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

Achint Chaudhary

MTech (CSA)

OLAP applications require a certain set of queries to be fired on databases with the change in certain constants in queries. For optimal execution of these queries, the query optimizer must select an optimal strategy known as the query execution plan. These choices are based on cardinality estimates of various predicates that differ from actual cardinality values encountered during execution. Due to this reason, we get bad choices of execution plans that led to high inflation in actual execution cost as compared to predicted during optimization phase.  
  
An altogether different approach for query processing is proposed in 2014, named Plan Bouquet. Basis of which is selectivity discovery at run-time by repeated execution of multiple plans. This technique provides strong bounds independent of data distribution.

Plan Bouquet on other hand is not robust against large updates in the database. This research work focuses on providing incremental algorithms that can use past information about plan bouquet and iso-cost contours, to provide further robust execution without incurring entire compilation overhead of plan bouquet.

**Sec.1 Introduction**

Optimal Plan from Optimizer comes from various structural choices of logical and physical operators for query execution. These choices are based on cost of each operator which is calculated using number of tuples it must process known as . Cardinality normalized is known as 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 known as Plan Bouquet, which provides for first time theoretical strong 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 contributes as a dimension in . The set of Optimal plans over the entire range of selectivity values in ESS is called . POSP is generated by asking optimizer's optimal plans for plans at various selectivity values using Selectivity injection module.

A subset of POSP is identified as Plan bouquet, which is obtained by the intersection of plans trajectory with

Iso-cost surfaces, each of which is placed at some ratio proportion of cost from the previous surface.

Since these executions form geometric progress, the total cost of which is can be also derived using the sum of geometric progression. The figure below shows the performance of bouquet w.r.t to optimal oracle performance.



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 leads to Sub-Optimality (ratio of incurred cost to optimal cost) of bouquet approach of

This value is minimized using , which provides theoretical worst case bound of 4 times the optimal execution time.

**Sec. 2 Notations, Notes & Assumptions**

**2.a Notations used in this work**

|  |  |
| --- | --- |
| **Notation** | **Description** |
|  | Query Predicate |
|  | Set of all Query Predicates |
|  | Set of Actually Known Predicates |
|  | Set of Error Prone Predicates |
|  | Set of Trivial predicates |
|  | EPP Selectivity Space |
|  | Dimension of ESS |
|  | Cost of Optimal Plan at selectivity |
|  | Scaling Factor of Predicate in Database |
|  | Minimum Selectivity on Predicate |
|  | Resolution on Each axis of ESS |
|  | Number of Iso-cost Contours |
|  | Cost Ratio of Iso-cost contours |
|  | Discretized interval for each axis of ESS |
|  | Set of Optimal Plans over entire ESS |
|  | Cost of plan P at selectivity location q in ESS |
|  | Cardinality of predicate p at location q with database size scale s |
|  | Difference in Selectivity values |
|  | Ratio of Selectivity values |
| β | Worst case slope of Plan Cost Function |
| α | Tolerance of contour thickening |

**2.b Notes**

**2.b.1 Interplay of Selectivity & Cardinality**

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.

**2.b.2 Distribution of Selectivity values on axis of ESS**

is will be discretized at finite resolution with number of points . Each axis denoting selectivity for each of predicate in will have points. Choice is to take these points in Arithmetic or Geometric progression in interval .

GP gives more focus on lower selectivities (near origin) since the plans and cost rapidly changes in that region of ESS. Choice of GP also helps in designing efficient incremental bouquet algorithm.

is minimum considered selectivity, sooner we will see bounds on this also under some assumptions we made.

**2.b.3 Trivial Predicates**

These are predicates never seen when dealing with a fixed size database, as they always have maximum selectivity of 1.0. Example of this is scanning of entire relation without any filter.

These predicates are useful when comparing old and new database with reference to each other.

