Design and Analysis of Parametric Query Optimization Algorithms

Sumit Ganguly
Department of Computer Science and Engineering
sganguly@iitk.ernet.in

Indian Institute of Technology, Kanpur, India 208016

Abstract

Query optimizers normally compile queries into one optimal plan by assuming complete knowledge of all cost parameters such as selectivity and resource availability. The execution of such plans could be sub-optimal when cost parameters are either unknown at compile time or change significantly between compile time and runtime [Loh89, GrW89]. Parametric query optimization [INS+92, CG94, GK94] optimizes a query into a number of candidate plans, each optimal for some region of the parameter space. In this paper, we present parametric query optimization algorithms. Our approach is based on the property that for linear cost functions, each parametric optimal plan is optimal in a convex polyhedral region of the parameter space. This property is used to optimize linear and non-linear cost functions. We also analyze the expected sizes of the parametric optimal set of plans and the number of plans produced by the Cole and Graefe algorithm [CG94].

1 Introduction

Database queries are optimized based on cost models that calculate costs for query plans. The cost of a query plan depends on parameters such as base and intermediate relation cardinalities, predicate selectivities, available memory, disk bandwidth, processor speeds and existence of access paths. The values of some of these parameters may change over time because of change in the database state, access paths and the execution environment. Moreover, estimating parameters for queries containing unbound variables is often not possible. The compilation of a query into a single "static" plan [GrW89] could result in significantly sub-optimal executions. This issue has been pointed out by Lohman [Loh89], Graefe and Ward [GrW89]. Approaches towards this problem have been presented by Ioannidis et.al. [INS+92], Cole and Graefe [CG94], Antoshenkov [Ant93] and by Krishnamurthy and this author [GK94].

1.1 The Parametric Query Optimization Problem

Let $s_1, s_2, \ldots, s_n$ denote n parameters, where each s_i quantifies some cost parameter, such as selectivity, table sizes, available memory etc. The cost of a plan p chosen from the space of feasible plans for the query is a function of the n parameters and is denoted by $C(p, s_1, s_2, \ldots, s_n)$. For every legal value of the parameters, there is some plan that is optimal for that value. Given a query and n parameters, the *parametric optimal set of plans* is the set of plans, each member of which is optimal for some point in the n-dimensional parameter space. Mathematically, the parametric optimal set may be defined as

$$\{p \mid p \text{ is optimal for a point in parameter space } \}$$

For every legal value of the parameters $s_1, \ldots, s_n$, there is a plan in the parametric optimal set that is optimal for that value and vice-versa. The *region of optimality* for a plan p is denoted by $R(p)$ and is the set defined as

$$\{(s_1, \ldots, s_n) | \ p \text{ is optimal at } (s_1, \ldots, s_n)\}$$

Proceedings of the 24th VLDB Conference
New York, USA, 1998

The problem of parametric query optimization is to find the parametric optimal set of plans and the region of optimality for each parametric optimal plan.

1.2 Review of Existing Solutions

Three techniques [CG94, INS+92, GK94] with varying degrees of generality have been proposed to solve the parametric query optimization problem. The most general technique is due to Cole and Graefe [CG94] to which we refer as the *CG* technique. In this approach, the cost of a plan p is modeled as an interval $[\ l(p), u(p)\ ]$, where $l(p)$ is the lowest cost of p over the parameter space and $u(p)$ is the highest cost of p over the parameter space. A partial order $<_{CG}$ is defined over the set of plans as follows. For plans p and q, $p <_{CG} q$, if the cost interval of p lies to the left of the cost interval of q, that is $u(p) < l(q)$. The *CG* technique computes the set of least plans with respect to $<_{CG}$, that is $\{p \mid \text{for all plans } q, \neg(q < p)\}$. There are two problems with the *CG* algorithm. First, it computes a superset of the parametric optimal set. As shown in Section 9.3, the expected number of plans generated by this technique could be much larger than the expected size of the parametric optimal set (for e.g., *CG* may yield $\sqrt{N}$ or more plans when the expected size of the parametric optimal set is $\approx ln\ N$, where N is the cardinality of the plan space). The second problem with the *CG* technique is that it does not include any mechanism to find the regions of optimality.

Krishnamurthy and this author present an algorithm to compute the parametric optimal set for the specific case when the parameter is the ratio s of the load factors of two sites in a distributed database system. The cost of a plan is modeled as $W_1 + s \cdot W_2$, where W_1 and W_2 quantify the work done by the plan at each of the two sites. This algorithm generates exactly the parametric optimal set together with the regions of optimality. An alternative formulation of this algorithm is presented in Section 4.

Ioannidis, Ng, Shim and Sellis [INS+92] present a randomization approach to this problem. This approach does not give guarantees about producing all parametric optimal plans nor does it give any bounds on deviations from optimality.

