AdaNexus: An Improved Nexus Algorithm

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Master of Technology

IN

Faculty of Engineering

BY

Achint Chaudhary

Computer Science and Automation

Indian Institute of Science

Bangalore – 560 012 (INDIA)

July, 2020

Declaration of Originality

I, **Achint Chaudhary**, with SR No. **04-04-00-10-42-18-1-15879** hereby declare that the material presented in the thesis titled

AdaNexus: An Improved Nexus Algorithm

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2018-20**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Advisor Signature

© Achint Chaudhary
July, 2020
All rights reserved

DEDICATED TO

My Parents & Close People

For those whom I have lost, wish you can see this

Acknowledgements

I am deeply grateful to Prof. Jayant R. Haritsa for his unmatched guidance. He was and always will be an inspiration to me for his thoughts and his perception of looking at things around us, be it academia or real life. It is my good luck that I got a chance to work with him and obtain few chunks of knowledge from him.

I am thankful to Anshuman Dutt, Srinivas Karthik and Sanket Purandare for their guidance and discussions. I am thankful to my lab mates for providing me with valuable suggestions during this work. My sincere thanks goes to my friends Diksha and Hemant, who have patiently listened and understood many complications and helped me bring out clearer pictures to work with. My heartfelt appreciation goes to CSA and CDS office staff who made the academic functioning smooth with best co-ordination possible.

Finally, I would express my gratitude wholeheartedly to my parents, who have taught me how to stay calm and keep working on what is essential. Nothing I have done or will ever do will match their support that I carry within.

Abstract

Declarative query processing in database systems often leads to sub-optimal performance due to wrong selectivity estimation from those encountered during actual execution. Plan Bouquets is a technique proposed to substitute selectivity estimation by selectivity discovery at run-time, to provide worst case performance guarantees. This is done by performing multiple partial executions for same query in an incremental fashion of cost budget from a bouquet which is compiled at very first stage.

This technique is suitable for OLAP queries as high overheads of optimizing most part of selectivity error space are amortized over multiple invocation of query in OLAP scenarios. Full space exploration overheads in the past are improved upon with NEXUS, is an algorithm, that only discovers points useful for bouquet compilation.

In this work, we proposed an adaptive version of NEXUS named AdaNEXUS, which utilizes geometrical properties of contours to be discovered for bouquet. This algorithm reduces overheads of compilations empirically keeping the worst case performance complexity same. Further, we provide upper bounds for maximum cost deviation possible during bouquet compilation due to use of either NEXUS or AdaNEXUS. Evaluation of proposed system is done on TPC-DS benchmark with different scales to test system. It is demonstrated that around an order of magnitude reduction is observed in compilation overheads when compared with NEXUS. Also, quality of plans discovered by AdaNEXUS is better than those discovered by NEXUS.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

SQL query processing is declarative in nature, user only specifies what needs to be done where, how it will be done is role of underlying system. There are a large number of execution strategies, each called a query plan. All these query plans will yield same results but have high variations in running times. Role of database optimizer is to select optimal (in terms of running cost) query plan. During choice of optimal query plan, database optimizer makes multiple cost based decision to compare different query plans. Cost for each physical operator in a query plan is a function of number of tuple, it processes, known as cardinality. Cardinality normalized in range of $[0, 1]$ is known as selectivity throughout literature. Selectivity estimations of optimizer for identifying the optimal query plan are done using statistical models and meta-data information about schema. Selectivity estimates are often sub-optimal which results in inflated query response time.

1.1 Background

There are multiple techniques proposed in literature to improve quality of selectivity estimates like better statistical models, on-the-fly re-optimization, etc., but none of them provides bounds on worst case performance guarantees.

An entirely different approach based on run-time selectivity discovery is proposed called *plan bouquets*, which for the first time, provides strong theoretical bounds on worst-case performance as compared to oracular optimal performance possible from all the available plan choices.

For each given query, predicates prone to selectivity error contribute as dimension in *Error-prone Selectivity Space(ESS)*. ESS is a multi-dimensional hyper-cube. The set of optimal plans over the entire range of selectivity values in ESS is called *Parametric Optimal Set of Plans(POSP)*. POSP is generated by asking optimizer's chosen plans at various selectivity locations in ESS using *selectivity injection module.* Cost surface generated over entire ESS is called *Optimal Cost Surface(OCS).* An *Iso-cost Surface(IC)* is a collection of all points from OCS which have same cost of optimal plan at each of these locations cost.

1.2 Motivation

Compilation of plan bouquet is process of drawing iso-cost contours, which involves getting selectivity points and corresponding optimal plans at each point for any contour we are drawing. In experimental setting ESS is discretized at some resolution 'res' where let 'd' be the dimension of ESS. Number of optimizer calls in entire ESS can reach res^d. This number is exponential in number of dimensions and results in high overheads of bouquet compilation. NEXUS is an algorithm developed in past to avoid doing optimizer calls on entire ESS, it does so by making optimizer calls only on points lying on contours. Number of optimizer calls made by NEXUS to draw 'm' contours in worst case is twice of $m * res^{(d-1)}$, which is still exponential in nature.