**2.c Assumptions**

**2.c.1 Plan Cost Monotonicity (PCM)**

This assumption implies that if location b dominates location a in selectivity component of each predicate, Processing more tuples will have more cost

**2.c.2 Perfect Cost Model of Optimizer**

**2.c.3 Axis Parallel Concavity (APC)**

**2.c.4 Only**

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 applied over it.

Rational behind this assumption is that, if we supply same selectivity value to both old and new database, they will give outputs of different cardinalities. Also, we need to determine change of scale for and , in that case, maybe re-compilation of bouquet is needed.

**Sec. 3 Challenges in database size change**

Plan bouquet is suitable for canned queries as compilation overhead of is amortized over repeated invocations. Under situation of change of database size, placements of ideal contours will be totally different from existing contours built for old database size, in worst case it may result that old bouquet plans are totally different from new bouquet contours

**Sec. 4 Present Directions**

**4.a Using Old Plans**

During compilation phase of old Plan bouquet, multiple calls to query optimizer were made, by changing values of selectivities across . Doing this we already have POSP plan on old database, if new database has not changed much from old version, then same POSP set can be used as some approximation or POSP which can be obtained on new database, this approach is left for later exploration.

**4.b Using Old Contours**

Compilation phase of Bouquet based approach needs , this much cost cannot be afforded if database size changes frequently, also what we care is change of scale of each predicate as compared to what it was when bouquet was initially compiled.

Placement of contours in bouquet works like we first create a contour of and keep placing other contours from cost ratio less than cost of previous created contour.

Suppose we will be able to:

1. Use previous complete contours
2. Extend previous incomplete contour into extended
3. Create new contours in extended region only of

We will later prove how above extension to Plan bouquet can be made without any loss in MSO guarantee.

Consider a 1-Dimensional version of scale-up for now for a predicate , that goes through scale change, which can be evaluated as

By having scale factor, and assumption, we can view predicate on new database, as scaled up version of predicate on old database by extending maximum selectivity up to .

From now onwards, our entire formulation takes old database as reference, because bouquet is compiled on that.

Formula for number of contours

Using Same, number of contours in old and new database will be

If is extended to , in any dimensional ESS, last contour is always a point, a single optimal plan at that location is just an optimizer call away, so that last contour need not be saved.

IF [ is an INTEGER ] :

ELSE :

Note:

**4.c Number of Points Extra Points Needed**

Ratio for Geometric progression of selectivity values of old data base and new database to be created are

But, when we wish to extend old GP it becomes

Re-arranging to find , we get

To solve this equation further, we need to find what is, this minimum selectivity possible should also have some lower bound.

To derive lower bound on ℇ, we can consider worst case of Axis Parallel Concavity assumption

Using extreme lower bound value of , we get

Extra points needed on axis will be , which will be

This bound on Number of points needed is independent of RES. While if we have done through Arithmetic progression-based range of selectivities on each axis, it will be

Pathological case of all 0s selectivity will be skipped from this.

**5 Conclusion and Future Work**

Each predicate axis in will demand for only logarithm of scale change of that predicate from old to new database, which is also independent of taken to discretize .

Given proof of concept is not able to handle change of scale in , for now to handle this we can go with a conservative assumption of , which comes with curse of dimensionality.

Also, there can be some trivial predicates within a query, like simple scan of a relation without any filter will result different cardinalities on different sized database, and hence will impact optimal plans generated for both scale. In order to take care of this, Trivial predicates should be created and combined with .

Plans in certain cases will move in different ratio from their present iso-cost contour.

Considering , we can apply dimensionality reduction techniques, only if we have given distributional changes that will happen in database for

GENERAL DOUBT THAT WILL MATTER IN CASE OF ARBITRARY SCALE UP

1. Whether Sel values on axis are generated as A.P. or G.P
2. If generated as G.P. what is min\_sel, as this will impact certain future decisions
   1. G.P. will led to linear increase in number of points in exponential increase in scale

GENERAL NOTES