Plan Bouquets:

Query Processing without Selectivity Estimation

E0 261

Jayant Haritsa
Computer Science and Automation
Indian Institute of Science

Talk Theme

Declarative query processing with performance guarantees has been a highly desirable but equally elusive goal for the database community over the last three decades. Here, we present a conceptually new approach, called "plan bouquets", to address this classical problem.

[PhD Dissertations:
Anshuman Dutt, Microsoft Research Redmond
Srinivas Karthik, EPFL Switzerland]

Canonical Query Optimization Framework

Declarative Query (Q) → **Query Optimizer** → *Optimal Plan P(Q)* [Min Cost]

Operator Execution Cost Estimation Model

function of (Hardware, DB Engine)

e.g. NL Join = $|R_{outer}| + |R_{outer}| \times |R_{inner}|$

Operator Result Cardinality Estimation Model

function of (Data Distributions, Data Correlations)

e.g. Output Cardinality of Join = $|t_{outer}| \times |t_{inner}| \times$ join filter factor

Run-time Sub-optimality

The supposedly optimal plan-choice from the query optimizer may actually turn out to be highly sub-optimal when the query is executed with this plan. This adverse effect is due to errors in:

(a) cost model → limited impact, < 30 %

(b) cardinality model → huge impact, orders of magnitude

- Coarse statistics, attribute value independence (AVI) assumption, multiplicative error propagation, outdated statistics, query construction, …

Proof by Authority [Guy Lohman, IBM]

Snippet from April 2014 Sigmod blog post on

"Is Query Optimization a "Solved" Problem?"

The root of all evil, the Achilles Heel of query optimization, is the estimation of the size of intermediate results, known as cardinalities. The cardinality model can easily introduce errors of many orders of magnitude! With such errors, the wonder isn't "Why did the optimizer pick a bad plan?" Rather, the wonder is **"Why would the optimizer ever pick a decent plan?"**

Prior Research (lots!)

- Sophisticated estimation techniques
  - SIGMOD 1999, VLDB 2001, VLDB 2009, SIGMOD 2010, VLDB 2011, …
  - e.g. wavelet histograms, self-tuning histograms, learning histograms
- Selection of Robust Plans
  - SIGMOD 1994, PODS 2002, SIGMOD 2005, VLDB 2008, SIGMOD 2010, …
  - e.g. variance-aware plan selection, LEC (least expected cost)
- Runtime re-optimization techniques
  - SIGMOD 1998, SIGMOD 2000, SIGMOD 2004, SIGMOD 2005, …
  - e.g. POP (progressive optimization), RIO (Re-optimizer)

VVS LAXMAN

RAHUL DRAVID

M S DHONI

Several novel ideas and formulations, but lack performance guarantees

Talk Summary

- We present "plan bouquets", a query processing technique that completely eschews making estimates for error-prone cardinalities

- Plan Bouquet Approach: run-time discovery of selectivities using a compile-time selected bouquet of plans
  - provides worst case performance guarantees wrt omniscient oracle that knows the correct selectivities
    - e.g. for a single error-prone selectivity, relative guarantee of 4
  - empirical performance well within guaranteed bounds on industrial-strength environments

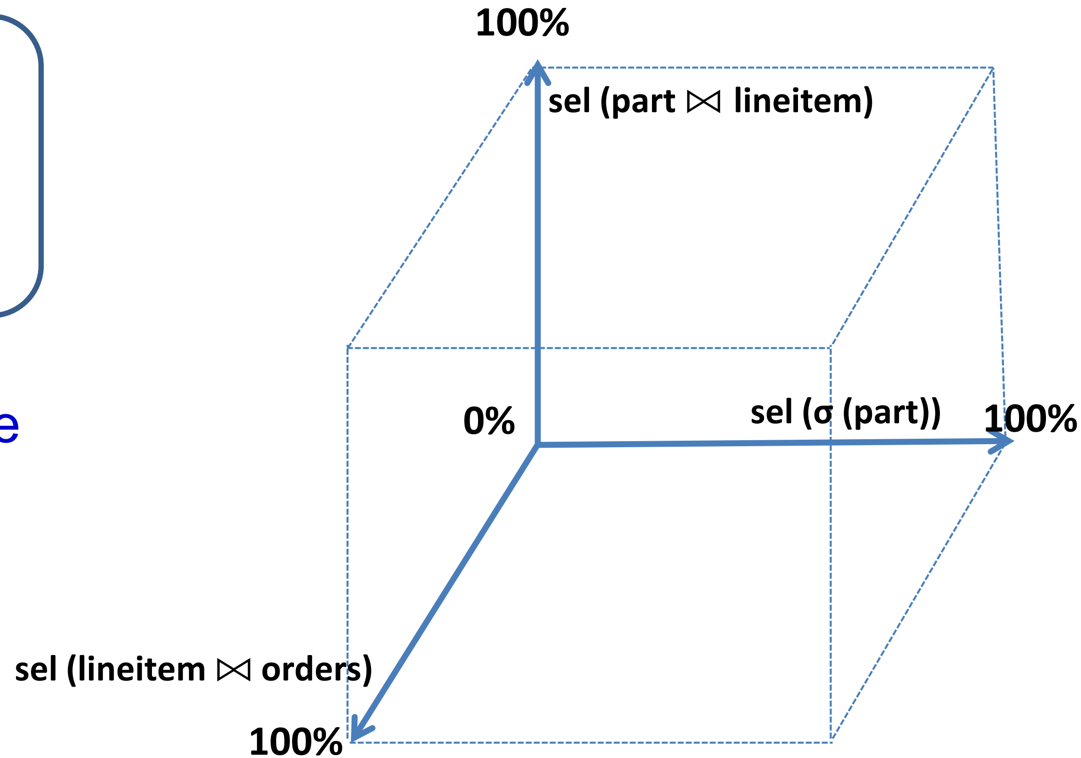
Problem Framework

Selectivity Dimensions

Example Query: EQ

```
select   *
from     lineitem, orders, part
where    p_partkey = l_partkey and
         l_orderkey = o_orderkey and
         p_price < 1000
```

ESS – Error Selectivity Space

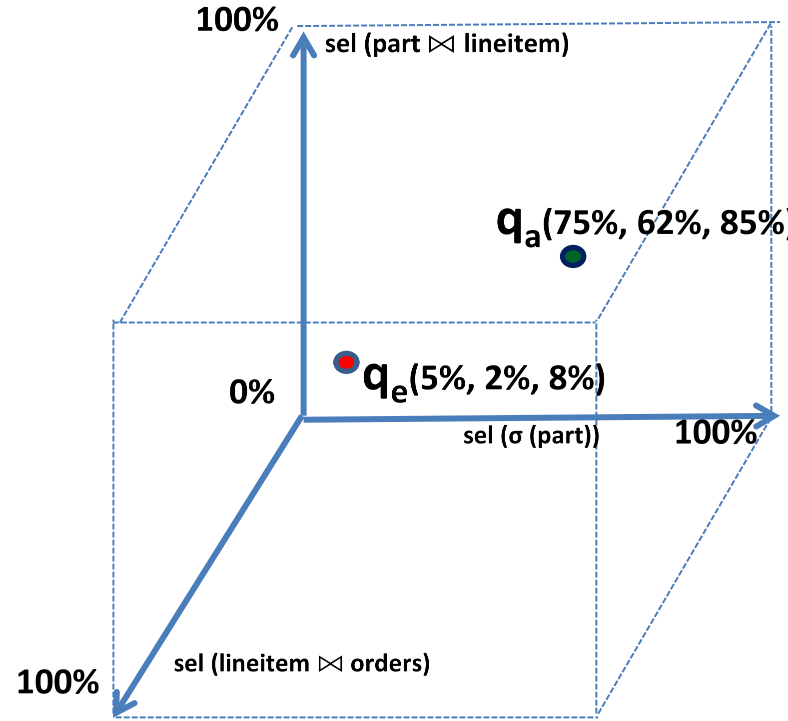


Performance Metrics

- q_e – estimated selectivity location in SS
- q_a – actual run-time location in SS

- P_{oe} – optimal plan for q_e
- P_{oa} – optimal plan for q_a

$$SubOpt(q_e, q_a) = \frac{cost(P_{oe}, q_a)}{cost(P_{oa}, q_a)} \qquad [1, \infty)$$



$$MaxSubOpt\ (MSO) = MAX[SubOpt(q_e, q_a)] \quad \forall q_e, q_a \in SS$$

$$AvgSubOpt\ (ASO) = AVG[SubOpt(q_e, q_a)] \quad \forall q_e, q_a \in SS$$

Main Assumptions

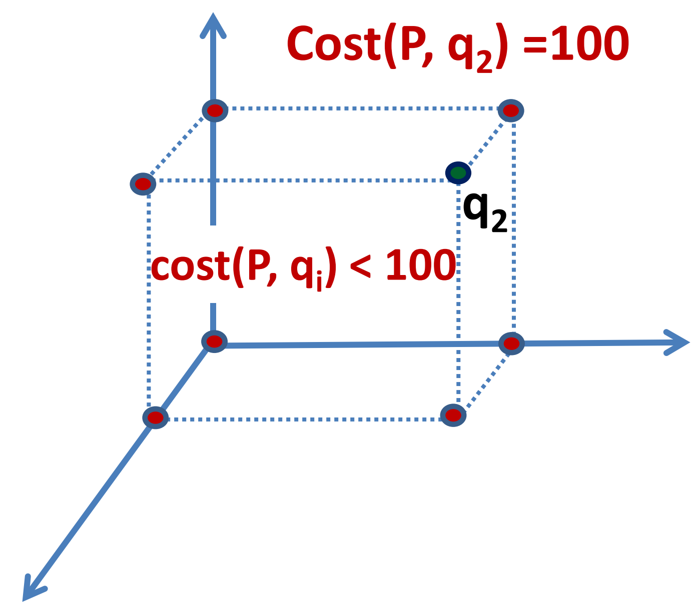
- Plan Cost Monotonicity (Mandatory)
- Perfect Cost Model (relaxed at end of talk)
- Independent SS Dimensions (open question)

