**Plan Bouquet based Techniques for Variable Sized Databases**

Achint Chaudhary

February 2, 2020

MTech Project Report (CSA)

**0 Abstract**

OLAP applications often require a certain set of canned queries to be fired on database with the varying constants in queries. For optimal execution of these queries, query optimizer does select a strategy known as the query execution plan. These choices are based on cardinality estimates of various predicates that often hugely differ from actual cardinality values encountered during execution. Due to this reason, optimizer choice leads to high inflation in actual execution cost as compared to predicted cost during optimization.  
  
An altogether different approach for query processing was proposed in 2014, named Plan Bouquet. Basis of which is selectivity discovery at run-time by repeated cost bounded execution of carefully chosen to set of plans. This technique provides strong bounds independent of data distribution.

However, Plan Bouquet on cost sub-optimality is not designed to be robust against large updates in the database. This work focuses on observing limits up to which size of database can be increased without serious deterioration in performance guarantee. Also, we will provide incremental algorithms that can use information from plan bouquet compiled in past and extend it, to provide further robust execution without incurring overhead of re-compiling entire plan bouquet.

**Sec 1 Introduction**

Database query optimizer chooses a plan covering various structural choices of logical and physical operators for query execution. These choices are based on the cost of each operator which is calculated using number of tuples it will process known as cardinality. Cardinality normalized in range of [0, 1] is known as selectivity throughout our analysis.

These selectivity values are estimated before query execution based on some statistical models used in classical cost-based optimizers. An entirely different approach based on run-time selectivity discovery is proposed called Plan Bouquet, which provides for the first time strong theoretical bounds on worst-case performance as compared to optimal performance possible from all the available plan choices.

For each given query, predicates having the potential of selectivity error contribute as a dimension in Error-prone Selectivity Space (ESS). 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 chose plans at various selectivity locations in ESS using Selectivity injection module. Cost surface generated over entire ESS is called Optimal Cost Surfaces (OCS). An Iso-cost surface is collection of all points from OCS which have same optimal plan cost at that location.

A subset of POSP is identified as Plan bouquet, which is obtained by the intersection of plans trajectory with OCS, creating multiple Iso-cost surfaces, each of which is placed at some ratio proportion (r\_pb) of cost from the previous surface.

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

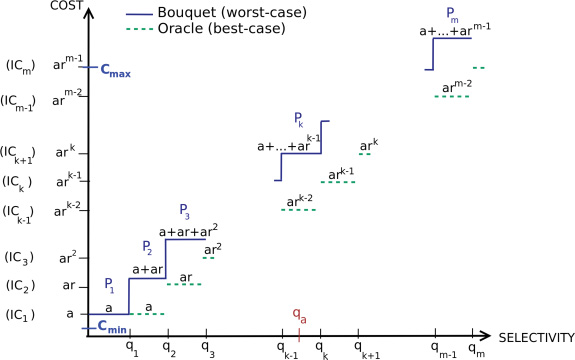


Fig 1. Cost incurred by Oracular vs Bouquet Execution

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

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 4\*p, where p is maximum cardinality of plans on any of iso-cost surface. Since computing value of p will need huge compile time effort, it is platform dependent and also low value of p is desired, which is dependent & obtained using Anorexic reduction heuristic at the time plan bouquet was developed.

Later a improved algorithm called SpillBound is invented, 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 by Spillbound is

D\*\*2+3\*D

We will be using Spillbound for our work, as it provides pre-compilation performance guarantee

**2. PROBLEM FORMULATION**

2.a Notations

2. b Assumptions

**Plan Cost Monotonicity (PCM)**

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

(b>a) 🡪 Cost(b) > Cost(a)

This also comes from a simple fact that processing more tuples will incur more cost

**Axis Parallel Concave (APC)**

This assumption is on Plan Cost Function (PCF) which is not just monotonic but exhibit a weak form of concavity in their cost trajectories. For 1D ESS, PCF is said to be concave if for any two selectivities locations x1, x2 from ESS and any w in [0,1] below condition holds

F(w\*a + (1-w)\*b ) >= w\*F(a) + (1-w)\*F(b)

Generalizing to D dimensions, a PCF is said to be Axis Parallel Concave (APC) if the function is concave along every axis-parallel 1D segment of ESS.

Which 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 PCFs exhibiting APC is that the corresponding OCS, which is the infimum of the PCFs, also satisfies APC. Finally, for ease of presentation, we will generically use concavity to mean APC in the remainder of this work.