1.3 Linear and Non-linear Cost functions

The cost of a plan as a function of the n parameters $C(p, s_1, s_2, \ldots, s_n)$ depends on the cost model followed by the optimizer. Consider a query in which the selectivity of a single predicate is unknown. Using a simple cost model based on a uniform distribution assumption of data values ([SAC+79]), $C(p, s)$ may be expressed as $C(p, s) = x_0(p) + x_1(p) \cdot s$, where

- $x_0(p)$ is the cost of that portion of p that is independent of s.
- $x_1(p) \cdot s$ is the cost of that portion of p that is linearly dependent on s.

If a query is the union of two subqueries, such that in each subquery, there is a predicate with unknown selectivity, then the cost function of a simple optimizer may be of the form

$$C(p, s) = x_0(p) + x_1(p) \cdot s + x_2(p) \cdot t$$

Consider a query in which two predicates with unknown selectivities (s_1 and s_2 respectively) are connected by an "and" operator, Then, the cost function may be expressed as (under simplifying conditions)

$$C(p, s_1, s_2) = x_0(p) + x_1(p) \cdot s + x_2(p) \cdot t + x_3(p) \cdot s \cdot t$$

where

- $x_0(p)$ is the cost of that portion of p that is does not depend on either selectivities,
- $x_1(p) \cdot s$ is the cost of that portion of p that depends only on s,
- $x_2(p) \cdot t$ is the cost of that portion of p that depends only on t, and,
- $x_3(p) \cdot s \cdot t$ is the cost of that portion of the plan that depends on both s and t.

This shows that non-linear functions arise naturally even in simple cost models. The cost function when memory is a parameter could be expected to be a piecewise linear functions.

1.4 Summary of Paper

This paper presents a solution technique for the parametric query optimization problem by considering the structure of the regions of optimality. First, it is shown that, for the n-parameter linear cost functions, the parameter space can be decomposed into polyhedrons, each polyhedron being the region of optimality for some plan in the parametric optimal set and vice-versa. We then consider the problem of computing the parametric optimal plans for single parameter (unary) and two parameter (binary) linear cost functions.

The algorithm for binary linear cost functions traverses the edges of each optimal region (convex polygons) in a counterclockwise order. This traversal is facilitated by two routines *neighbor* and *corder*. Given a vertex of a polygon representing a region of optimality, and the direction of an edge, *neighbor* yields the next vertex and the set of plan(s) optimal there. Given a set of equally optimal plans at a vertex, *corder* orders

these plans according to the sequence in which their regions would be encountered if we rotate a vector of very small magnitude in a clockwise direction starting along a specific direction. Solutions for non-linear cost functions is illustrated by embedding non-linear curves within the regions of optimality of a generalized linear cost function. We also present a probabilistic analysis of the expected sizes of the parametric optimal set. This section shows that for one and two parameters (a) the expected sizes of the parametric optimal set of plans is not very large and (b) the general approach of Cole and Graefe may give a large number of non-optimal plans.

1.5 Organization

The paper is organized as follows. Section 2 presents notations for cost functions. Section 3 discusses the structure of the regions of optimality and the subroutine *ccorder*. Section 4 discusses an algorithm for unary linear cost functions. Section 5 presents the *neighbor* subroutine and Section 6 discusses an algorithm for binary linear cost functions. Section 7 discusses an approach for non-linear cost functions by embedding curves or surfaces in a generalized linear space. Section 8 sketches an algorithm for specific unary non-linear cost functions. Section 9 is concerned with probabilistic analysis of the size of the parametric optimal set. Finally, we conclude in Section 10.

2 Structure of Parameter Space for Linear cost functions

In this section, we study some structural properties of the regions of optimality of parametric optimal plans. We also discuss the concepts of iso-cost planes for pairs of plans and iso-optimal set of plans. Finally we consider the problem of angular ordering of plans at a vertex of an optimal region and present an algorithm for solving it.

The region of optimality of a parametric optimal plan p for an n-ary cost function parameterized by $\mathbf{s} = s_1, s_2, \ldots, s_n$, is denoted by $R(p)$ and is defined as $\{\mathbf{s}|p$ is optimal at $\mathbf{s}\}$

2.1 Polyhedral decomposition of Parameter value space

Consider linear n-ary cost functions. The following lemma expresses a simple property of linear cost functions.

Lemma 1 *Let plan p be optimal with respect to an n-ary linear cost function at points $\mathbf{u}$ and $\mathbf{v}$ of the n-dimensional parameter space. Then p is optimal for each point $\mathbf{w}$ that lies on the line segment joining $\mathbf{u}$ with $\mathbf{v}$.*

The above lemma immediately implies the following theorem.