PCM:

For any plan **P** and distinct selectivity locations $\mathbf{q_1}$ and $\mathbf{q_2}$

$$\mathbf{cost\ (P,\ q_1) < cost\ (P,\ q_2)\ if\ q_1 \prec q_2}$$

(i.e. spatial domination $\Rightarrow$ cost domination)



Contemporary Optimizer Behavior on <u>One-dimensional</u> ESS

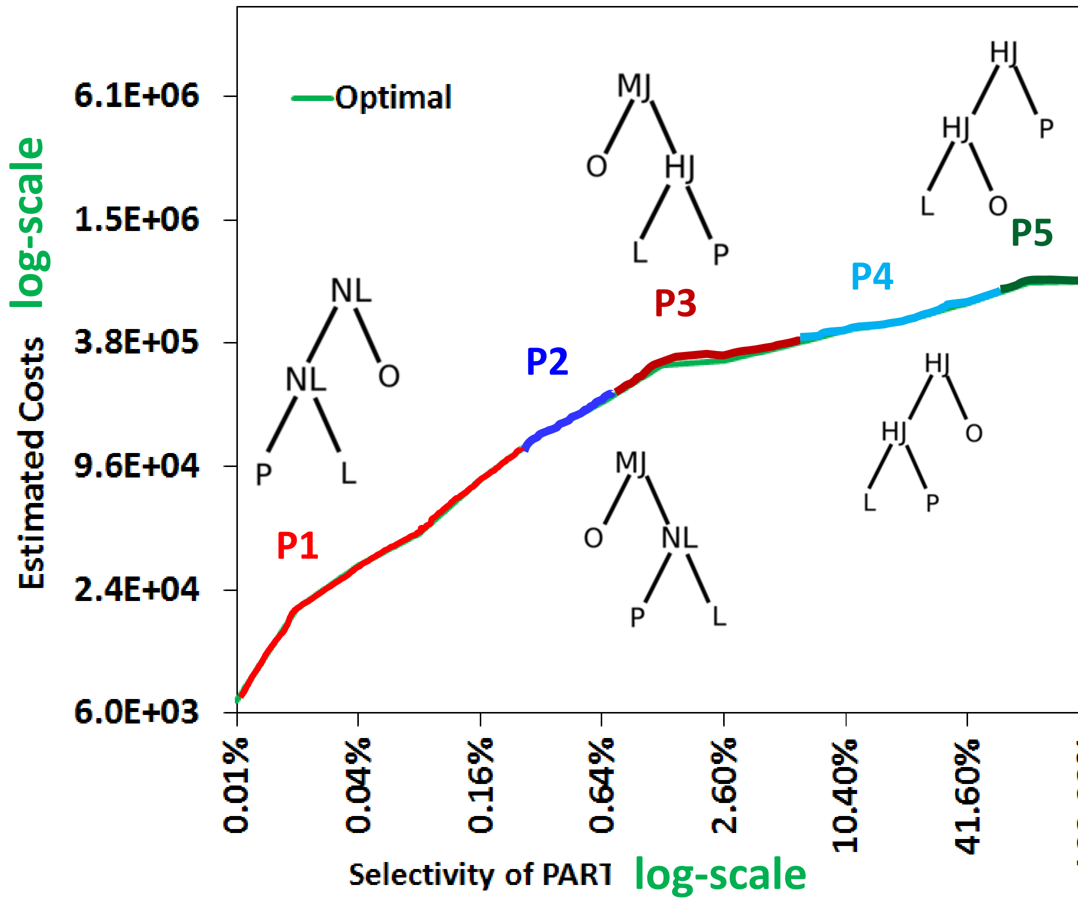
Parametric Optimal Set of Plans (POSP)

(Parametric version of EQ)
select *
from lineitem, orders, part
where p_partkey = l_partkey and
l_orderkey = o_orderkey and
SEL (PART) = $1