When 'm' is sufficiently high and 'd' is also high, for keeping things computationally feasible a moderate choice of 'res' is made. When this is the case, contours drawn by NEXUS can suffer from cost deviation from ideal desired cost value which does affect worst case performance guarantees. So it is desired to keep total overhead feasible with acceptable contour cost deviations.

1.3 Contributions

In this work, we have devised algorithms to improve contour discovery for plan bouquet, which constitutes most in compilation overhead. We have con

- **Speeding-up contour discovery:** We proposed AdaNEXUS algorithm, which is an improvement over NEXUS, that utilizes geometric properties of iso-cost contours generally observed in practice. This algorithm is designed to reduce total overheads in terms of optimizer calls and also to keep low contour cost deviations then what is there with NEXUS algorithm.

- **Contour cost deviation bounds:** We provide upper bounds on contour cost deviation values for both NEXUS and AdaNEXYS, from which it will be clear that AdaNEXUS should be preferred over NEXUS.

1.4 Organization

Chapter 2 provides a brief detail on Plan Bouquets technique. Chapter 3 discusses existing bouquet compilation techniques. Chapter 4 is about leveraging geometric properties of OCS and iso-cost contours to improve upon NEXUS and come up with AdaNEXUS algorithm. Worst case cost deviation bounds for both NEXUS and AdaNEXUS are given in Chapter 5. Experimental evaluation of our work is given in Chapter 6. At last, Chapter 7 discussed conclusion of our work and future work that can be done further.

Chapter 2

Plan Bouquets

Basics of Plan Bouquets from cite1 are given in this chapter which is an approach for robust query processing.

2.1 Overview

Plan Bouquet is a approach where compile time selectivity estimation is eschewed by systematic discovery of selectivity values at run-time by multiple partial execution carried in incremental cost-budgeted manner from a subset of POSP called Plan Bouquet.

A subset of POSP is identified as *Plan bouquet*, which is obtained by the intersection of plan trajectories with OCS, creating multiple Iso-cost surfaces, each of which is placed at some cost-ratio (r_{pb}) from the previous surface. Following Fig 1.[8] depicts an exemplar OCS and its intersection with IC trajectories for a sample 2-Dimensional ESS.

2.2 MSO_g due to Plan Bouquet

Since each plan on an iso-cost surface has a bounded execution limit, and incurred cost by execution using bouquet will form geometric progression. The figure below shows the performance of 1D plan bouquet w.r.t to optimal oracular performance.

In the above Fig 2. [1], various plans up to actual selectivity value q_a are executed. Each plan has a limit provided by the next iso-cost surface. This yields total execution cost of

$$C_{bouquet} = \sum_{i=1}^{k} cost(IC_i) \; a + a * r_{pb} + a * r_{pb}^2 + ... + +a * r_{pb}^{k-1} = \frac{a*(r_{pb}^k - 1)}{r_{pb} - 1}$$

This leads to sub-optimality (ratio of incurred cost to optimal cost) of plan bouquets approach as

