

Robust Contour Discovery for Plan Bouquets

Achint Chaudhary

MTech

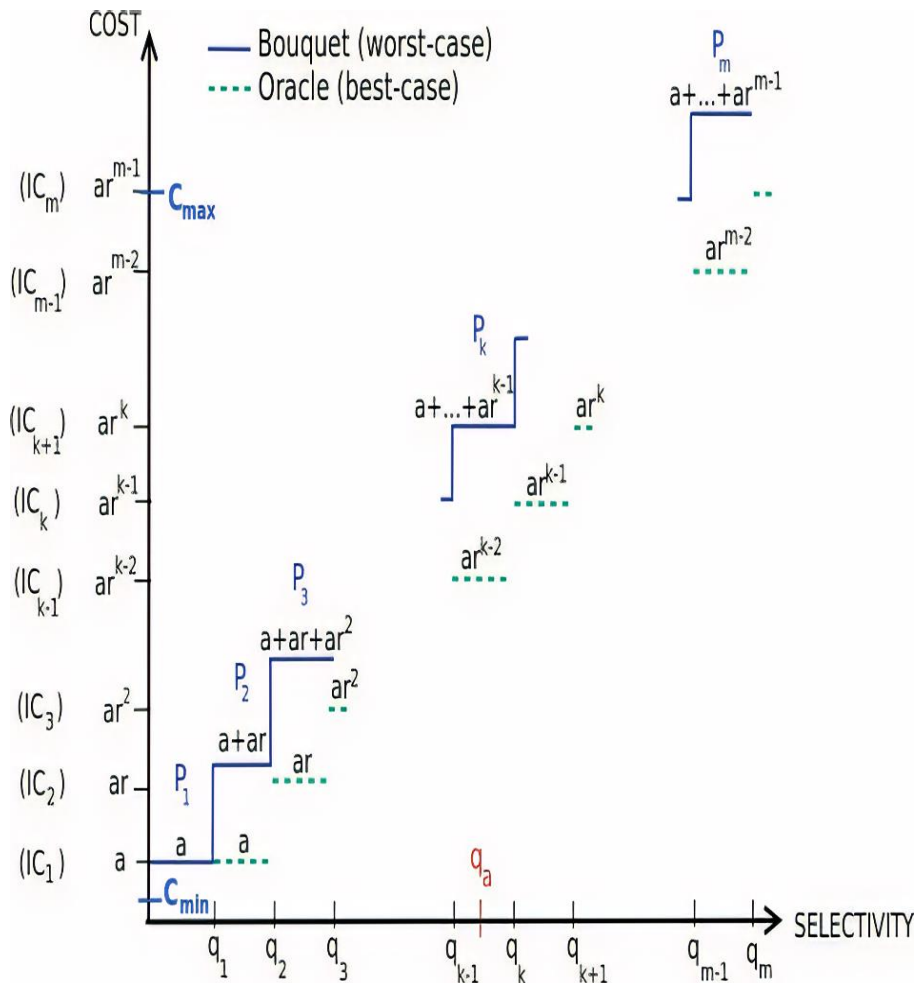
Advisor

Prof. Jayant R. Haritsa

Introduction

- *SQL* queries are declarative in nature
- Many execution strategies for a SQL query are possible, each is called *query plan*
- Database optimizer select a plan for execution
- Plan choices are suboptimal, as they are based on selectivity estimation
- *Plan bouquets* was proposed to provide worst case performance guarantees for OLAP queries

Plan Bouquets



$$C_{pb}(q_a) = CC_1 + CC_2 + \dots + CC_k$$

$$= a * r_{pb}^0 + a * r_{pb}^1 + \dots + a * r_{pb}^{k-1}$$

$$= \frac{a(r_{pb} - 1)}{r_{pb} - 1}$$

$$SubOpt(*, q_a)$$

$$= \frac{a * (r_{pb}^k - 1)}{(r_{pb} - 1)}$$

$$\leq \frac{a * r_{pb}^{k-2}}{a * r_{pb}^{k-2}}$$

$$\leq \frac{r_{pb}^2}{r_{pb} - 1}$$

$$Using r_{pb} = 2, MSO = 4$$

Performance Metrics

■ Suboptimality

$$Subopt(q_e, q_a) = \frac{Cost(P_{opt}(q_e), q_a)}{Cost(P_{opt}(q_a), q_a)} \quad Subopt(*, q_a) = \frac{\sum_{exec \in BS} Budget(exec)}{Cost(P_{opt}(q_a), q_a)}$$