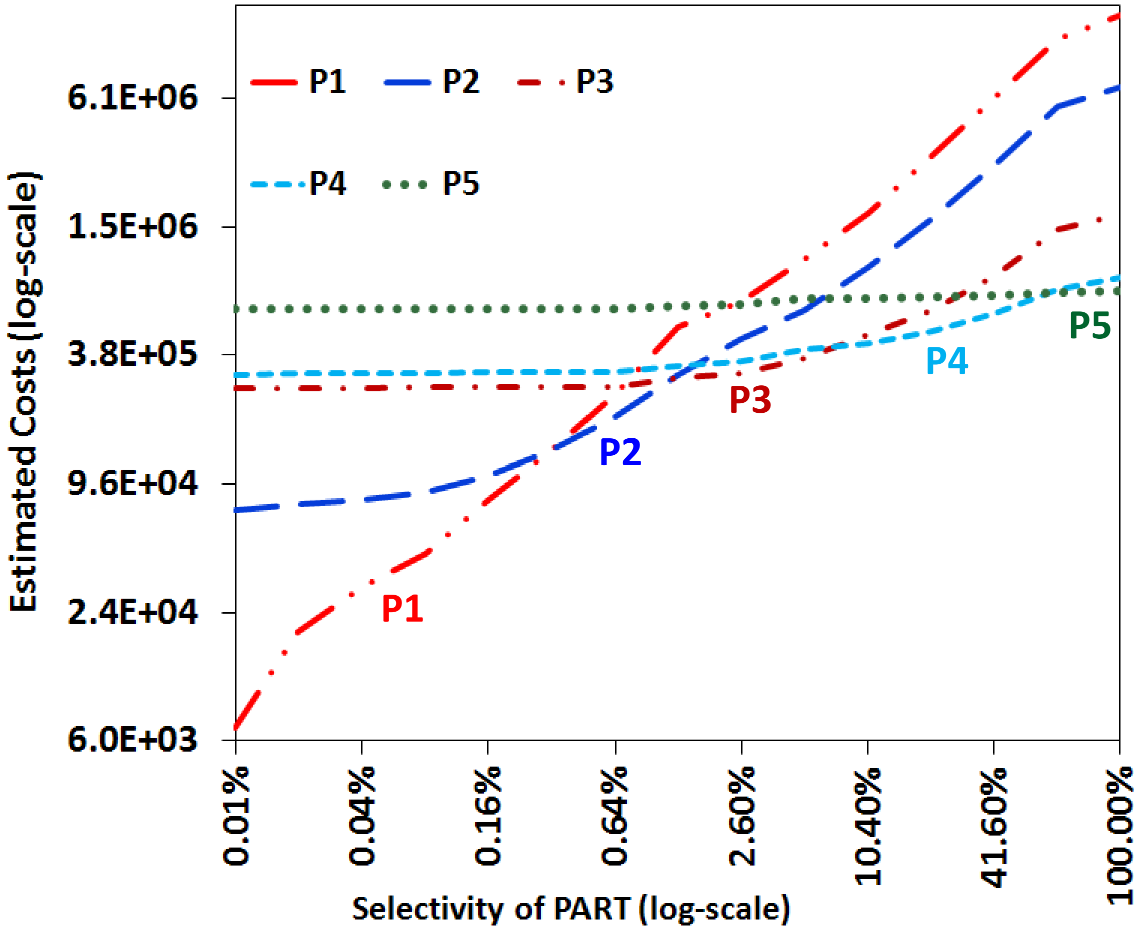
Using Selectivity Injection

NL: Nested Loop Join
MJ: Merge Join
HJ: Hash Join

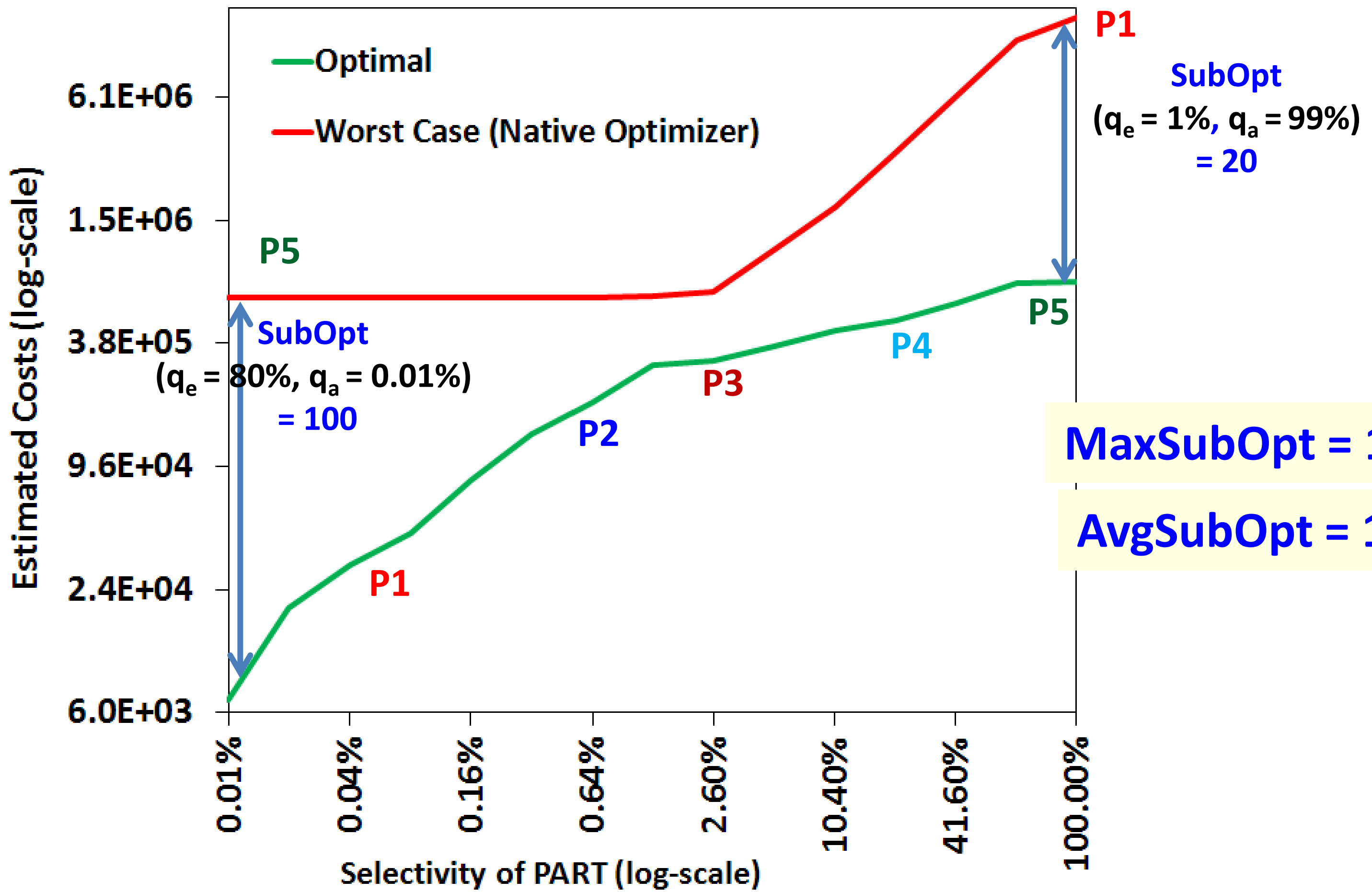
L: Lineitem
O: Orders
P: Part



POSP Performance Profile (across ESS)



Sub-optimality Profile (across ESS)



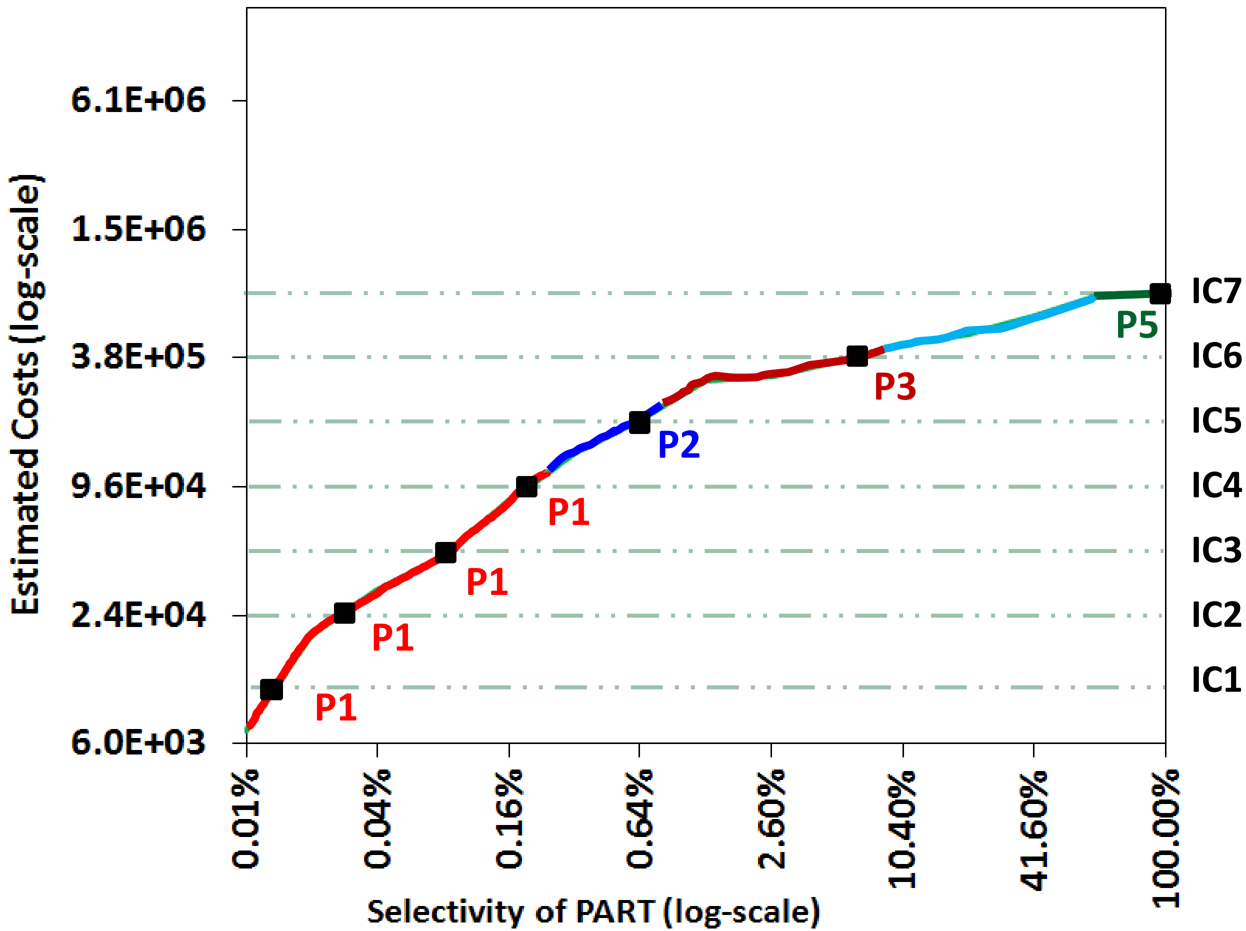
MaxSubOpt = 100

AvgSubOpt = 1.8

Plan Bouquet

Bouquet Approach in <u>1D ESS</u>

Plan Bouquet Identification

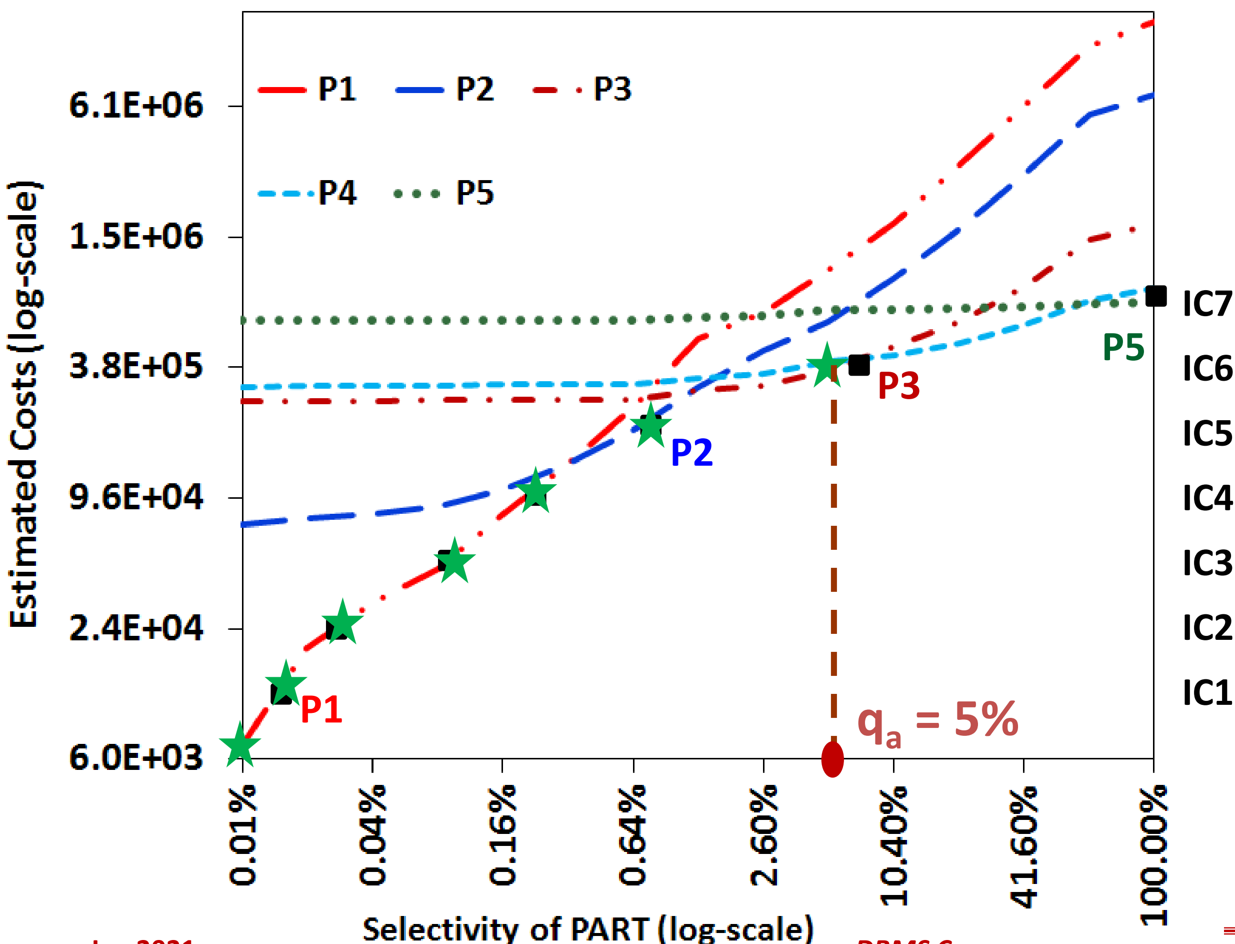


Step 1: Draw cost steps with cost-ratio **r=2** (geometric progression).