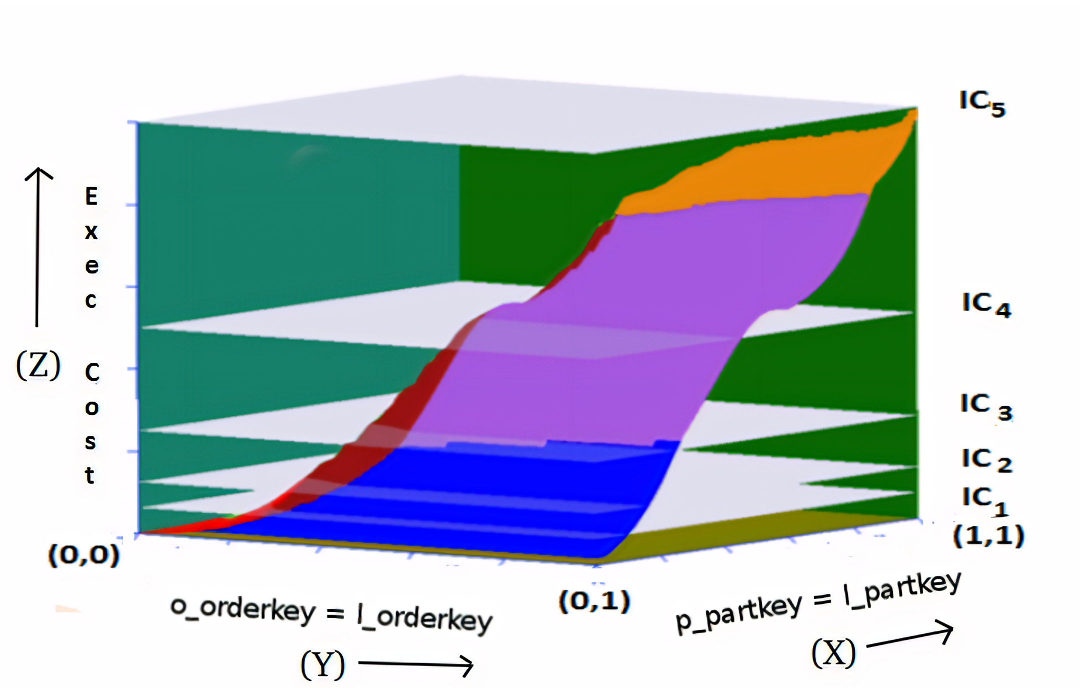


Figure 2.1: OCS and Plan Trajectories intersection

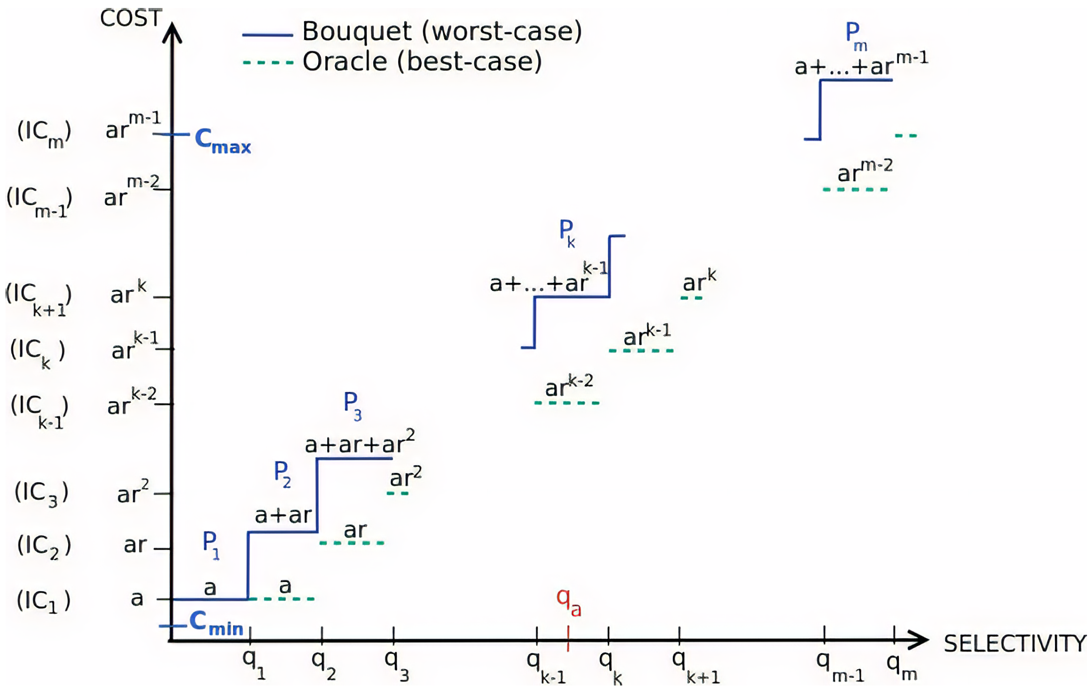


Figure 2.2: Cost incurred (Oracular vs Bouquet)

$$SubOpt(*, q_a) \leq \frac{\frac{a*(r_{pb}^2-1)}{r_{pb}-1}}{a*r_{pb}^{k-2}} \frac{r_{pb}^2}{r_{pb}-1} - \frac{r_{pb}^{2-k}}{r_{pb}-1} \leq \frac{r_{pb}^2}{r_{pb}-1}$$

This value is minimized using $r_{pb} = 2$, which provides theoretical worst case bound of 4 times the optimal execution time.

Extending the same idea to multiple dimensional ESS, MSO guarantee will become 4ρ , where ρ is maximum cardinality (of plans) on any of iso-cost surface.

Computing value of ρ requires huge compile time effort. Also, it is platform dependent and low value of ρ is desired for practical MSO_g which was obtained using anorexic reduction heuristic at the time plan bouquets was developed.

Later an improved algorithm called *SpillBound*[2], which is able to provide performance guarantee based only on query inspection and is quadratic function in number of error-prone predicates, which is same as dimensionality of ESS. MSO guarantee obtained by SpillBound is

$$D^2 + 3D$$

It is notable that Spillbound provides pre-compilation guarantees independent of ρ, which is platform dependent.

2.3 Impact of Cost Deviation on MSO_g

Let

Chapter 3

Problem Formulation

3.1 Notations

Notation	Description
SP	Selectivity Predicates
WKP	Well Known Predicates
EPP	Error Prone Predicates
ESS	EPP Selectivity Space
OCS	Optimal Cost Surface
$POSP$	Parametric Optimal Set of Plans
RES	Resolution of Discretized ESS
Dim or D	Dimensions of ESS
ϵ_i	Minimum Selectivity of Predicate SP_i
m	Number of Iso-cost Contours
IC_i	i_{th} Iso-cost contour
CC_i	Cost budget of IC_i
r_{pb}	Cost Ratio of Iso-cost contours
$(0, 1]$ or $[\epsilon, 1]$	Selectivity interval of Discretized ESS
P_j	Plan with assigned identity j
F_j	Plan Cost Function(PCF) for Plan P_j
$Cost(P, q)$	Cost of plan P at location q in ESS
d_{sel}	Uniform spacing of selectivity in ESS
r_{sel}	Ratio of selectivity values in ESS
β_{max}	Worst case slope of Plan Cost Function
α	Tolerance of contour thickening