■ Worst Suboptimality

$$Subopt_{worst}(q_a) = \max_{q_e \in ESS} Subopt(q_e, q_a)$$

■ Maximum Suboptimality (MSO)

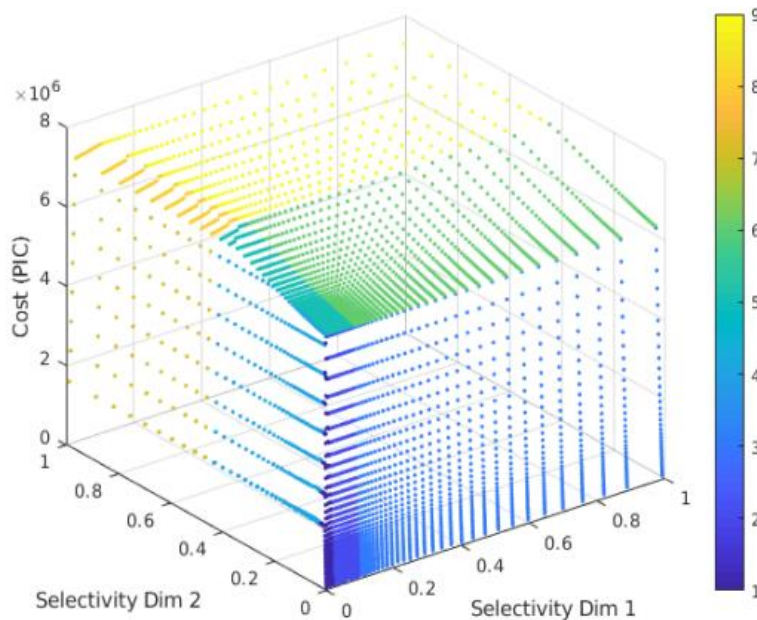
$$MSO = \max_{q_a \in ESS} Subopt_{worst}(q_a)$$

$$MSO = \max_{q_a \in ESS} Subopt(*, q_a)$$

Assumptions

- Plan Cost Monotonicity (PCM)
- Axis Parallel Concave (APC)
- Perfect Cost Model of Optimizer
- Bounded Cost Growth (BCG)
- Piece-wise Axis Parallel Linear (APL)
- Selectivity Independence

Linearity & BCG of OCS



(a) Exemplar piecewise linear OCS

For any PCF \mathcal{F}_p

$$\mathcal{F}_p(\alpha * q.j) \leq f(\alpha) * \mathcal{F}_p(q.j)$$

$$\forall j \in \{1, \dots, D\}, \forall \alpha \geq 1$$

Common observation is

$$f(\alpha) = \alpha$$

(b) Definition of BCG

Note: APL and BCG behavior of Plan Cost Functions implies similar behavior of OCS

Bouquets Compilation

■ Full Space Exploration

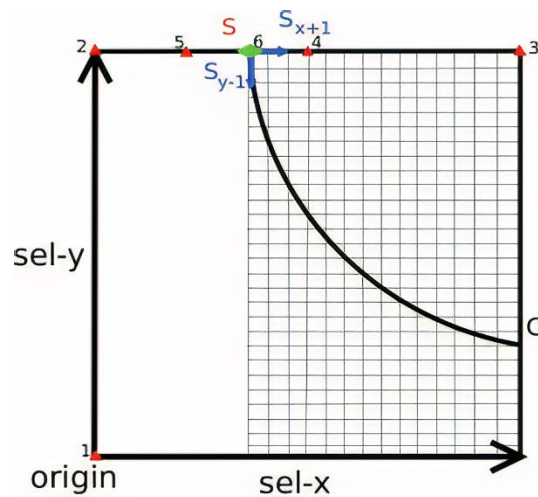
- ▶ Do optimizer calls at all points, and take only feasible points, i.e. those whose cost lies in $[CC_i, CC_i(1 + \alpha)]$
- ▶ Total calls made are $\Theta(RES^{Dim})$

