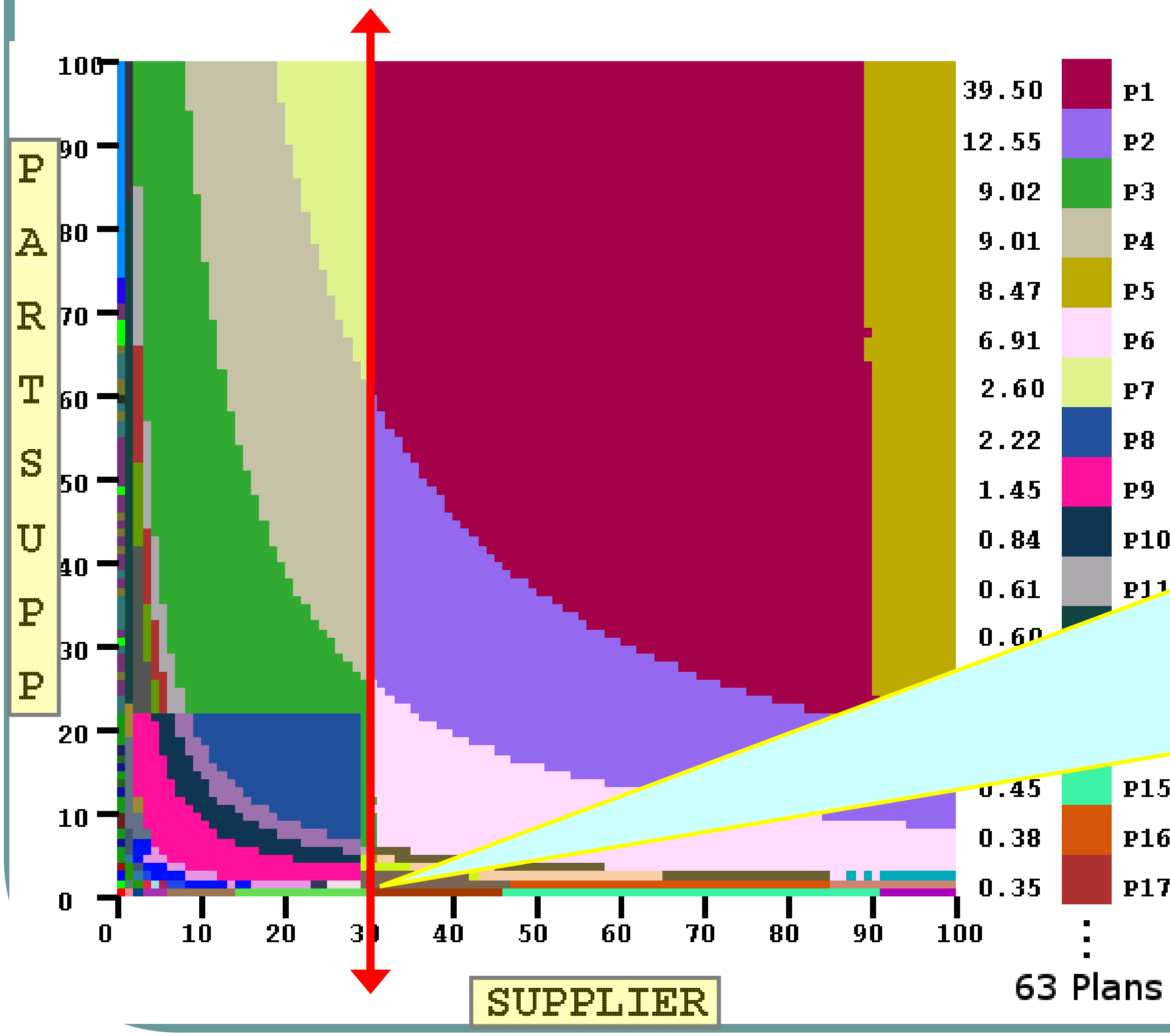
Duplicates and Islands

[QT10, OptA]