3.2 Assumptions

3.2.1 Plan Cost Monotonicity(PCM)

This assumption implies that if location q_j spatially dominates location q_i in ESS, cost of optimal plan at location q_j is more than cost of optimal plan at location q_i.

$$(q_j \succ q_i) \Rightarrow (Cost(q_j) > Cost(q_i))$$

This also comes from a simple fact that processing more tuples will incur more cost. We assume that Plan Cost Functions and OCS are continuous and smooth in nature.

3.2.2 Axis Parallel Concavity(APC)

This assumption, as stated in [3], is on Plan Cost Function (F_p) which is not just monotonic but exhibits a weak form of *concavity* in their cost trajectories. For 1D ESS, F_p is said to be concave if for any two selectivity locations q_i, q_j from ESS and any $\theta \in [0, 1]$ following condition holds

$$F_p(\theta * q_i + (1 - \theta) * q_j) \geq \theta * q_i + (1 - \theta) * q_j$$

Generalizing to D dimensions, a PCF F_p is said to be *axis parallel concave* (APL) if the function is concave along every axisparallel 1D segment of ESS. It simply states that each PCF should be concave along every vertical and horizontal line in the ESS. Further, an important and easily provable implication of the PCF exhibiting APC is that the corresponding *Optimal Cost Surface*(OCS) which is the infimum of the PCFs, also satisfies APC. Finally, for ease of presentation we will generically use concavity to denote APC in the remainder of this work.

3.2.3 Bounded Cost Growth(BCG)

BCG property as defined by [4], is as follows for plan cost function F_p.

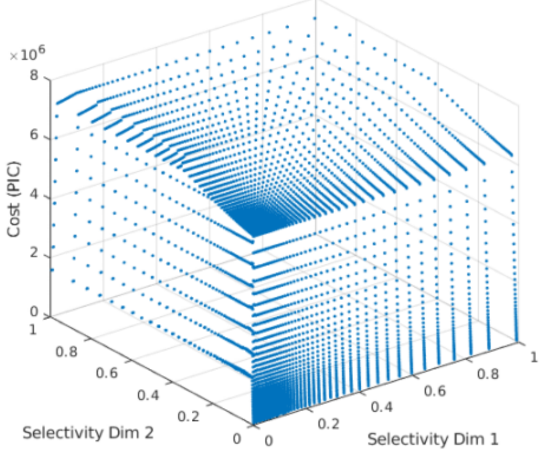
Here, $f(\alpha)$ is an increasing function. Increase in selectivity by $\alpha \geq 1$ will result in maximum cost increase by a factor of $f(\alpha)$. As in the case of APC assumption, BCG is also proven to hold for OCS when it is true for all POSP plan cost functions.

$$F_p(\alpha * q.j) \leq f(\alpha) * F_p(q.j)$$
$$\forall j \in 1, 2, ..., D \wedge \forall \alpha \geq 1$$

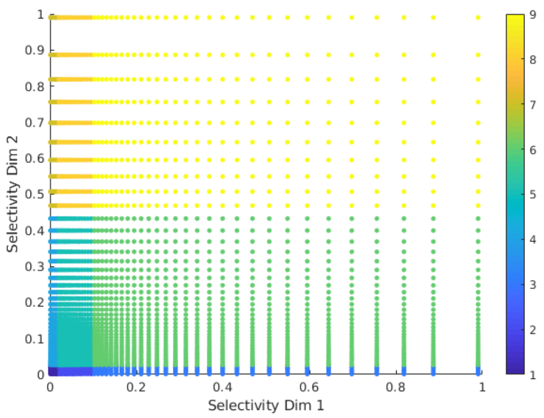
They have also claimed that identity function $f(\alpha) = \alpha$ suffices in practice.

3.2.4 Piece-wise Axis Parallel Linearity(APL)

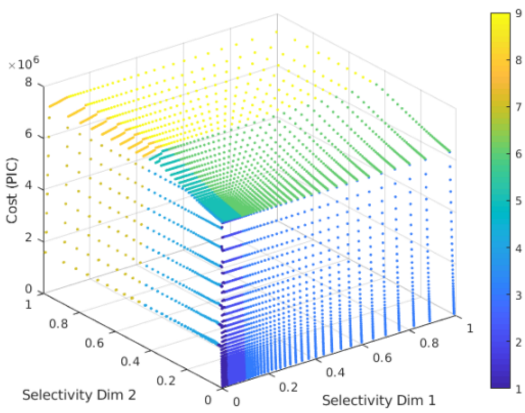
Plan Cost Functions and OCS are shown to be piece-wise linear in [5]. This property commonly comes from the fact that partial derivatives of common physical operators (except the sort operator, which is seldom found in industry strength benchmark [4]) are linear in nature.