Step 2: Find plans at intersection of optimal profile with cost steps

Bouquet = {P1, P2, P3, P5}

Bouquet Execution



Let q_a = 5%

(1) Execute P1 with budget IC1(1.2E4)

(2) *Throw away results of P1*
Execute P1 with budget IC2(2.4E4)

(3) *Throw away results of P1*
Execute P1 with budget IC3(4.8E4)

(4) *Throw away results of P1*
Execute P1 with budget IC2(9.6E4)

(5) *Throw away results of P1*
Execute P2 with budget IC5(1.9E5)

(6) *Throw away results of P2*
Execute P3 with budget IC6(3.8E5)

P3 completes with cost 3.4E5

Stupid Ideas?

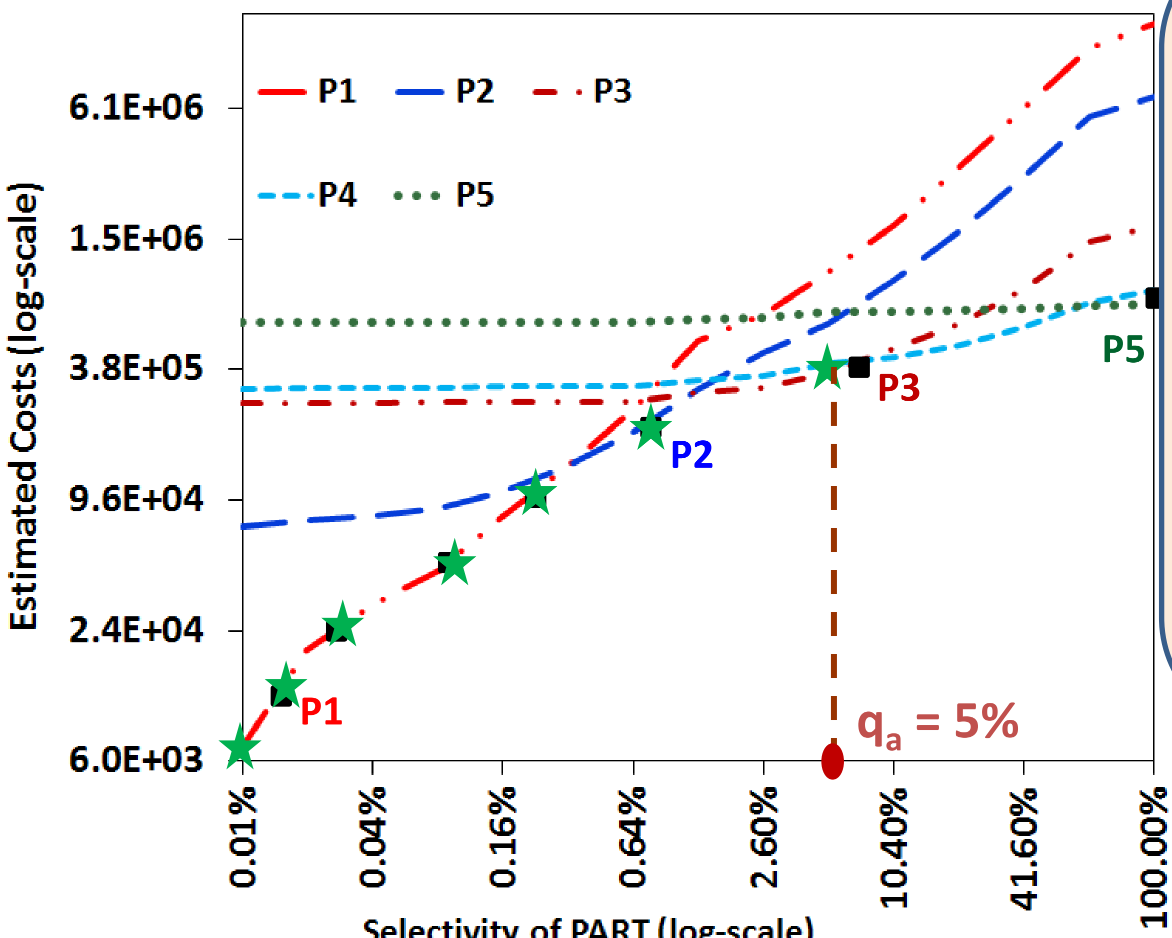
Yes! Very stupid!

We are expending lots and lots of wasted effort at both planning time (producing PIC) and at execution time (throwing away work) !

Certainly a recipe for disaster …

But, with careful engineering, can actually be made to work surprisingly well → rest of talk

Plan Bouquet Execution



Let q_a = 5%

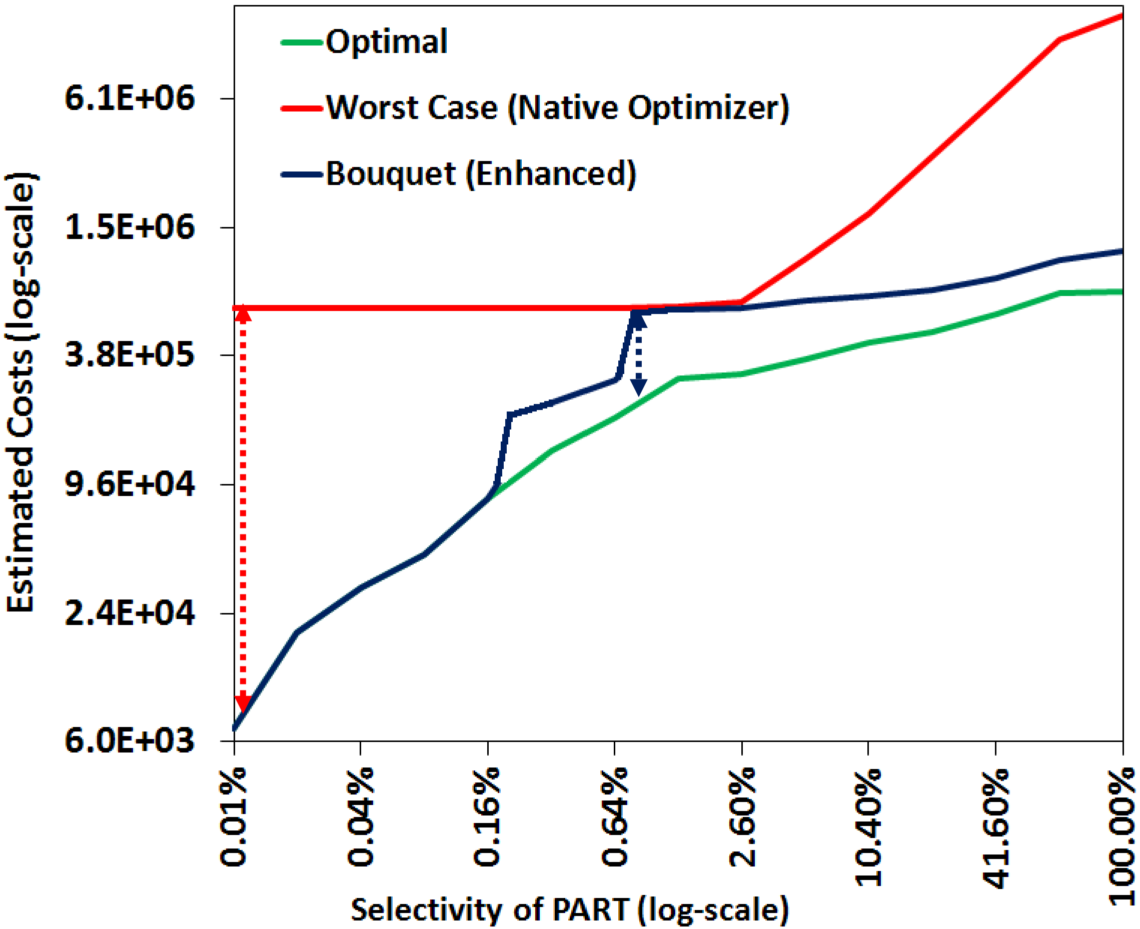
Bouquet Cost = 3.4 E5 (P3) +
1.92 E5 (P2) +
0.96 E5 (P1) +
0.48 E5 (P1) +
0.24 E5 (P1) +
0.12 E5 (P1)
= 7.1 E5