Plan Switch Points

[QT9,OptA]

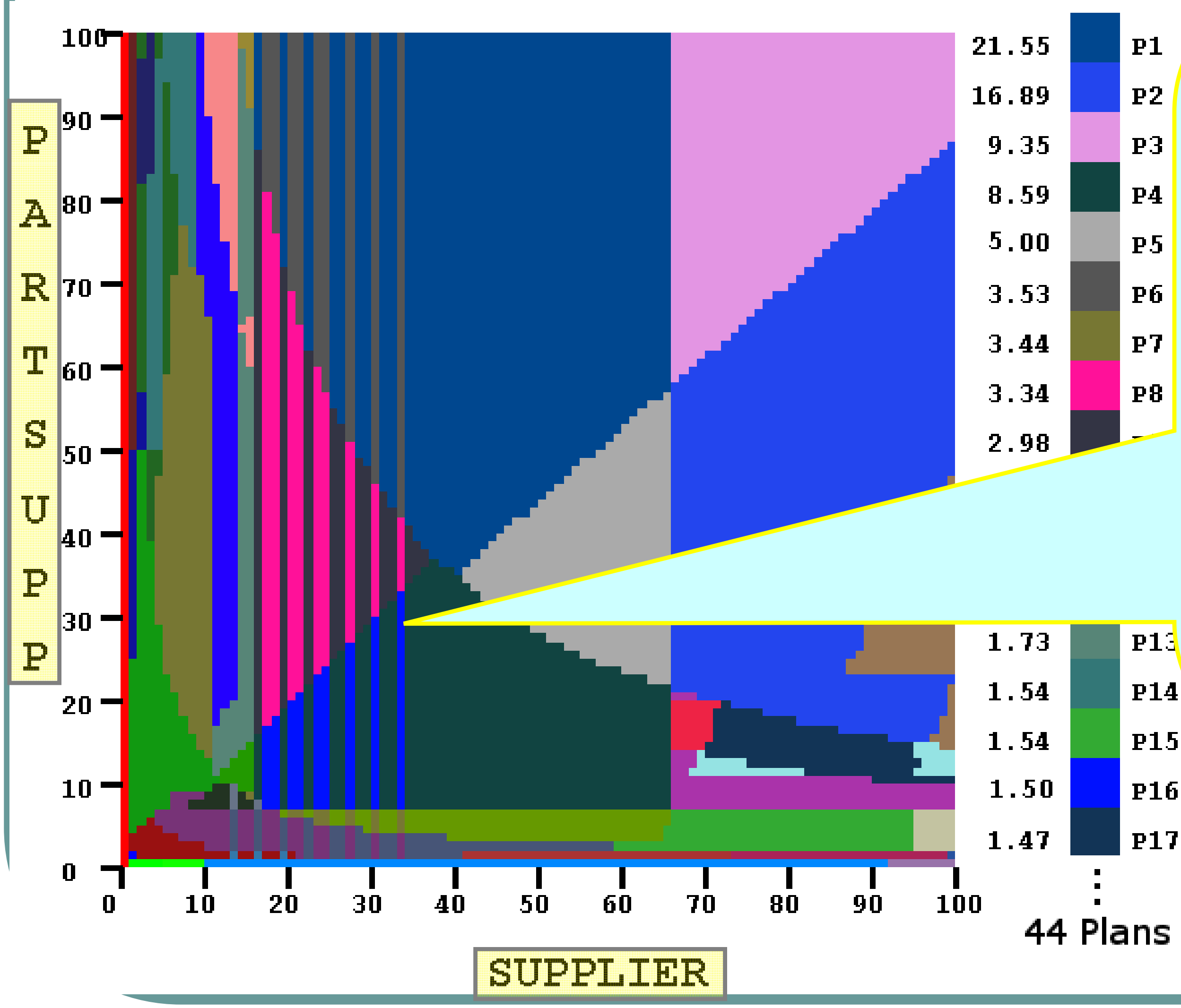


Plan Switch Point:
line parallel to axis with a plan shift for all plans bordering the line.