■ NEXUS

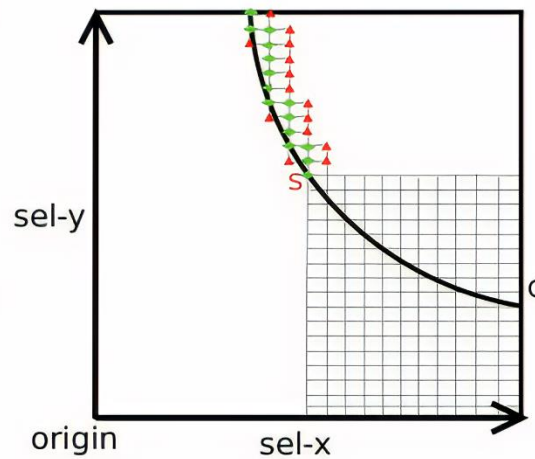
- ▶ Initial seed on one boundary is located
- ▶ Neighboring points of seed are explored to create contours
- ▶ Worst case calls made are $\Theta(m * Dim * RES^{Dim})$
- ▶ Speed-up guaranteed when $m * Dim < RES$

Understanding NEXUS

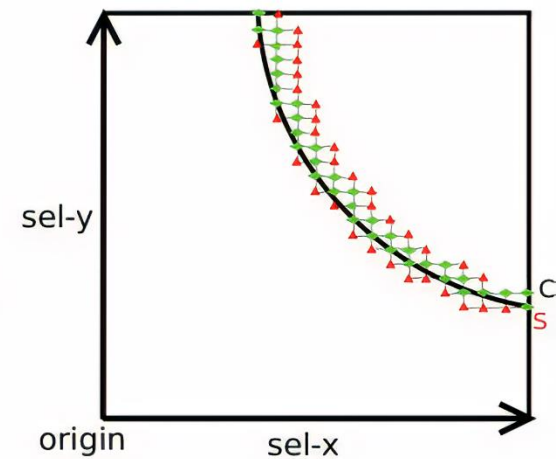
- Reduces optimizer calls by avoiding all points of ESS
- n – *dimensional* version solved by multiple 2D exploration in recursive fashion



(a) Finding seed location



(b) Contour exploration (intermediate)



(c) Contour exploration (finished)

MSO_g on Cost Deviation

Let,

The total number of contours be k (IC_1, IC_k)

Cost of all contours except IC_{k-1} is deviated by $(1 + \alpha)$

Cost Incurred due to Bouquet Sequence

$$BS_{cost} = CC_{k-1} + (1 + \alpha) * (CC_k + \sum_{i=1}^{k-2} CC_i)$$

Best possible cost

$$OPT_{cost} = \lim_{h \rightarrow 0} (CC_{k-1} + h) = CC_{k-1}$$

$$MSO_g = \frac{BS_{cost}}{OPT_{cost}} = \frac{(1 + \alpha) * \sum_{i=1}^k CC_i - \alpha * CC_{k-1}}{CC_{k-1}}$$

$$\begin{aligned} &= (1 + \alpha) * \frac{r_{pb}^2}{r_{pb}-1} - \alpha \\ &\leq \frac{r_{pb}^2}{r_{pb}-1} * (1 + \alpha) = \frac{r_{pb}^2}{r_{pb}-1} * \eta \end{aligned}$$