SubOpt (*, 5%) = 7.1/3.4 = 2.1

With obvious optimization
SubOpt(*, 5%) = 6.3/3.4 = 1.8

with budget IC6(3.8E5)
P3 completes with cost 3.4E5

SO Performance over ESS



Native Optimizer

MaxSubOpt = 100

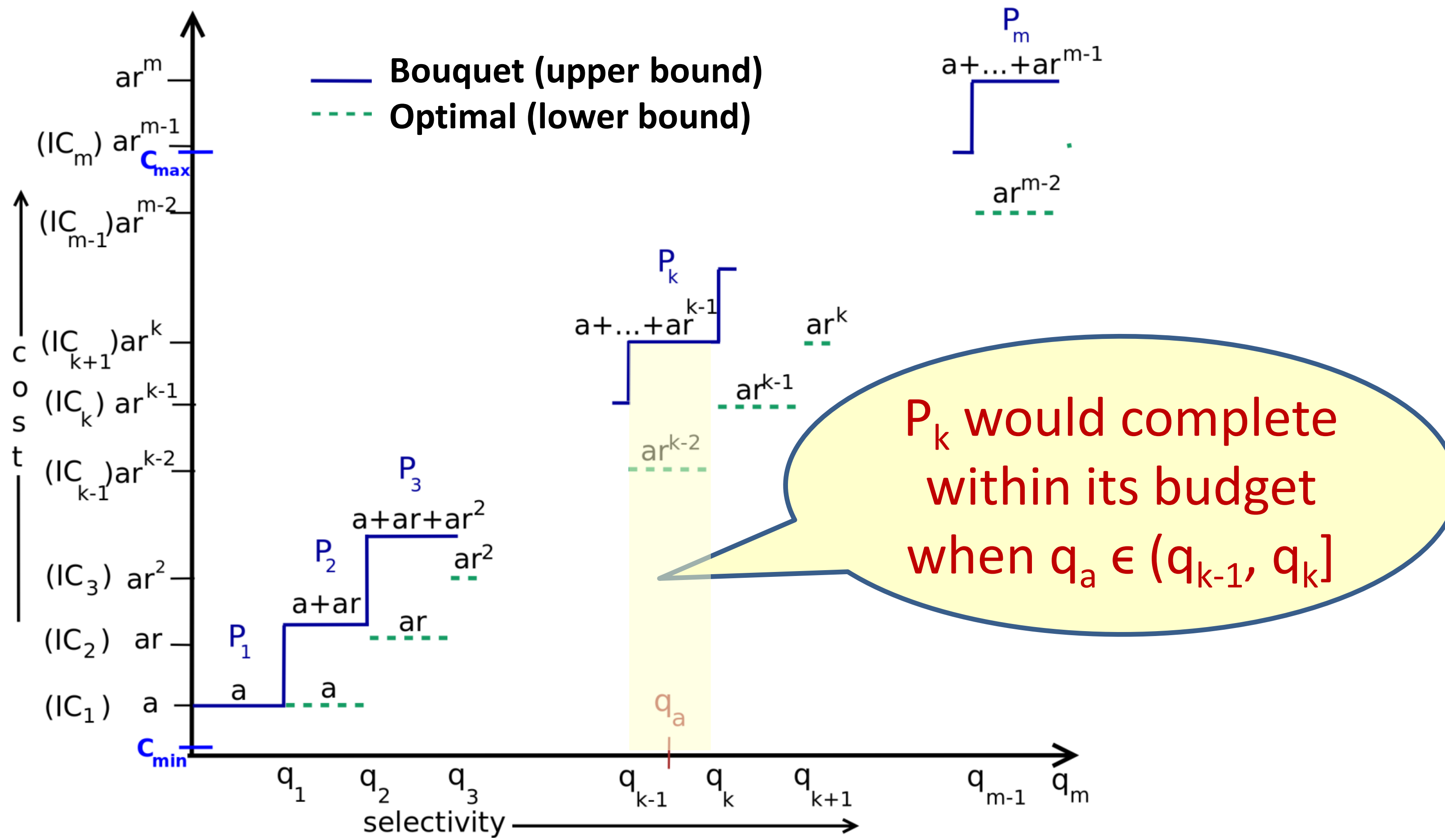
AvgSubOpt = 1.8

Bouquet (Enhanced)

MaxSubOpt = 3.1

AvgSubOpt = 1.7

Worst Case Cost Analysis



1D Performance Bound

$$C_{bouquet}(q_{k-1}, q_k] = cost(IC_1) + cost(IC_2) + \ldots + cost(IC_{k-1}) + cost(IC_k)$$
$$= a + ar + \ldots + ar^{k-2} + ar^{k-1}$$
$$= \frac{a(r^k - 1)}{(r-1)}$$

$$C_{optimal}(q_{k-1}, q_k] \geq ar^{k-2}$$ (Implication of PCM)

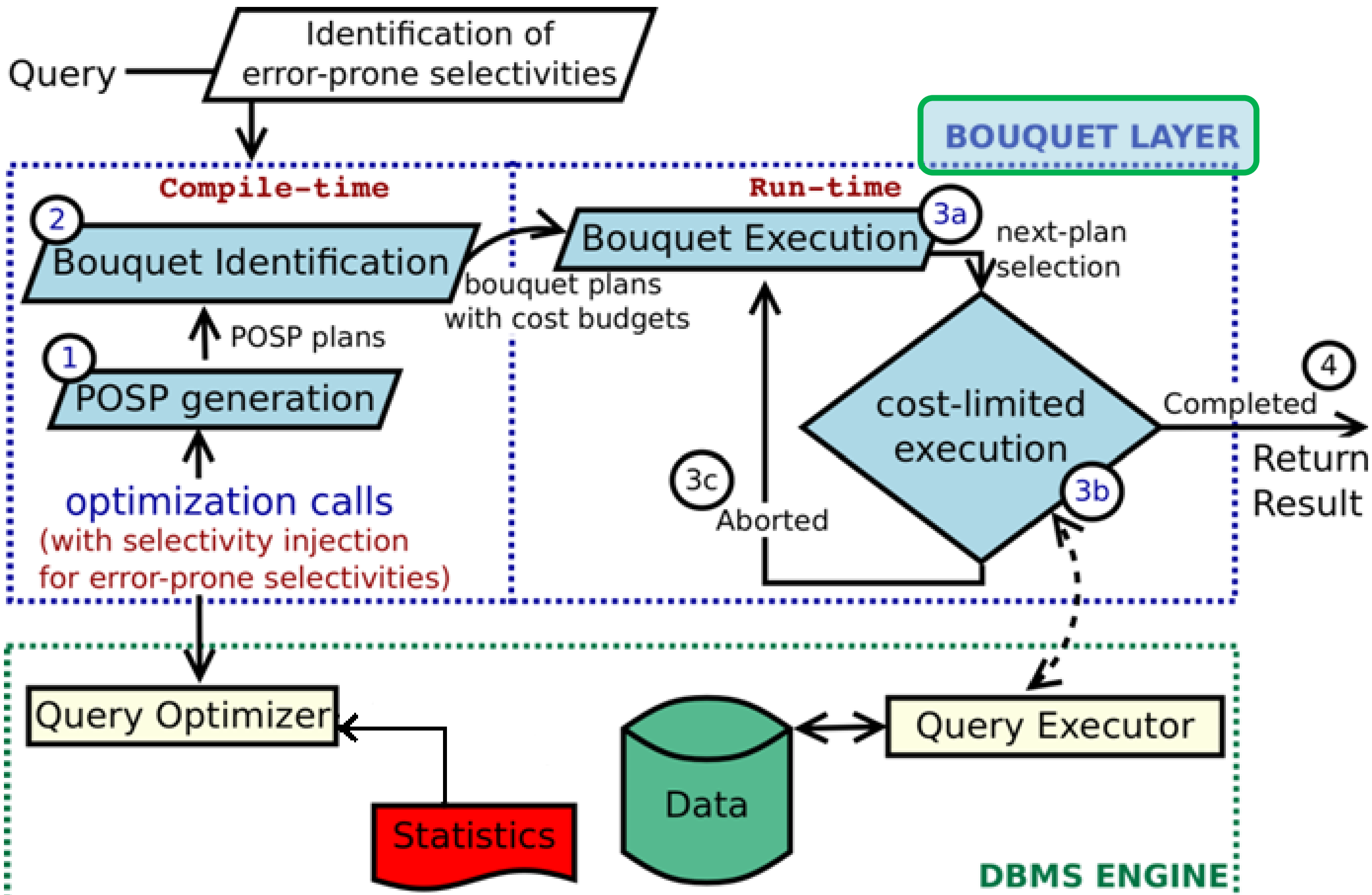
$$SubOpt_{bouquet}(*, q_a) \leq \frac{1}{ar^{k-2}} \times \frac{a(r^k - 1)}{(r-1)} \leq \frac{r^2}{r-1} \quad \forall q_a \in (q_{k-1}, q_k]$$

Reaches minima at r = 2

➔ MSO = 4

Best performance achievable by any deterministic online algorithm!