Hash-Join sequence
PARTSUPP►◄SUPPLIER►◄PART
is altered to
PARTSUPP►◄PART►◄SUPPLIER

Venetian Blinds

[QT9,OptB]

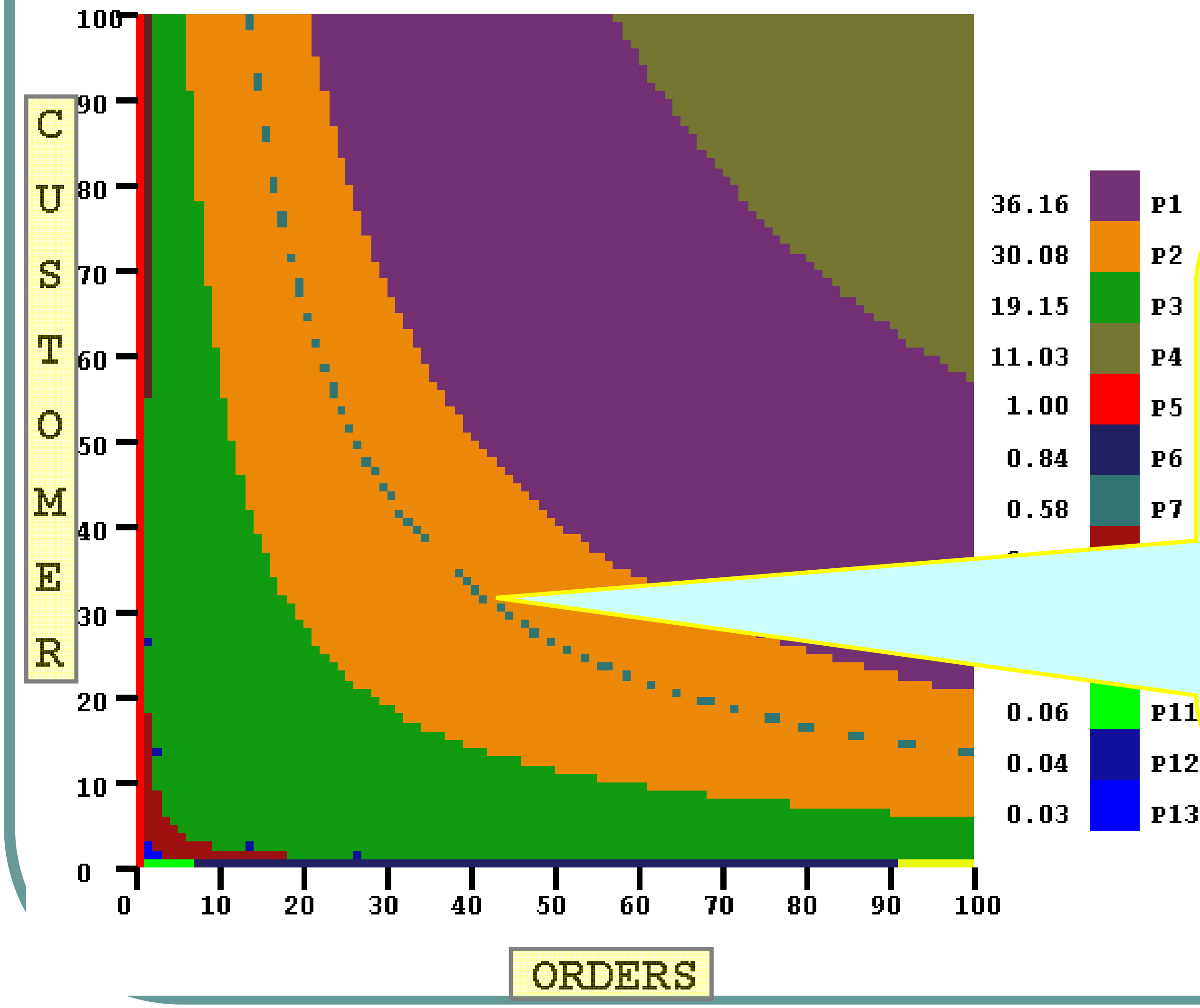


Six plans simultaneously change with rapid alternations to produce a “Venetian blinds” effect.