(a) Original OCS



(b) Partitioned OCS Domain



(c) OCS fitted with Piecewise functions

Fig 3. Multiple APL functions to fit OCS

When it is the case that OCS or Plan Cost Functions are not truly piece-wise linear, a coarse approximation of piece-wise linear function can still be fitted to them. Similar work has been done in our lab in the past [6].

While in our work, there is no need to fit any such piece-wise linear function. This will reduce our effort of fitting points from entire OCS into a piece-wise APL function which will itself be exponential in nature.

3.2.5 Perfect Cost Model of Optimizer

This assumption states that poor choices of plan come only from the cardinality estimation error of optimizer and not from the cost model itself. While we have assumed perfect cost model of optimizer, an optimizer with bounded cost model will also work well. Improving the cost model is an orthogonal problem. One work on offline tuning [7] proves that the cost model can be tuned to predict value within 30% of the estimated cost values.

3.2.6 Selectivity Independence

We assume that selectivity of predicates is independent of each other. While this is a common assumption in query optimization literature, it often does not hold in practice.

3.3 Existing Compilation Strategies

3.4 Overheads of existing methods

3.5 Problem Statement

The first chapter should have the motivation of the problem and the brief description of the solutions provided in the thesis. There should be one outline section which should give detailed description of the thesis chapters. Here is an example citation, [**?**].

Chapter 4

Leveraging Contour Geometry

4.1 How to Write a Thesis: An Introduction

The first chapter should have the motivation of the problem and the brief description of the solutions provided in the thesis. There should be one outline section which should give detailed description of the thesis chapters. Here is an example citation, [**?**].

Chapter 5

Cost Deviation Bounds

5.1 How to Write a Thesis: An Introduction

The first chapter should have the motivation of the problem and the brief description of the solutions provided in the thesis. There should be one outline section which should give detailed description of the thesis chapters. Here is an example citation, [**?**].

Chapter 6

Experimental Evaluation

Now we will move towards profiling the performance of our proposed methods over existing ones in literature. Overheads incurred during bouquet compilation are discussed in terms of optimizer calls, while quality of contours discovered is measured in terms of cost deviation from ideal contour cost.

Database environment: Queries for evaluation are taken from TPC-DS benchmark that covers wide spectrum of join geometries including *star, chain,* etc. Number of base relations varies from 3 to 6. Number of error prone predicates varies from 3 to 5, all of which are join selectivity errors. Since physical schema has indexes on all columns, which lead to more variety of plans possible, making it difficult to achieve robustness due to large ratio of C_{max}/C_{min}. TPC-DS benchmark is used at different scale, varying from 1GB to 250GB. Benchmark metadata is scaled (keeping the same distribution) for each scale using CODD[13]
System environment: Database engine used in our experiments is modified version of Postgres-9.4. Hardware platform is vanilla HP-Z4-G4 workstation with Intel Core i9-7900X CPU, 32GB DDR4 2666MHz RAM and 2TB disk storage. Now we will compare different frameworks of bouquet compilation and also our devised algorithms over existing algorithms like NEXUS.

6.1 BCG Validation

We have verified that bounded cost growth does hold in all our queries on different scales of database instances we have experimented upon. To find out β_{max}, we did optimizer calls in close neighborhood of all end points of ESS, this took total cost of

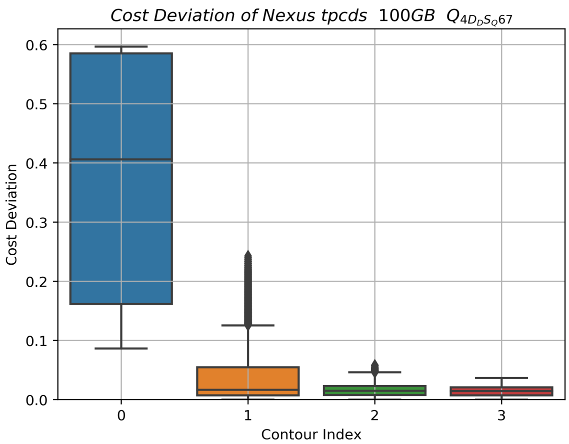
$$\Theta(2^{(2*Dim)})$$

We have observed that claim of [4] does hold, and β_{max} is bounded by 1.0 in all our experiments.

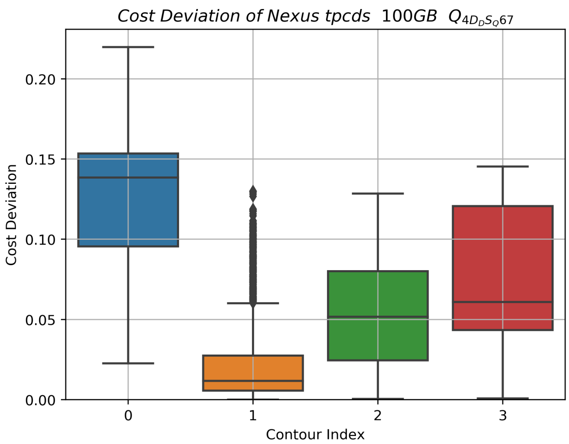
This is due to the fact that contribution from Sort is very less compared to the total cost of a plan in industrial strength benchmarks[4]. This bound on β_{max} is then used to calculate cost deviation bound during contour discovery.