Bouquet Architecture



Connection to Online Bidding Problem

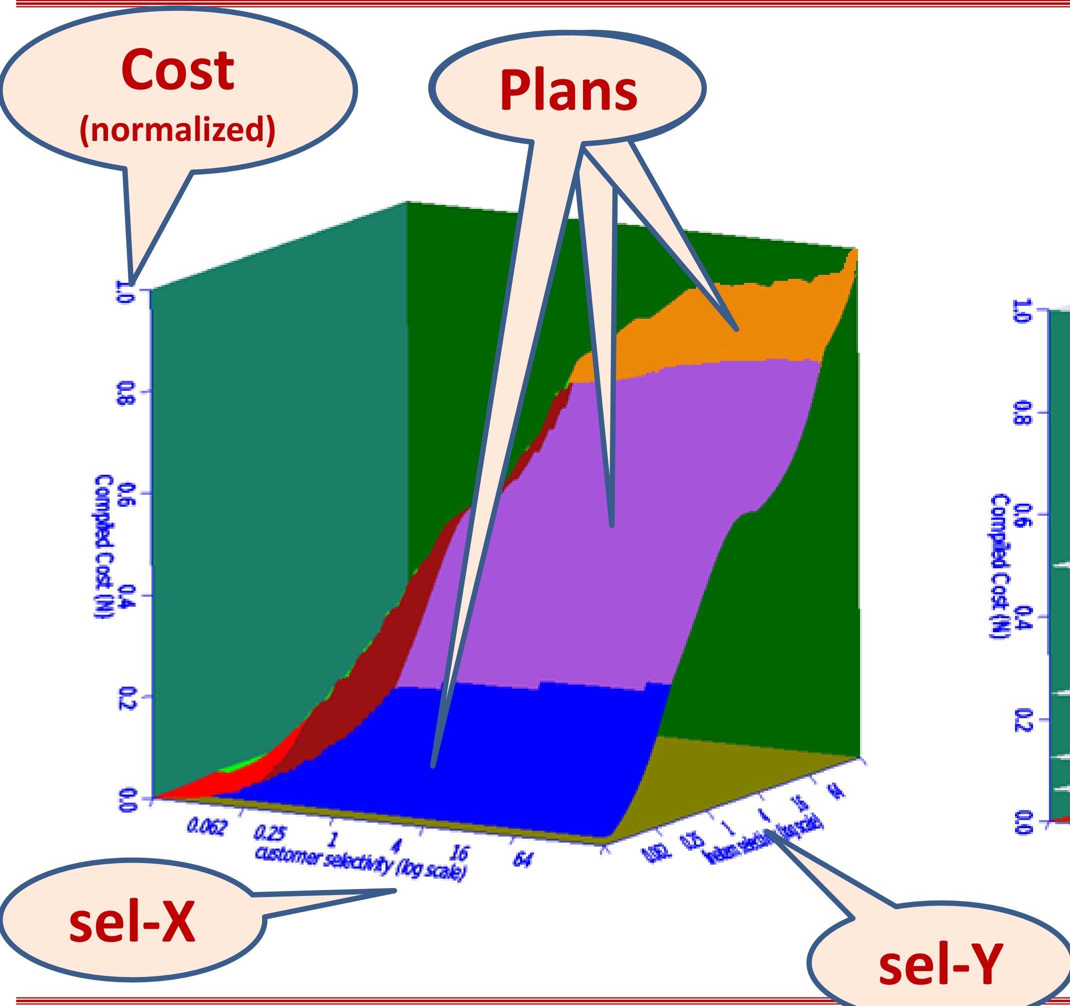
- There is an object D with hidden value V in range (1, 100)
- Your task is to bid for D until you acquire it under the following rules:
  - If the bid B < V, then you forfeit B, and bid again
  - If the bid B ≥ V, then you pay B and acquire D
- Your goal is to minimize the worst-case ratio of your total payment to the object value, i.e.

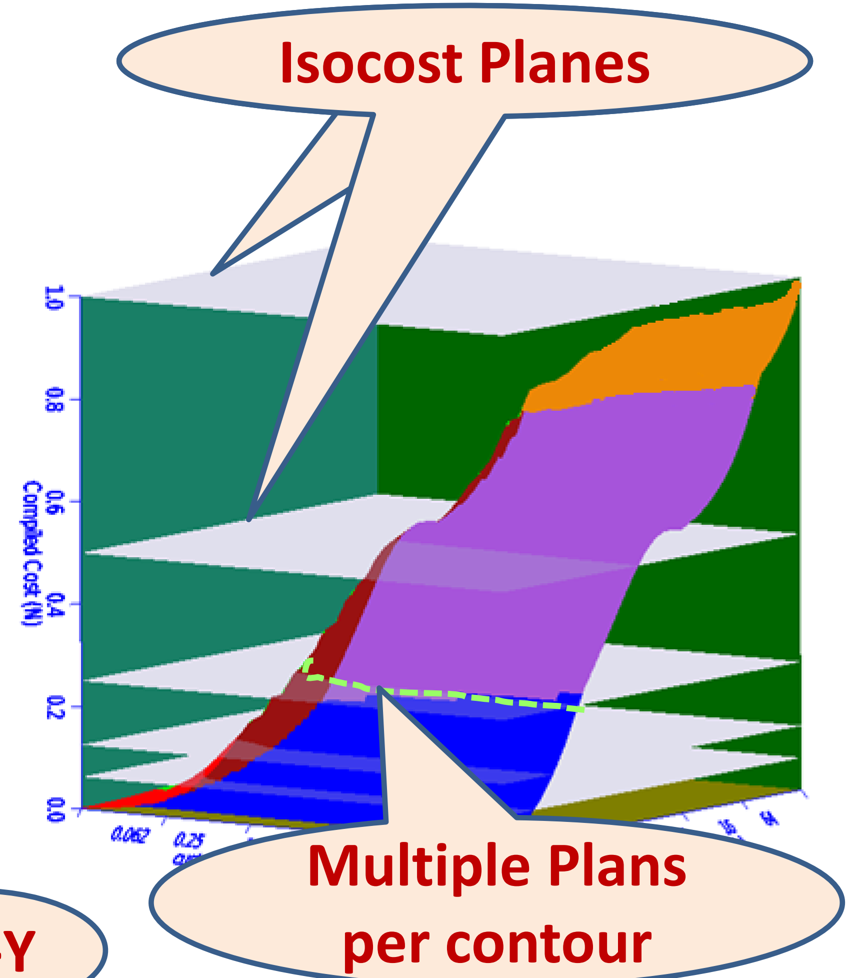
$$\min \left((B_1 + B_2 + \ldots + B_k) / V \right)$$