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

Footprint Pattern

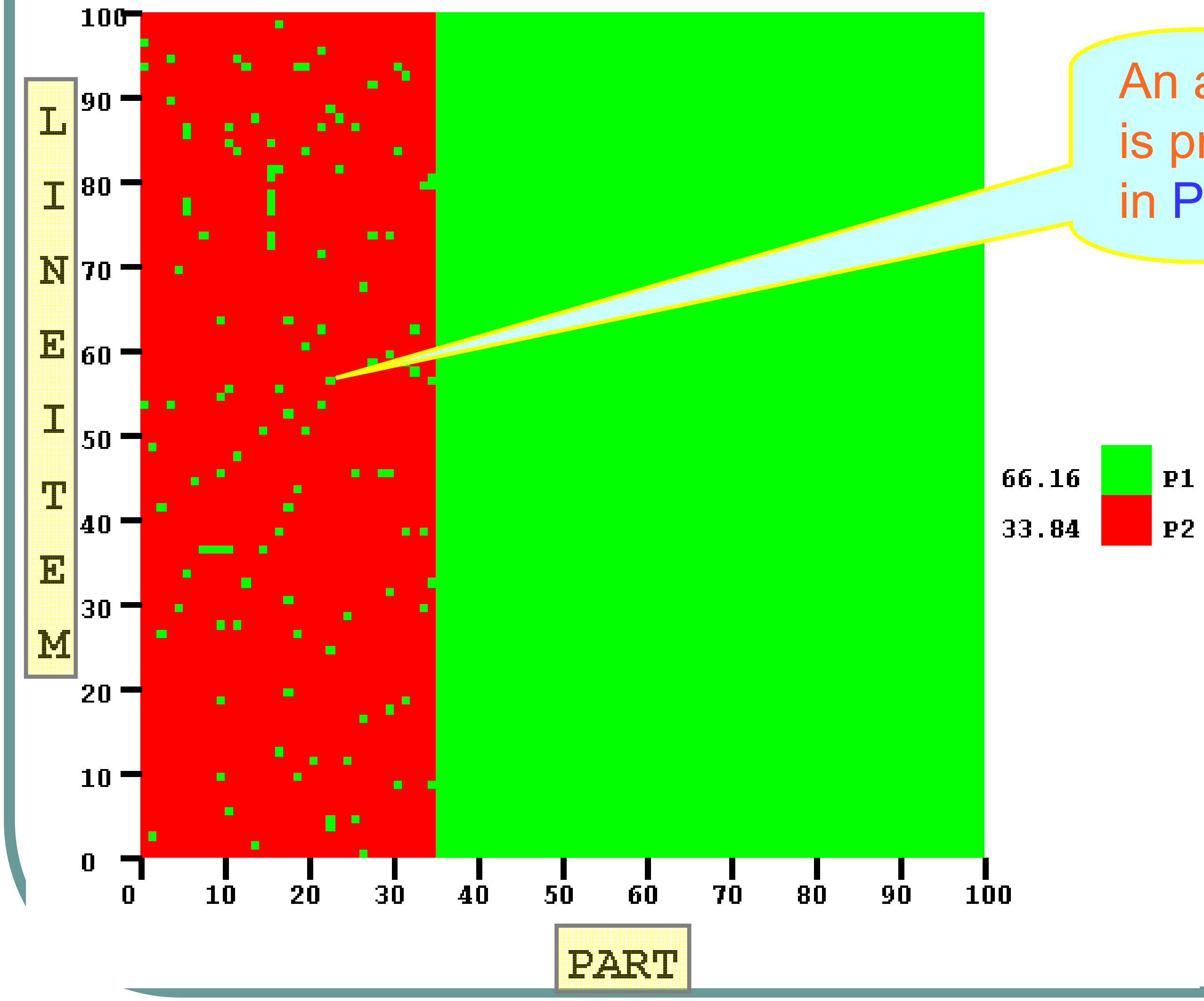
[QT7,OptA]