Using $r_{pb} = 4$, we will get MSO_g

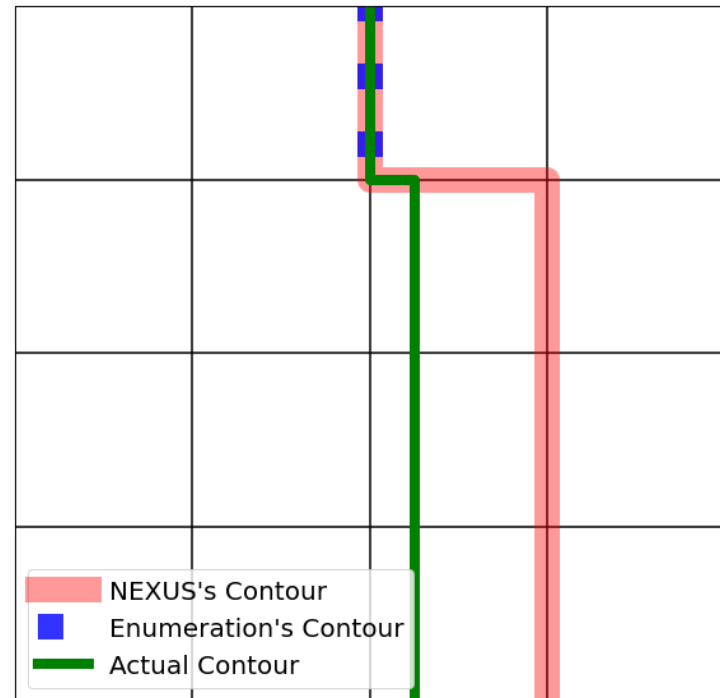
$$4 * (1 + \alpha) = 4 * \eta$$

Cost Deviation (η) should always respect

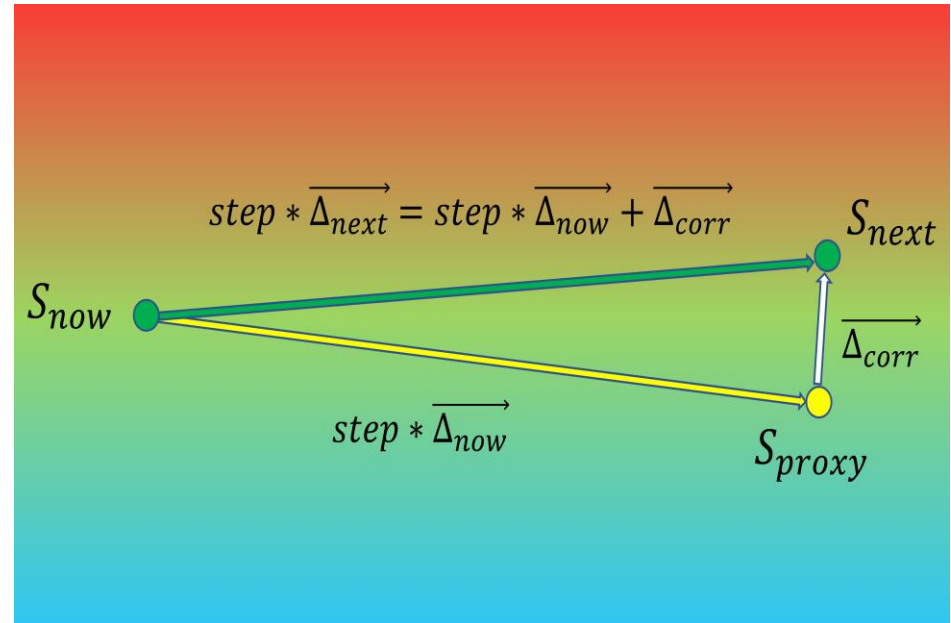
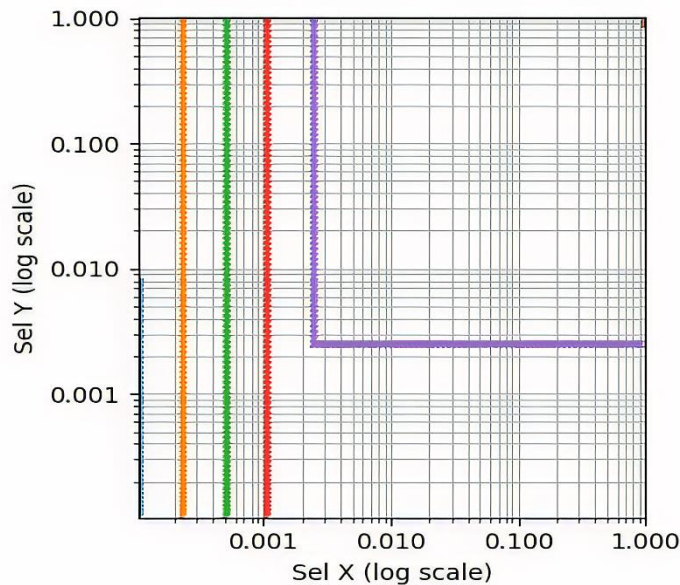
$$1 \leq (1 + \alpha) = \eta \ll r_{pb}$$

Cons of existing methods

- Discretization limits the search to finite number of points
- Cost deviated from ideal cost CC_i of contour IC_i
- May lead to slits in IC_i when used with the full space exploration



AdaNEXUS



- Piece-wise linear contours are observed in practice
- Step sizes during contour discovery can be increased exponentially when the further points are in a desired cost interval
- When the next given direction is wrong, aggressive search finds it