6.2 Uniform vs Exponential Distribution

We have evaluated both uniform distribution and exponential distribution of selectivity values to discretize ESS before applying NEXUS with a moderate resolution to make compilation time feasible (few hours for 5 dimensional queries), cost deviation values (α) for each contour for a Q67 using uniform and exponential distribution is given in Fig 7.



(a) Uniform distribution



(b) Exponential distribution

Fig 7. Comparison of Cost deviation for Q67 compilation

It is notable that $(1 + \alpha) \leq r_{pb}$ and uniform distribution will lead to high value of α. For some queries in our test suite, we have observed that this assertion of $(1 + \alpha) \leq r_{pb}$ is not respected.

Hence we can conclude that uniformly spaced selectivity values should not be used even for discretization of ESS with moderately high resolution.

6.3 Tuning exponential smoothing

We have given an expression for step-size adaptive exponential smoothing at end of section 3.5 where α is constrained in interval (0, 1). We have empirically observed that α=0.7 gives best results during our experiments. In the further experiments we have used the same value as smoothing constant. Fig 8. gives an idea about different values of smoothing constant and present step-size on contribution of latest direction vector towards tuned vector.

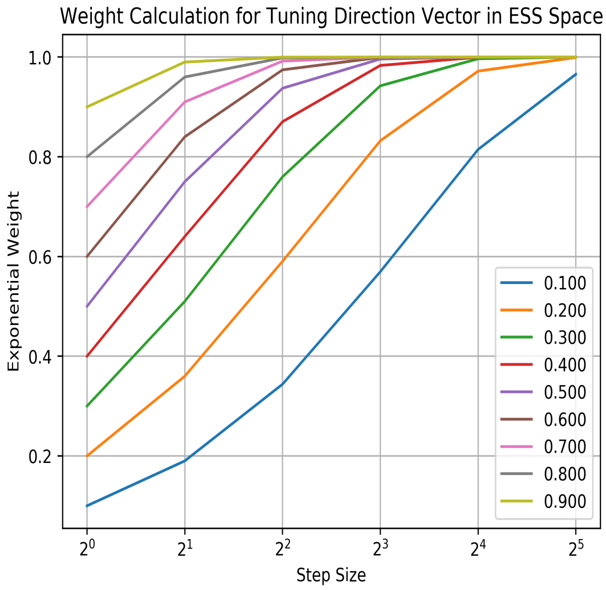
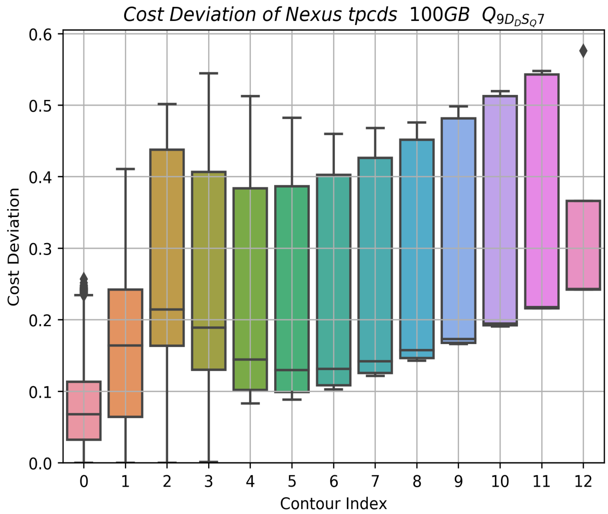


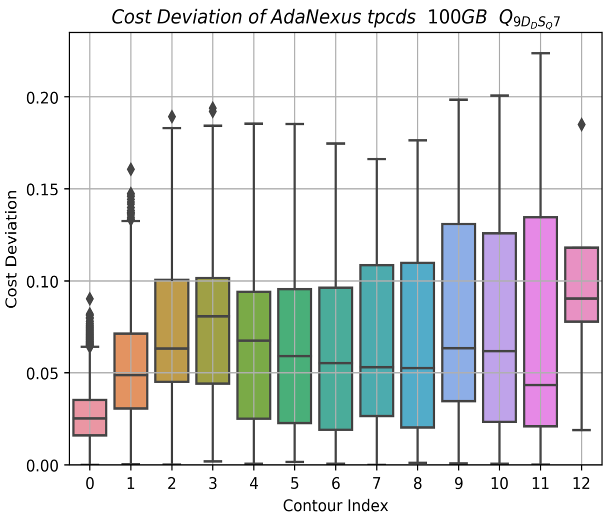
Fig 8. Step-size and α impact on tuning

6.4 Deviation bounds due to Exponential Distribution

In the end of section 3.3, we have come up with cost deviation bounds when using either exponential discretization with NEXUS or exponential step size into continuous ESS space for AdaNEXUS with an example of Q7 which is a 4-dimensional query. Below Fig 9. shows contour-wise cost deviation distribution for NEXUS and AdaNEXUS respectively.