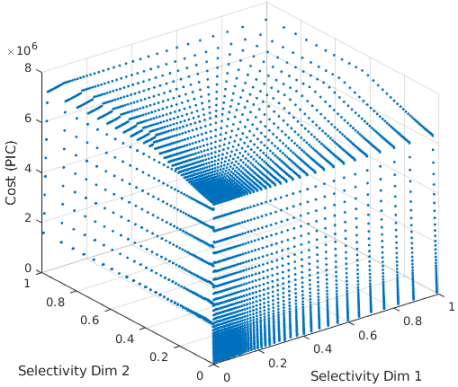
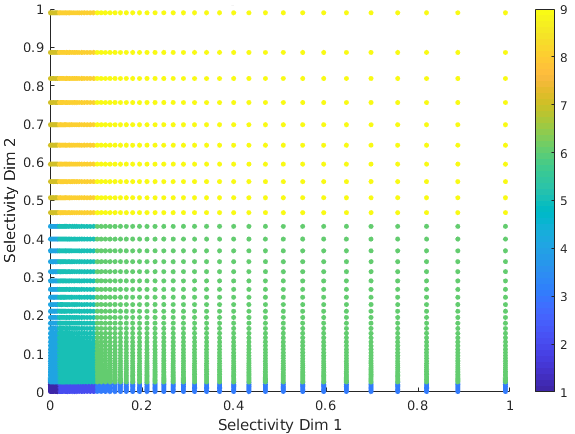
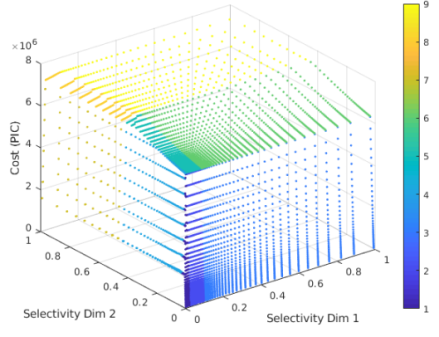
**Bounded Cost Growth (BCG)**

hello

**Piecewise Axis Parallel Linear (APL)**

Plan Cost functions & OCS are shown to be piecewise linear in [HS02]. 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) are linear in nature. We will use this property for design of various algorithms and improvements in existing methods.

Similar assumption is made by [SUDARSHAN], and have proved that in practice slope of plan cost functions and OCS are bounded in form of a polynomial function in chance of selectivity. They have claimed that identity function f(a)=a suffice in practice

When it is the case that OCS or Plan cost functions are not truly piecewise linear, a coarse approximation of piecewise linear function can still be fitted to them. Similar work is done in our lab also in past by [SANKET]. While we don’t explicitly need to fit any such piecewise linear function. It will reduce our effort of fitting points from entire OCS into a piecewise APL function

**EPP are the only predicates**

For present analysis, we have considered that all of query predicates are Error-prone, there is also no trivial predicate, which means each relation has some filter predicate applied over it, which will be also considered as error-prone in our conservative assumption.

Rational behind this assumption is that, if we supply same selectivity value to both old and new database, their outputs will be of different cardinalities. If we have not gone with this conservative assumption, we will need to determine change of scale for AKP and TP, and in that case, maybe re-compilation of bouquet may be needed. Removing this assumption and detecting is a problem we will later try to encounter

**Perfect Cost Model of Optimizer**

This assumption states that poor choices of plan come only from cardinality estimation error of optimizer and not from cost model itself. While, we have assumed perfect cost model of optimizer, an optimizer with bounded cost model will also work well. Improving, which is an orthogonal problem, one work on static tuning, proves most actual cost values to lie in 30% of estimated cost values after tuning.

2.c Performance Metrics

2.d Selectivity Intervals for study

Selectivity is the fraction of tuples out of maximum possible tuples that can come out of a query predicate. Notation of selectivity is devised to make study of ESS independent of cardinality values. This has motivated in past literature that selectivity value be always bounded in range [0,1].

Looking at type of changes possible in database, we first consider that Plan bouquet and techniques later developed on base of it are robust to data distribution, so only changes in distribution will not call for need of re-compilation since plan bouquet from past will provide same performance guarantee as it used to in some point of time. This is with the assumption that all tuple processing nodes are error prone

Considering increase in size of database, keeping same distribution as before. Number of tuples to process now has increased. So, if we go with [0,1] picture of selectivity again, same selectivity value will result in different cardinality. As query optimizers are cost based, cost model consider the cardinality and for sake of uniformity in picture of old and new database throughout our analysis should provide same cardinality provided same selectivity. In a loose sense, selectivity to cardinality mapping should not be changed.

Under this case, if a

**3. CHALLENGES IN HANDLING DATABASE UPDATE**

Plan bouquet is suitable for canned queries as compilation overhead of entire ESS enumeration will take O(RES\*\*Dim) cost, which is amortized over repeated invocations. Under situation of change of database size, placements of ideal contours can be totally different from existing contours built for old database size, which may result that old bouquet plans are totally different from new bouquet contours. Under above mentioned conditions it seems that a re-compilation will be needed.