AdaNEXUS

```
def bisectionEXP(left, right) :  
     $P_L, P_R = P_{opt}(left), P_{opt}(right)$   
    if (left ≤ right) and ( $P_L \neq P_R$ ) :  
         $mid = \frac{left+right}{2}$   
         $P_M = P_{opt}(mid)$   
        if ( $P_L \neq P_M$ ) :  
            bisectionEXP(left, mid)  
        if ( $P_R \neq P_M$ ) :  
            bisectionEXP(mid, right)
```

```
def AdaNEXUS( $CC_i$ ) :  
     $S_{now}, P_{now} = InitializeSeed(CC_i)$   
     $C_{now} = Cost(P_{opt}, S_{now})$   
    step,  $\Delta_{now} = 1, [-1, 0]$   
    while (There exist next point) :  
         $S_{proxy} = S_{now} + step * \Delta_{now}$   
         $C_{proxy}, P_{proxy} = Cost(P_{opt}, S_{proxy})$   
         $S_{next}, \Delta_{next} = Correct(CC_i, C_{proxy})$   
         $C_{next}, P_{next} = Cost(P_{opt}, S_{next})$   
        if ( $\max(\frac{C_{next}}{C_i}, \frac{C_i}{C_{next}}) \leq (1 + \alpha)$ ) :  
             $\Delta_{now} = TuneDir(\Delta_{now}, \Delta_{next}, step)$   
            bisectionEXP( $S_{now}, S_{next}$ )  
             $S_{now}, step = S_{next}, 2 * step$   
        else :  
            if step > 1 :  
                step = step / 2  
            else :  
                 $\Delta_{now} = Rotate(S_{now}, CC_i)$ 
```

Experiments

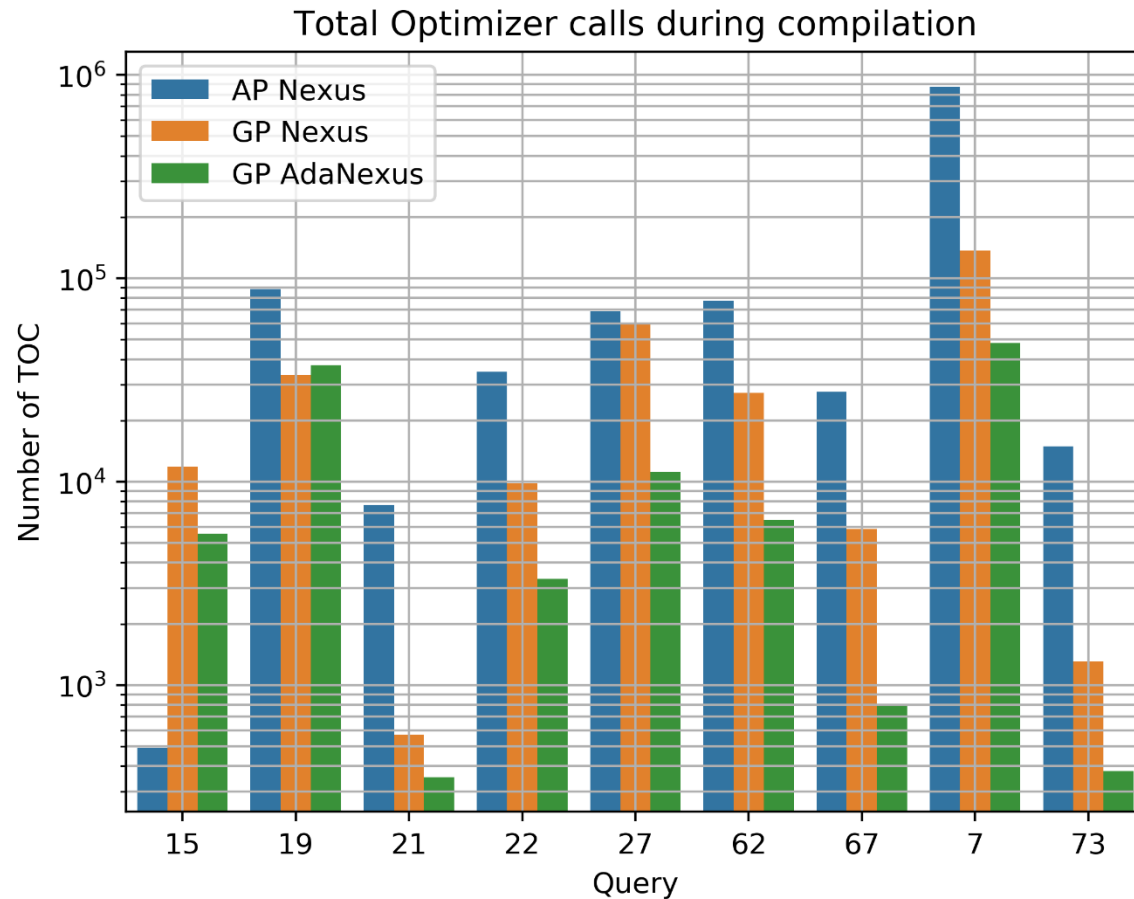
■ Database Environment

- ▶ Evaluation on 8 TPC-DS SQL queries
- ▶ Dimension of ESS from 3 to 5
- ▶ DB instances in range of 1GB to 250GB via CODD
- ▶ FPC module is used for Foreign Plan Costing