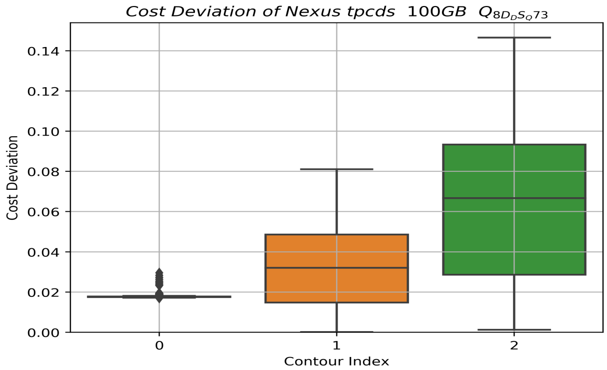
(a) NEXUS with exponentially discretized ESS



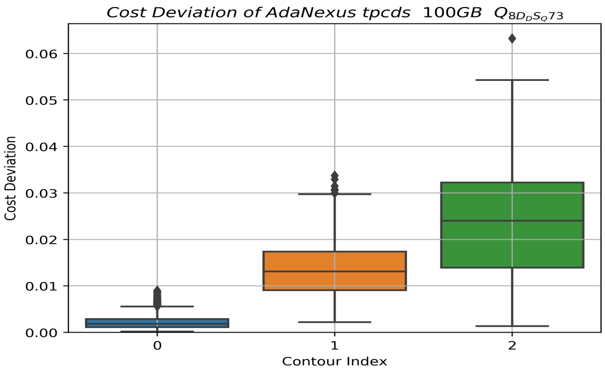
(b) AdaNEXUS with exponential steps in ESS

Fig 9. Cost deviation for NEXUS vs AdaNEXUS

This behavior of cost deviation reduction is not just observed in queries where NEXUS has high cost deviation, but also on queries having low deviation with NEXUS. As an example we will see Q73.



(a) NEXUS with exponentially discretized ESS



(b) AdaNEXUS with exponential steps in ESS

Fig 10. Cost deviation for NEXUS vs AdaNEXUS

We have observed in all our experiments that cost deviation bounds and empirically observed deviation are lesser for AdaNEXUS as compared to those obtained using NEXUS. These deviations in contour cost have a directly proportional impact on MSO_e.

6.5 Overhead reduction using AdaNEXUS

The main role of AdaNEXUS is to reduce overhead in compilation process, where reduced cost deviation is just a plus point of AdaNEXUS design. Now we will see a comparison of overheads incurred using different bouquet compilation approaches.

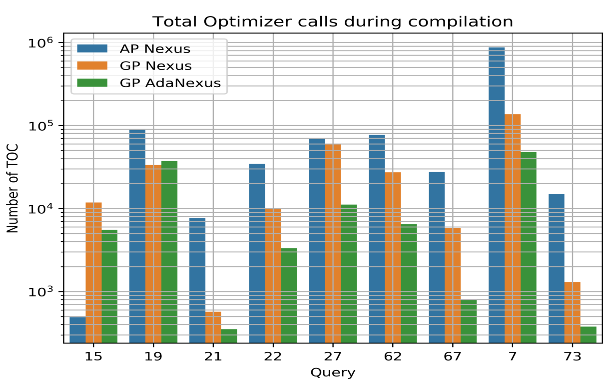


Fig 11. Overheads using different compilation approaches

AdaNEXUS brings approximately an order of magnitude reduction in compilation efforts even with moderate minimum step sizes. Also as a bonus, it reduces deviation observed in contour discovery.

An important note here is that *AP NEXUS*, the execution of NEXUS with uniformly spaced selectivity values, results in higher cost deviations that makes it useless for plan bouquets, specially in higher dimensions.

6.6 Plan Cardinalities

It is not claimed that AdaNEXUS along with binary exploration will find out all plans on contours, that were otherwise observed using NEXUS.

So, we have observed number of plans for all queries and reported the same using both uniform and exponential distribution in NEXUS and AdaNEXUS.

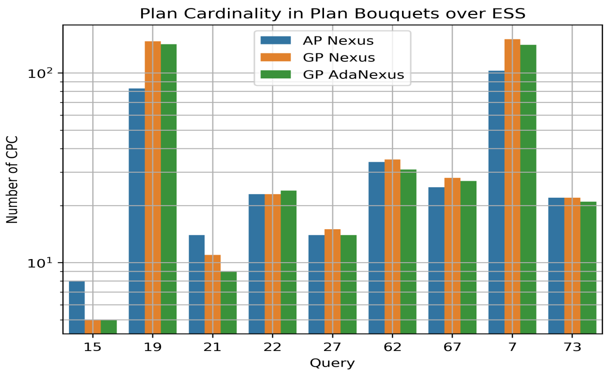


Fig 12. Total contour plan cardinalities

Note that we have calculated each plan on different contour multiple times, so above picture can be better understood in terms of number of executions on all contours. In general AdaNEXUS finds out almost all executions discoverable by NEXUS with geometric progression.