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
[QT17,OptA]

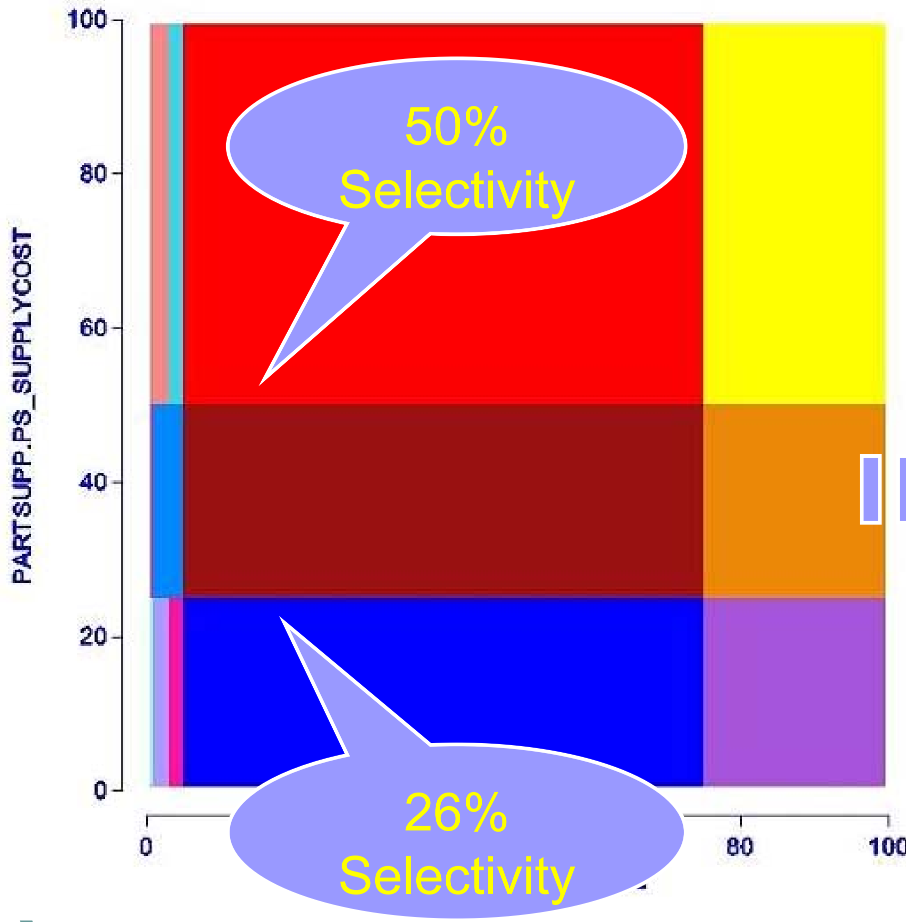


An additional sort operation is present on PART relation in P2, whose cost is very low