■ System Environment

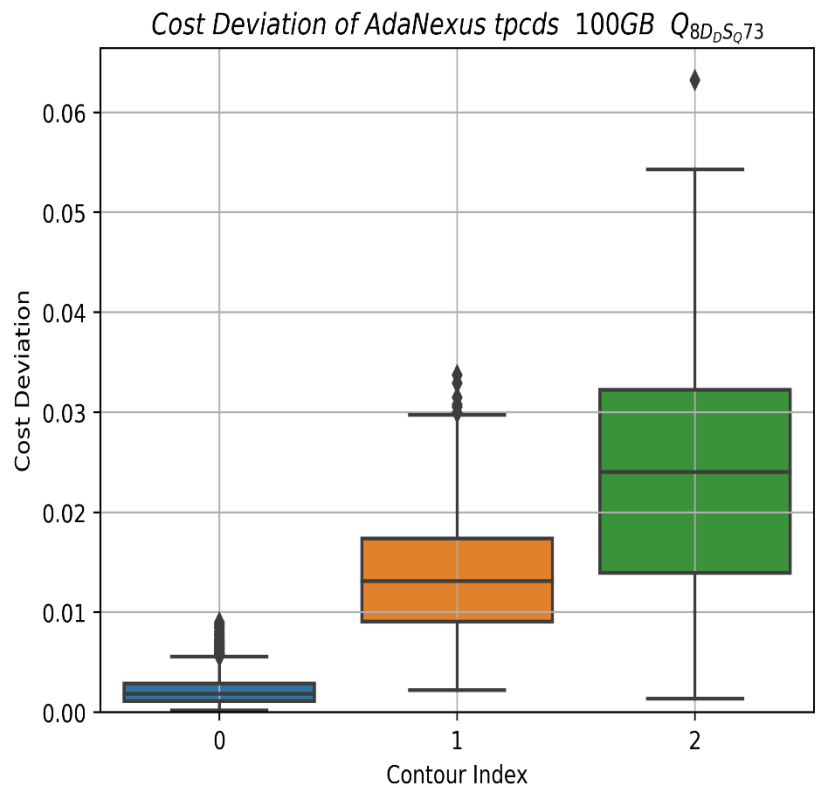
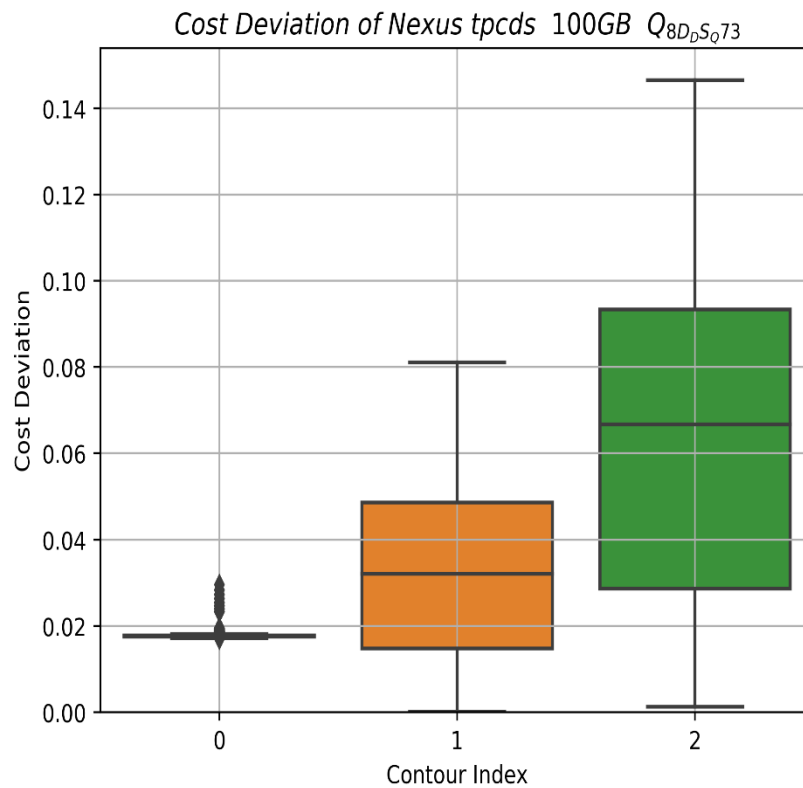
- ▶ DB engine is modified version of Postgres-9.4
- ▶ Intel® Core™ i9 CPU, 32GB 2666MHz RAM, 2TB Storage