Now we will look at how compilation (iso-cost surface identification) in past work is done, and why under all present options compilation at it first place is resource exhaustive.

Origin & Terminus are the extreme ends of ESS, at which optimal costs are obtained via getting optimal plan using selectivity injection. These cost values are called C\_min, C\_max respectively, using these two values, number of contours (m) is obtained as follows.

M = ceil( log( (C\_max/C\_min) , [base=r\_pb] ) ) + 1

These M iso-cost contours are to be drawn at r\_pb cost ratio successively. Now next part is to detect points from ESS & optimal plans at those locations, lying on each of the contour. For now there are two options available for contour construction Full ESS enumeration and NEXUS. Let’s see them one by one

Full discretized ESS Enumeration

This is a naïve approach, in which optimal plan and its cost at all points of ESS is asked from query optimizer. Points at which optimal plan’s cost is equal to any cost value of iso-cost contour is qualified to be added to that contour. This will incur O(RES\*\*Dim) cost. Where RES is resolution chosen to discretize ESS, while Dim is dimension of ESS. Each dimension in ESS represents a error-prone predicate.

This approach is certainly Exponential in Dim, and also a suitable value of RES should be chosen to make overall cost computationally feasible. Full space enumeration can totally exploit parallel architecture of modem multi-core systems available.

NEXUS

An optimization over full space enumeration is introduced with plan bouquet. NEXUS is an algorithm proposed to avoid making unnecessary optimizer calls to points lying in between contours. If we have total M iso-cost contours to discover, worst case complexity of NEXUS can go up to O( M\*D\*(RES\*\*(D-1)) )

At first NEXUS seems to be promising for reducing compilation overhead, but faces multiple issues (as mentioned by [SRINIVAS]):

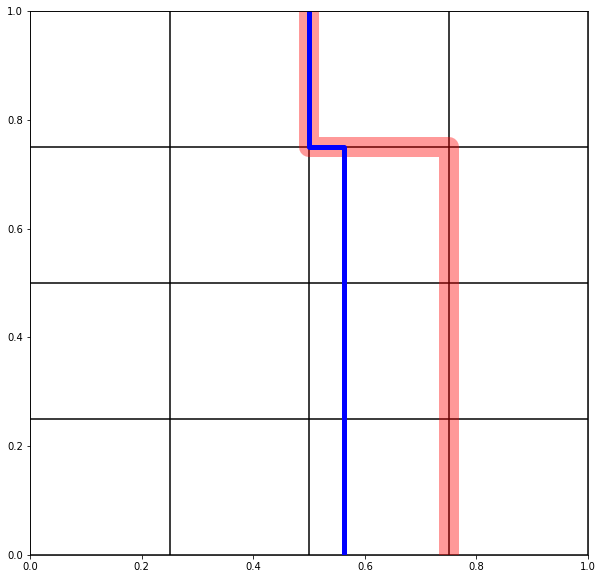
1. If large number of contours needs to be drawn, NEXUS is effectively close to Full space enumeration, especially in high dimensional ESS
2. If a lower bound is known on query selectivity, Spillbound is able to shrink ESS, while NEXUS needs redrawing of all contours from scratch

(3) Randomized contour placement to introduce fairness in Plan bouquet needs more contour need to be drawn. This makes NEXUS cumulatively more expensive than full space exploration

Also, both above methods of finding iso-cost contours make a common assumption that, Resolution of discretized ESS grid should be sufficiently high such that we can always find contiguous iso-cost locations even with small values of a, say, 0.05.

Next we will see, some issues which are common to both Full ESS exploration and NEXUS

ISSUE WITH USING LOW RESOLUTION FOR HIGH DIMENSIONAL ESS



To empirically avoid above explained issue, when working with low resolution and high dimensional ESS. To make things computationally feasible, geometric distribution of selectivity value is used on each axis.

This use of low resolution and geometric distribution is never explicitly stated in literature, and may violate MSO guarantees in practice. Which is not observed till yet, but proof for the same is also pending like a conjecture. Rational till now for use of Geometric distribution is that it captures many points in low selectivity values, and most changes in plan choices takes place in low selectivity values.

**4. OUR CONTRIBUTION**

Complexity of NEXUS & Scope of improvement (Reducing overall compilation efforts)

Using Geometric progression with ESS with lower bound

Efficient computation of Inflated MSO guarantee

Incremental bouquet maintenance algorithms

We will first reduce compilation overheads from what is done in past literature. Later will show efficient methods to determining empirical guarantees from using a old and extended plan bouquet, and at last provide incremental bouquet maintenance algorithms