So, an empirical claim can be made that almost all plans on each contour discoverable by NEXUS can be discovered also with AdaNEXUS.

Additionally now either a weighted greedy algorithm for anorexic reduction can be deployed before plan bouquet execution, or a contour density independent algorithm like *SpillBound* can utilize AdaNEXUS as a better substitute to NEXUS.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Firstly, we have proved (with an additional assumption of Bounded Cost Growth) that exponential distribution in ESS discretization leads to usable cost deviation in contour discovery. We have given upper bounds on cost deviations and validated that they are followed in all our test queries.
Next, we have devised an improvement over NEXUS with complete removal of explicit ESS discretization which leads to contour discovery with lesser cost deviations and also with lesser optimizer calls.

7.2 Future Work

Both NEXUS and AdaNEXUS use 2-dimensional seed exploration as a subroutine. This way solution to multi dimensional contour discovery is obtained by merging the results of multiple low dimensional sub-problems in a recursive manner. Solutions to all these sub-problems do not share any information with each other e.g. related to slope of adjacent sub-problems being solved. So, AdaNEXUS can gain further speed-up using information shared by related sub-problems already solved. Hence, contour discovery can gain speed with a dynamic programming style solution instead of a simple design and conquer based search.
Tuning during contour discovery can be improved using a full PID control with parameter tuning using machine learning techniques (as parameter tuning is crucial in PID, even when they are suitable for optimizing linear processes). Also, some full-fledged machine learning algorithms (preferably robust and interpretable) can be used to speed up contour discovery process.
Given proper framework to work on discretization of ESS with exponential distribution, in-

cremental algorithms can be devised in conjunction with AdaNEXUS, as additional work in database scale-up is expected to be sub-linear in scale-up.

Bibliography

[1] Anshuman Dutt and Jayant R. Haritsa. *Plan bouquets: A fragrant approach to robust query processing*. In ACM Trans. on Database Systems (TODS), 41(2), pages 1–37, 2016. 4

[2] Srinivas Karthik, Jayant R. Haritsa, Sreyash Kenkre and Vinayaka D. Pandit. *Platform-independent Robust Query Processing*. In Proc. of the 32nd Intl. Conf. on Data Engg., ICDE '16, pages 325-336, 2016. 6

[3] Srinivas Karthik, Jayant R. Haritsa, Sreyash Kenkre and Vinayaka D. Pandit. *A Concave Path to Low-overhead Robust Query Processing*. In Proc. of the VLDB Endow., 11(13), pages 2183-2195, 2018. 9

[4] Anshuman Dutt, Vivek Narasayya, and Surajit Chaudhuri. *Leveraging re-costing for online optimization of parameterized queries with guarantees*. In Proc. of the 2017 ACM SIGMOD Intl. Conf., pages 1539–1554, 2017. 9, 10, 14, 15

[5] Arvind Hulgeri and S. Sudarshan. *Parametric query optimization for linear and piecewise linear cost functions*. In Proc. of the 28th Intl. Conf. on Very Large Data Bases, VLDB '02, pages 167–178, 2002 10

[6] Sanket Purandare, Srinivas Karthik and Jayant R. Haritsa. *Dimensionality Reduction Techniques for Robust Query Processing*. Technical Report TR-2018-02, DSL CDS/CSA, IISc, 2018. DSL Website 11

[7] Wu, Yun Chi, Shenghuo Zhu, Junichi Tatemura, Hakan Hacigumus, and Jeffrey F. Naughton. *Predicting query execution time: Are optimizer cost models really unusable?*. In Proc. of the 29th IEEE Intl. Conf. on Data Engg., ICDE '13, pages 1081– 1092, 2013. 11

[8] Srinivas Karthik V. *Geometric Search Techniques for Provably Robust Query Processing*. PhD thesis, Indian Institute of Science Bangalore, December 2019. 4

[9] D. Harish, Pooja N. Darera, and Jayant R. Haritsa. *On the production of anorexic plan diagrams*. In Proc. of the 33rd Intl. Conf. on Very Large Data Bases (VLDB'07). 1081–1092.

[10] Guido Moerkotte, Thomas Neumann, Gabriele Steidl. *Preventing Bad Plans by Bounding the Impact of Cardinality Estimation Errors*. In Proc. of the VLDB Endow., 11(13), pages 2183-2195, 2009.

[11] Wikipedia contributors. (2020, March 12). *PID controller*. In Wikipedia, The Free Encyclopedia. Retrieved 09:25, April 11, 2020.

[12] Anshuman Dutt. *Plan bouquets: An Exploratory Approach to Robust Query Processing.* PhD thesis, Indian Institute of Science Bangalore, August 2016..

[13] Ashoke S. and J. Haritsa. *CODD: A Dataless Approach to Big Data Testing* PVLDB Journal, 8(12), August 2015. 14