Lesser Optimizer Calls



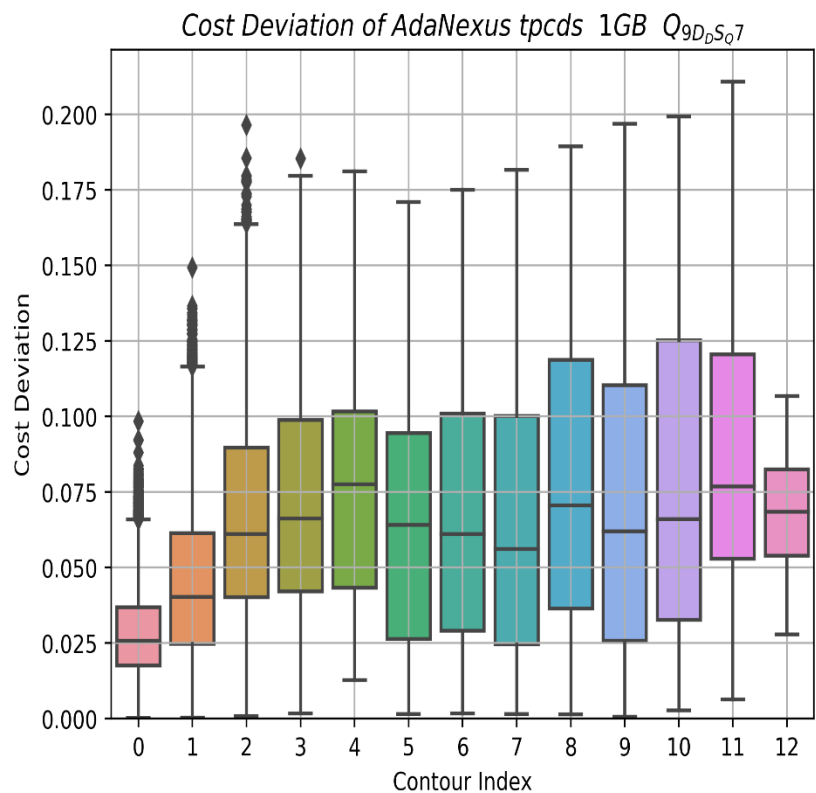
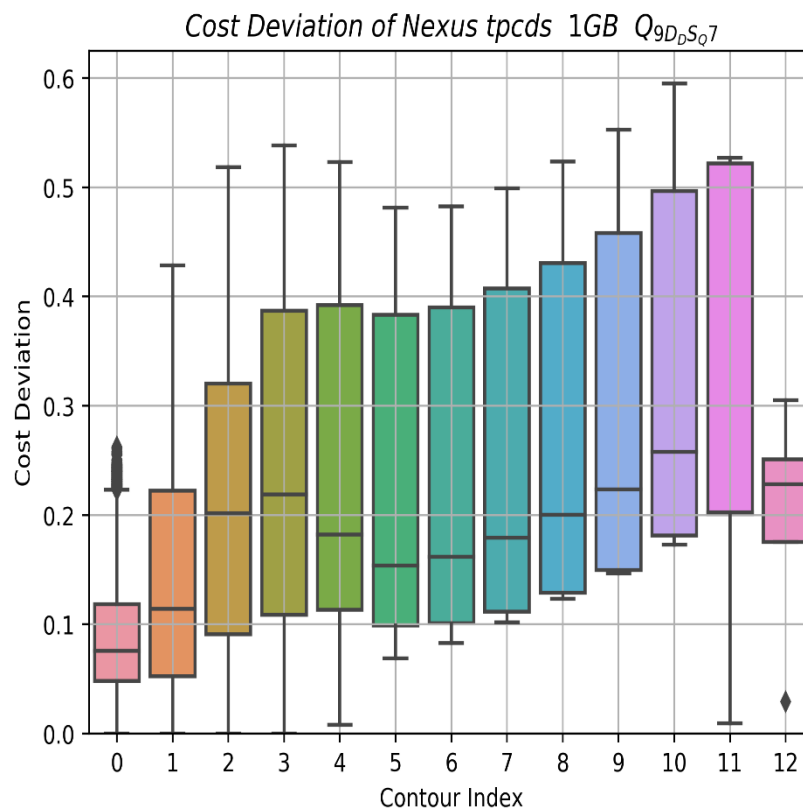
Reduced Cost Deviation