- Bid doubling algorithm is best possible choice

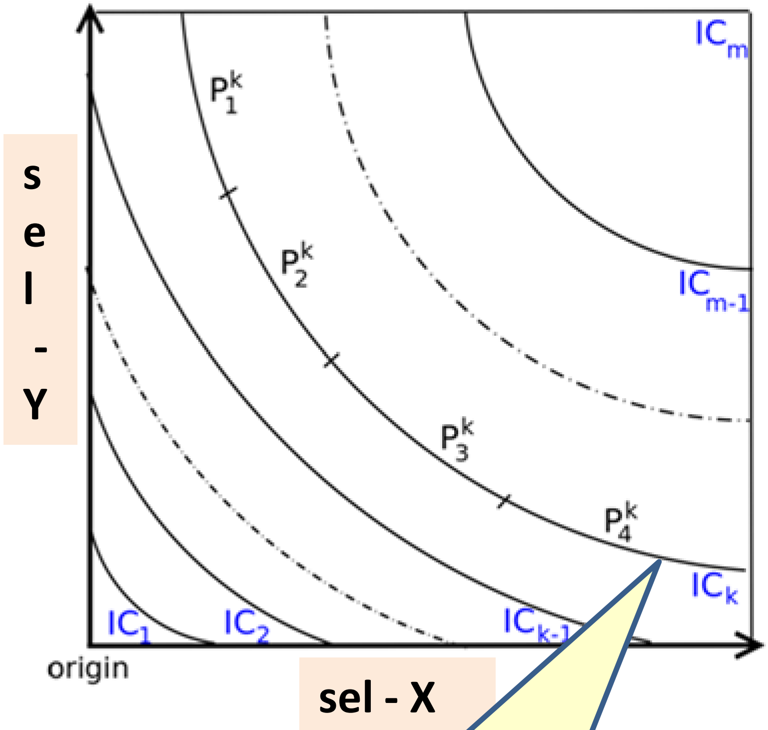
Bouquet Approach in 2D SS

2D Bouquet Identification





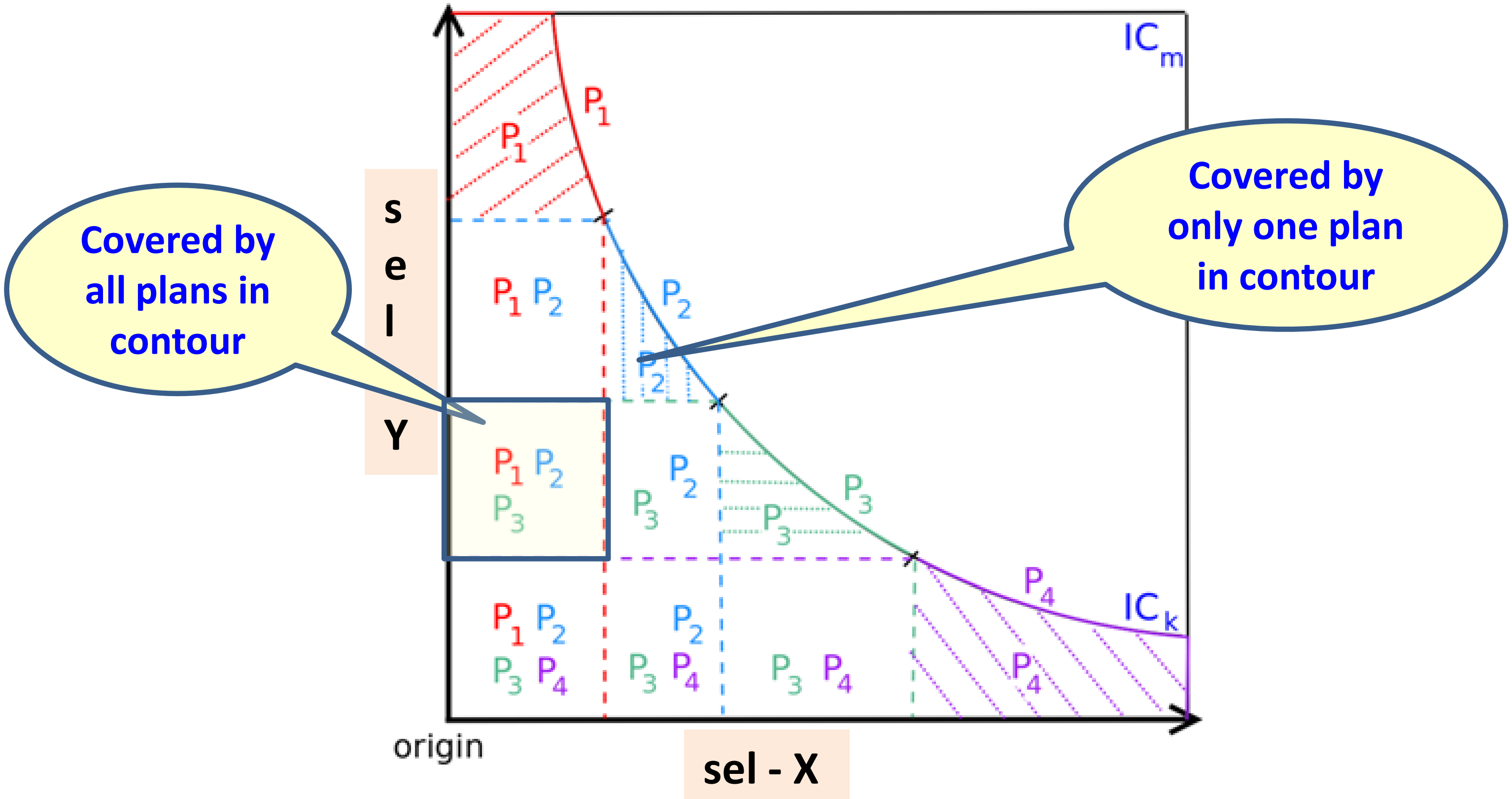
Characteristics of 2D Contours



2D contours
- **Hyperbolic curves**
- **Multiple plans per contour**

Third quadrant coverage (due to PCM)
P_2^k can complete any query with actual selectivities(q_a) in the shaded region within cost(IC_k)

Crossing 2D Contours



⇒ Entire set of contour plans must be executed to fully cover all locations under IC_k

2D Performance Analysis

- **When $q_a \in (IC_{k-1}, IC_k]$**

Number of plans on i^{th} contour

$$C_{bouquet}(q_a) = \sum_{i=1}^{k} [n_i \times cost(IC_i)]$$

$\rho = \max(n_i)$

$$C_{bouquet}(q_a) \leq \rho \times \sum_{i=1}^{k} cost(IC_i)$$

$$SubOpt_{bouquet}(q_a) = 4\rho$$ (Using 1D Analysis)

Bound for N-dimensions: $SubOpt_{bouquet}(q_a) = 4 \times \rho_{ICsurface}$

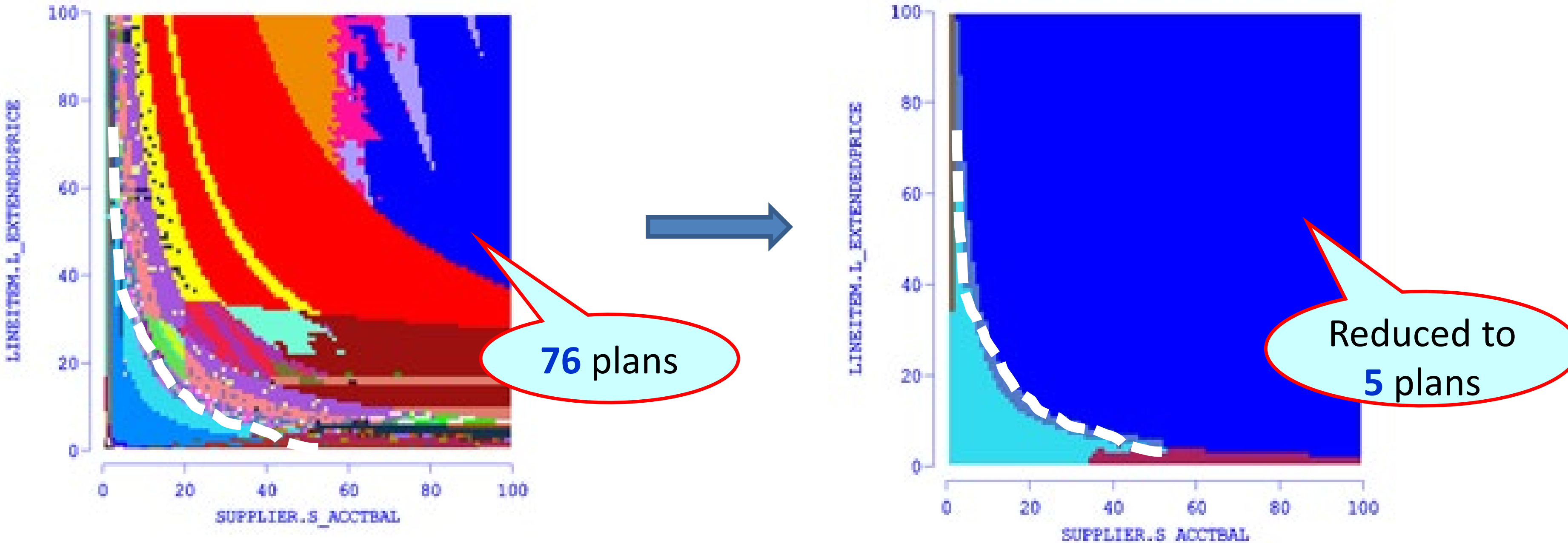
Dealing with large ρ

- In practice, ρ can often be large, even in 100s, making the performance guarantee of 4ρ impractically weak

- Reducing ρ
  - Compile Time:
    - Anorexic POSP reduction [Cost Greedy]
  - Run Time:
    - Explicit Monitoring of Selectivity Lower Bounds
    - Spilling-based Execution

1) Reducing ρ with Anorexic Reduction

- Collapse a large set of POSP plans on a selectivity space into a reduced cover that provides performance within a (1+ λ) factor of the optimal at all locations in the ESS. With λ = 0.2, invariably obtain a small-sized (< 10) cover.



MSO guarantees (compile time)

	Query (dim)	ρ_{POSP}	MSO Bound (POSP) $= 4\rho_{POSP}$	$\rho_{reduced}$ (λ=0.2)	MSO Bound (reduced) $= 4\rho_{reduced}(1+\lambda)$
TPC-H	Q5 (3D)	11	44	3	14.4
	Q7 (3D)	13	52	3	14.4
	Q8 (4D)	88	352	7	33.6
	Q7 (5D)	111	444	9	43.2
TPC-DS	Q15 (3D)	7	28	3	14.4
	Q96 (3D)	6	24	3	14.4
	Q7 (4D)	29	116	4	19.2
	Q19 (5D)	159	636	8	38.4
	Q26 (4D)	25	100	5	24.0
	Q91 (4D)	94	376	9	43.2

2) Reducing ρ with Selectivity Monitoring

- When (cost-limited) execution of plans on IC_k does not complete the query, we know that q_a does not lie under IC_k
  - but q_a could lie anywhere beyond IC_k

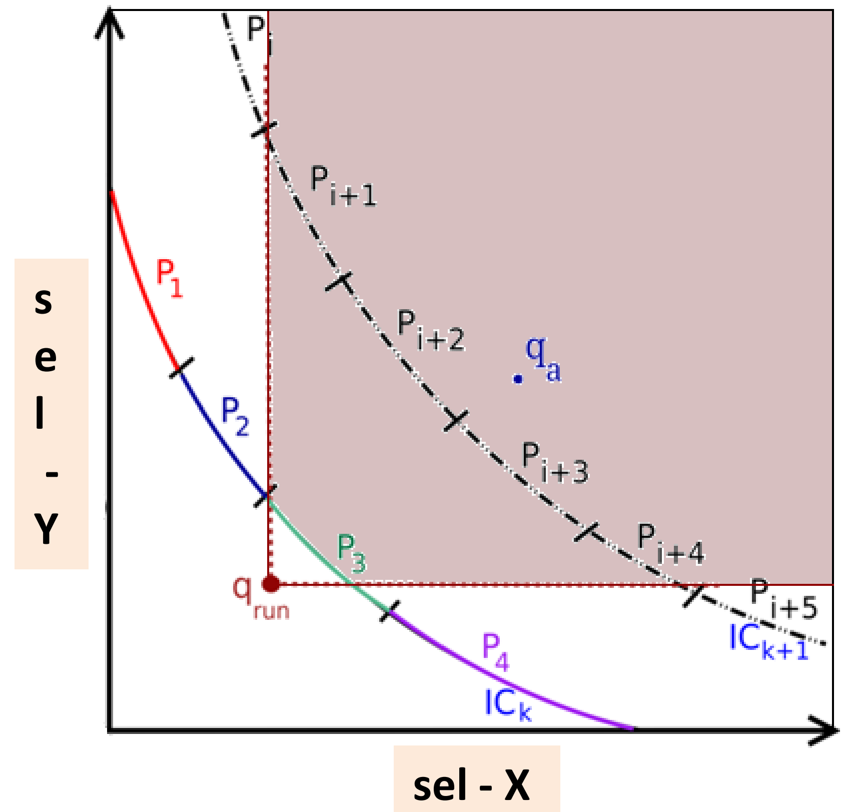
- By monitoring lower bounds on selectivities during execution **(q_{run})**

  q_a can only be in **first quadrant** of q_{run}

  (# of tuples at a node can only be greater than what has already been seen)