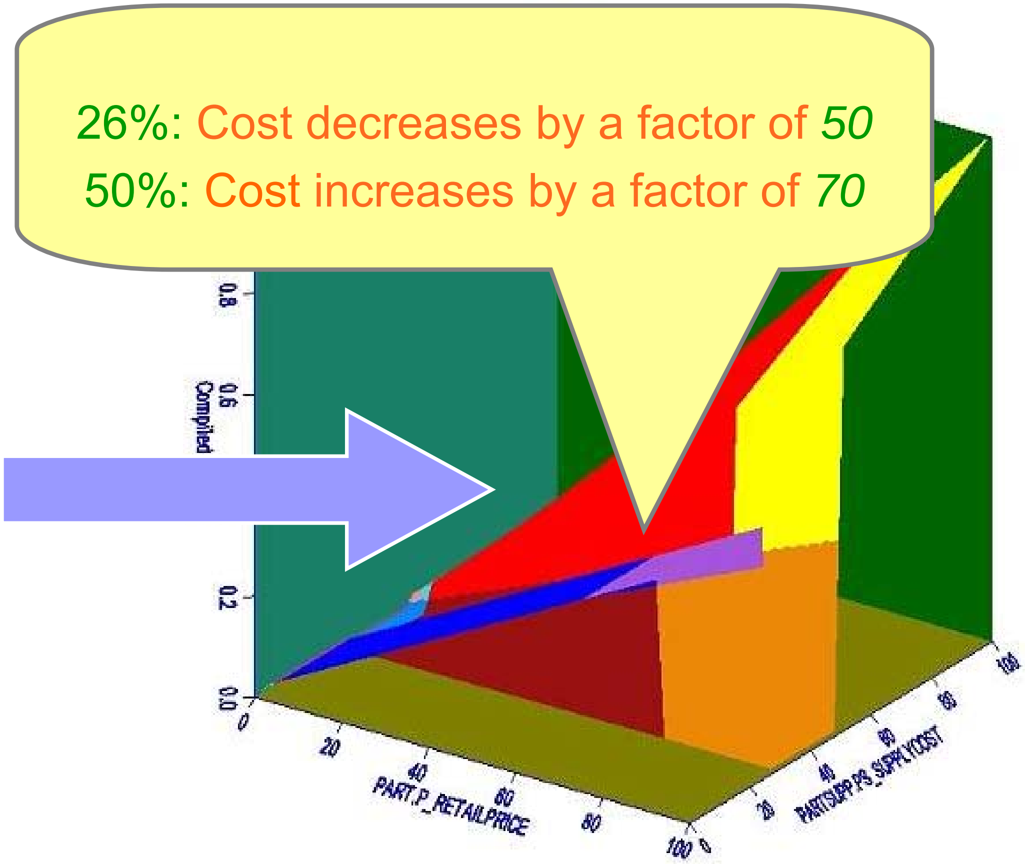
Non-Monotonic Cost Behavior

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

Plan-Switch Non-Monotonic Costs
[QT2,OptA]