Query 73 100GB instance

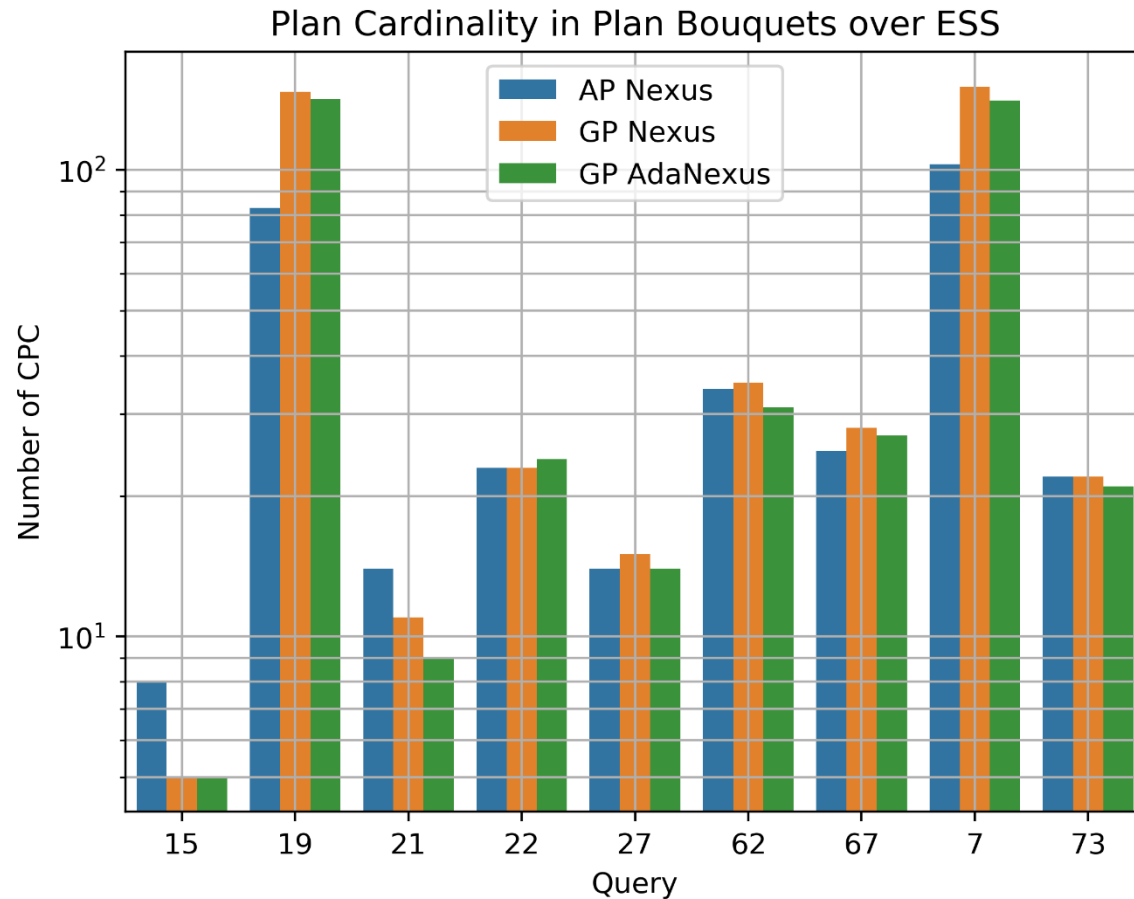


Reduced Cost Deviation

Query 7 1GB instance



Plan Cardinalities



Conclusion & Future Work

- Discretization of ESS is now removed in AdaNEXUS.
- Exponentially varying step size leads to bounded cost deviation.
- Speed up in contour discovery is obtained (with lesser cost deviation) by using slope information of piece-wise linear contours
- Adjacent sub-problems can share information for further speed-up for constructing a contour.
- Direction of contour discovery can be tuned using a full learnable PID controller or any (robust and interpretable) machine learning model.



Thank You!



*Slide Template Source: ©Department of Computational and Data Science, IISc, 2016
This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/)
Copyright for external content used with attribution is retained by their original authors*