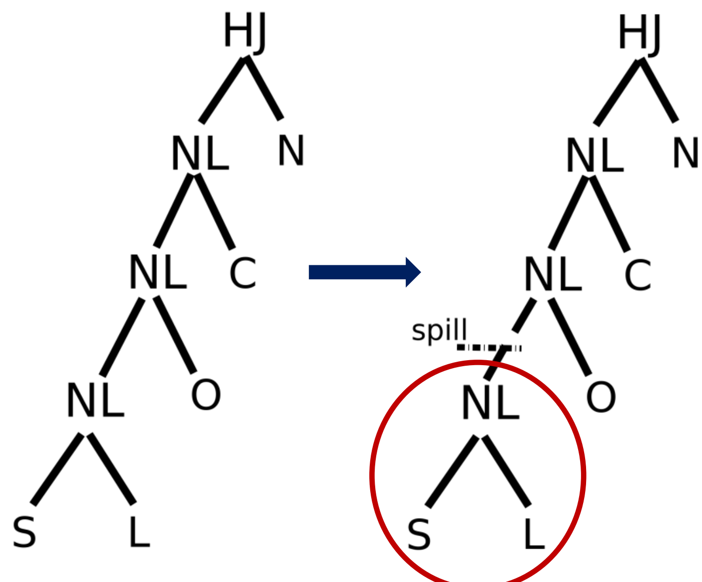
  ➔(P_i, P_{i+5} need not be executed)

  ➔lesser effective value of ρ



3) Maximizing selectivity movement

- The selectivity movement at a node N in the plan tree is increased by "spilling" (dropping without forwarding) the output of N, thereby focusing the entire execution budget on the sub-tree rooted at N.



Spill modification to a plan

To enhance movement of join selectivity SL, the join output tuples are **spilled**, instead of being forwarded to the upstream nodes.

SpillBound [Followup Paper]

For D dimensions,
MSO guarantee is $D^2 + 3D$

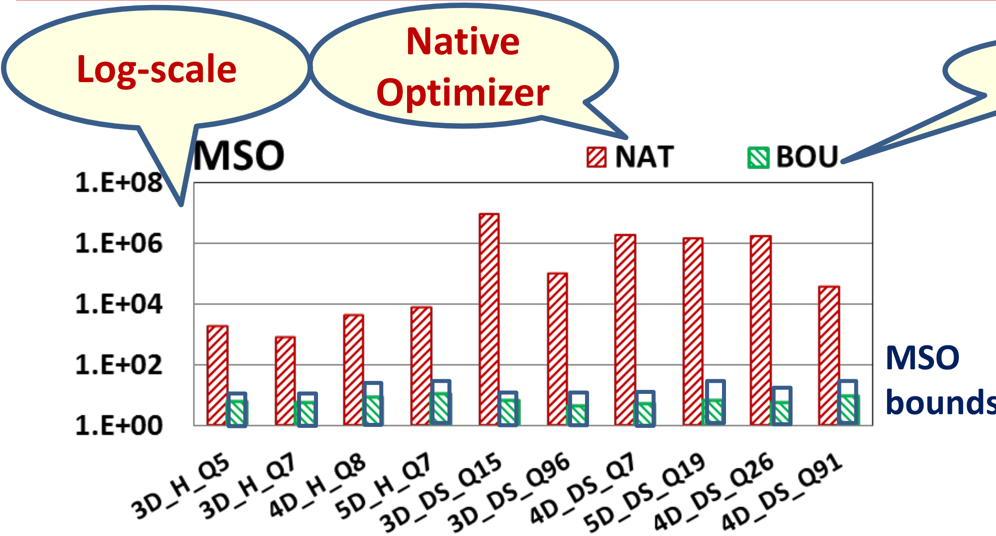
Platform-independent for a query

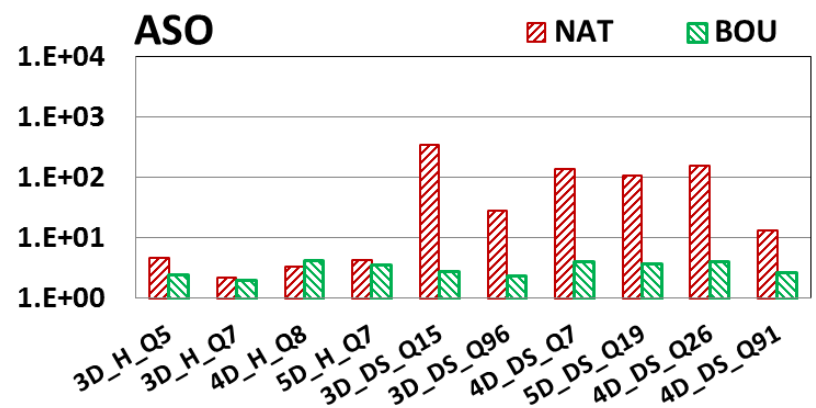
Empirical Evaluation

Experimental Testbed

- Database Systems: PostgreSQL and COM (commercial engine)
- Databases: TPC-H and TPC-DS
- Physical Schema: Indexes on all attributes present in query predicates
- Workload: 10 complex queries from TPC-H and TPC-DS
  - with SS having upto **5 error dimensions**
- Metrics: Computed MSO and ASO using Abstract Plan Costing over SS

Performance on PostgreSQL

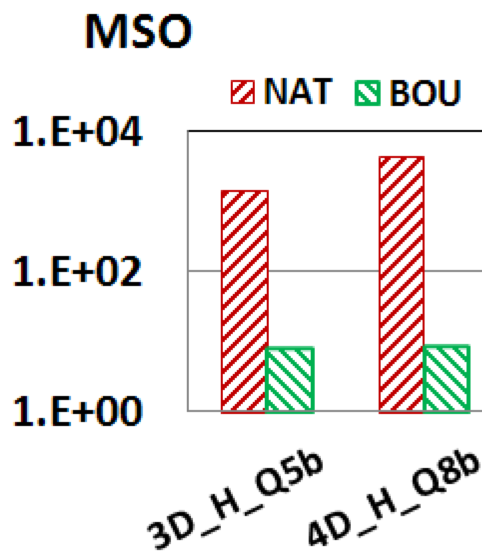


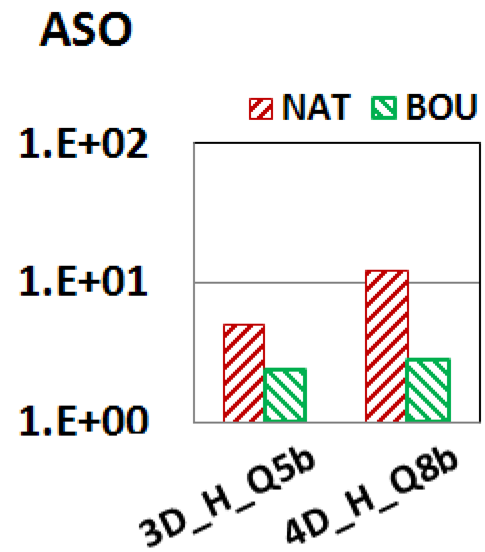


- For many DS queries
  - MSO improves from ≈10^6 to ≈10
  - ASO improves from ≈10^2 to ≈ 5

ASO not compromised to reduce MSO!

Performance with COM





⇒Robustness improvements not artifact of a specific engine

Sample Savings in Wall-clock Time

Performance Summary	NAT (PostgreSQL)	Enhanced Bouquet	Optimal
	600 sec	69 sec	16.1 sec

Contour ID	Avg. Execution Time (in sec)	# Executions (Enhanced Bouquet)	Time (in sec) (Enhanced Bouquet)
1	0.6	2	1.2
2	3.1	2	6.2
3	4.8	3	14.4
4	6.2	3	18.6
5	12.2	1	12.2
6	16.1	1	16.1
Total		**12**	**68.7**

In spite of uncalibrated cost model

Query based on TPC-H Q8

Summary

- Plan bouquet approach achieves
  - bounded performance sub-optimality
    - using a (cost-limited) plan execution sequence guided by isocost contours defined over the optimal performance curve
  - robust to changes in data distribution
    - only q_a changes – bouquet remains same
  - easy to deploy
    - bouquet layer on top of the database engine
  - repeatability in execution strategy (important for industry)
    - q_e is always zero, depends only on q_a
    - independent of metadata contents

Important distinction from re-optimization techniques

Incorporating Cost Model Error

- If cost model error is bounded by $\boldsymbol{\delta}$, that is

$$\frac{cost_{estimated}}{cost_{actual}} \in \left[\frac{1}{1+\delta}, 1+\delta\right]$$

then

$$MSO_{bounded} \leq MSO_{perfect} * (1+\delta)^2$$

– for δ = 0.4 ➔ $\text{MSO}_{\text{bounded}} \leq 2\ \text{MSO}_{\text{perfect}}$

For more details, visit project website:

dsl.cds.iisc.ac.in/projects/QUEST

- **Concepts paper: ACM SIGMOD 2014**
- **Demo paper: VLDB 2014 (Best Demo Award)**
- **Concepts + Implementation: ACM TODS (June 2016)**
- **Followup Work: IEEE ICDE 2017 (Best Student Paper Award)**
  + IEEE TKDE 2017 + VLDB 2019
- **Tutorial on RQP: IEEE ICDE 2019, VLDB 2020,**
  Cods-Comad 2020

Take Away

SQL Query

Do you know the correct selectivities ?

Near Optimal Query Execution Performance