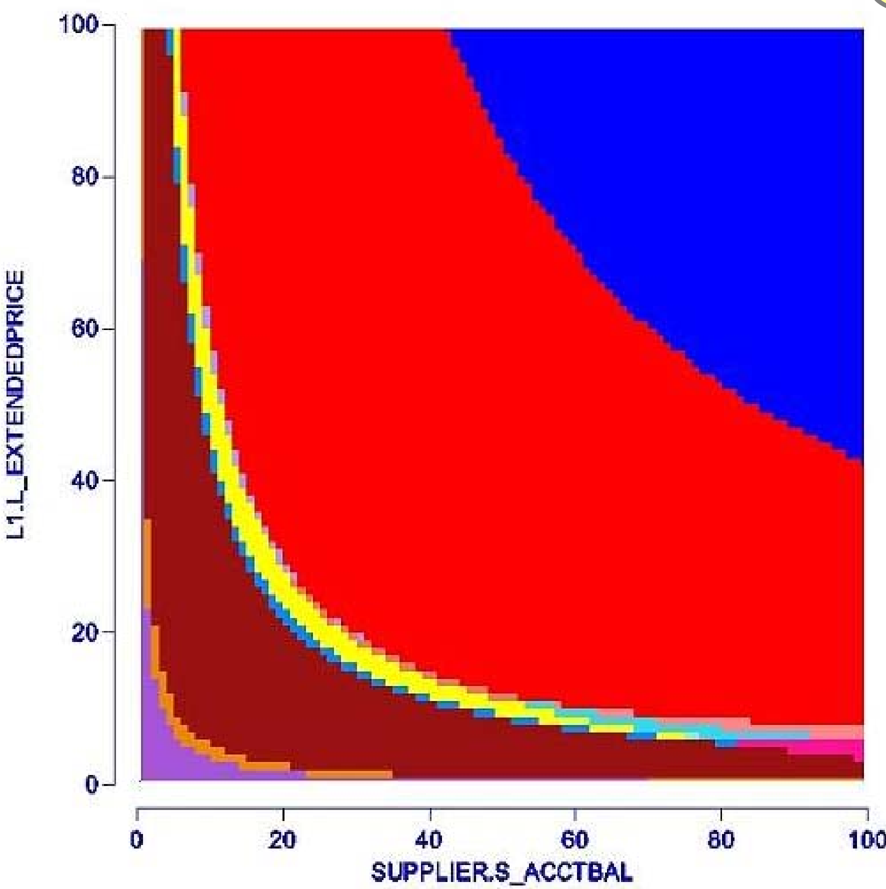
Plan Diagram



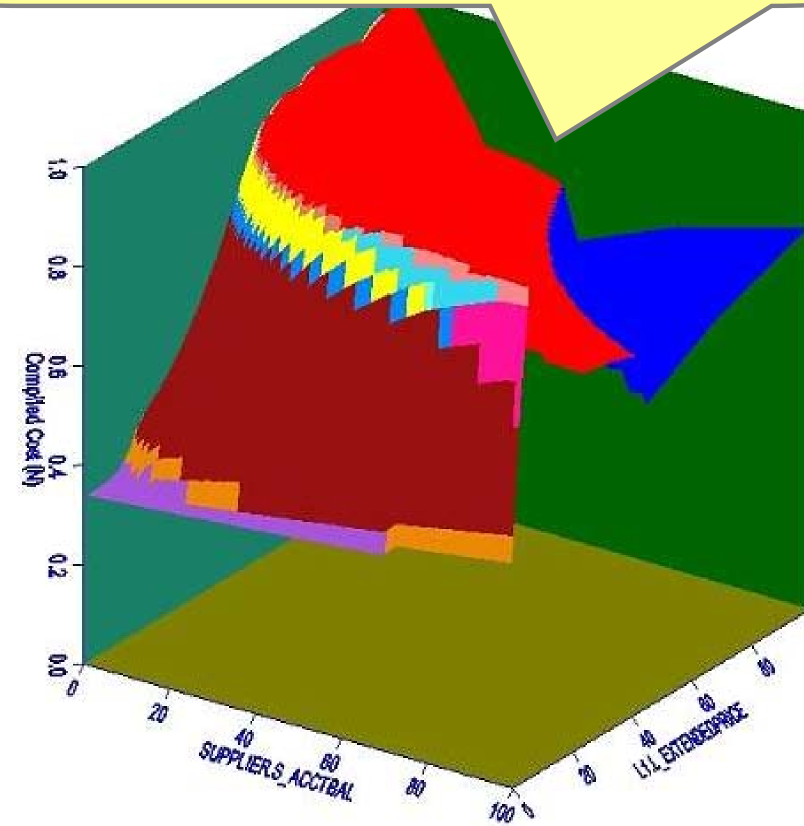
Cost Diagram

Intra-Plan Non-Monotonic Costs
[QT21,OptA]

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 …

CONCLUSIONS

Picasso Visualizer

- Conceived and developed the **Picasso tool** for automatically generating plan, cost and card diagrams
  - optimizer debugger / research platform / teaching aid
- Analyzed representative plan diagrams on popular commercial query optimizers
  - Optimizers make *fine grained* choices
  - Plan optimality regions can have *intricate patterns* and *complex boundaries*
  - *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

Plan Diagram Reduction

While the optimization process is sensitive to many parameters, including query structure, data distribution, system resources, etc., the reduction process is largely indifferent to these factors and most complex plan diagrams can be reduced to a "few good plans".

This result could have useful implications for the design and use of next-generation database query optimizers, especially w.r.t. to plan cacheing, parametric query optimization, selectivity-error resistance, adaptive query processing, etc.

More Details:

http://dsl.serc.iisc.ernet.in/projects/PICASSO

Publications, Software, Sample Diagrams

END PRESENTATION