Theorem 2 *Let p be a parametric optimal plan with respect to an n-ary linear cost function. Then the region of optimality for a plan p is a convex polyhedron.*

Theorem 2 argues that the parameter value space is partitioned into convex polyhedrons, each polyhedron being the region of optimality of a parametric optimal plan. The parametric query optimization problem for linear cost functions may be equivalently viewed as *finding the polyhedral decomposition of the parameter space induced by the parametric optimal plans.*

2.2 Iso-cost planes, iso-optimal plans and restrictions of cost functions

Consider n-ary linear cost functions. For any plans p and q, the set of points $\mathbf{u}$ in the n-dimensional parameter space such that p and q have the same cost at $\mathbf{u}$ is a plane and is called as the *p, q iso-cost plane*. To see this, consider the equation $C(p, \mathbf{u}) = C(q, \mathbf{u})$. Since C is a linear function in u, the equation represents a plane in n-dimensions.

Let us consider unary linear cost function with $C(p, u) = x_1 + y_1 \cdot s$ and $C(q, u) = x_2 + y_2 \cdot u$. The equation $C(p, u) = C(q, u)$ can be solved to give a point $u = (x_1 - x_2)/(y_2 - y_1)$. The p, q iso-cost plane for unary linear cost functions is a point and is called the p, q iso-cost point, or even more simply as the p, q iso-point. Similarly, the p, q iso-cost plane for binary linear cost functions is a line and is referred to as the p, q iso-cost line or as the p, q iso-line.

Multiple plans could be equally optimal at a point. For example, at each vertex of the polyhedral decomposition which lies in the interior of the admissible parameter space, multiple plans have the same cost and are optimal. Such plans are called iso-optimal plans.

Consider an n-ary linear cost function of the form

$$C(p, s_1, s_2, \ldots, s_n) = x_0(p) + x_1(p) \cdot s_1 + \ldots + x_n(p) \cdot s_n$$

Given a set of plans pl, the function $lowest(i, pl)$ returns the plan(s) with the least value of the i^{th} coordinate x_i. Let $\mathbf{u}$ be an n-dimensional vector and $\mathbf{d}$ be a unit length n-dimensional vector. The set of points that can be expressed in the form $\mathbf{u} + \alpha\, \mathbf{d}$ for $\alpha \geq 0$, represents a ray emanating from $\mathbf{u}$ in the direction $\mathbf{d}$. The cost of a plan p on this ray is a function of a single parameter α and may be expressed as

$$C_{u,d}(p, \alpha) = C(p, \mathbf{u}) + \alpha \cdot \mathbf{y}(p) \cdot \mathbf{d}^T$$

where $\mathbf{y}(p) = (x_1(p), \ldots, x_n(p))$ and $\cdot$ represents the vector dot product. The function $C_{u,d}$ is said to be the restriction of the original cost function C to the

given ray defined by **u** and **d**. Given a set of plans pl, the function $lowest(i, \mathbf{u}, \mathbf{d}, pl)$ returns the plan(s) with the least value in the i^{th} coordinate of $C_{u,d}(p, \alpha)$.

2.3 Angular Ordering of Iso-optimal plans at a vertex

Consider binary linear cost functions and the associated convex polygonal decomposition of the 2-dimensional parameter value space. Let v be a vertex of a polygon corresponding to $R(p)$ for some p in the parametric optimal set. Thus v is the intersection point of the optimal region of possibly more than one plan. Let pl be the set of plans iso-optimal at v. Choose a vector **l** of a very small magnitude δ, with one end fixed at v and initially aligned along a given direction **d**. If we rotate **l** clockwise a full circle, then the other end of **l** successively passes through the optimal regions of plans in pl. This circular order of plans so obtained is called the *clockwise ordering of the plans that are iso-optimal at v* with reference to **d**. Analogously, one can define the clockwise ordering. The idea is illustrated in Figure 1.

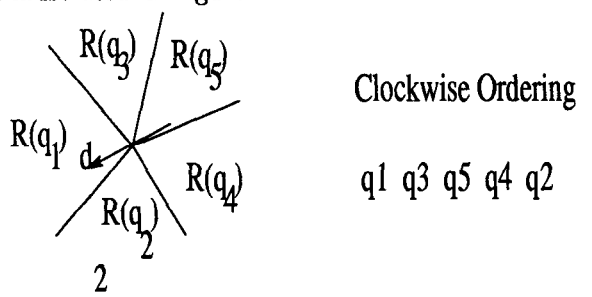

Figure 1: Angular ordering of iso-optimal plans in 2-dimensional parameter space

2.4 Computing the clockwise ordering of iso-optimal plans

Consider binary linear cost functions with parameters s and t. Given a direction **d**, a vertex v and the set of iso-optimal plans pv at v, we design an algorithm to compute the ordering of plans in pv with reference to **d**. In addition, we assume that we are also given $p_1 \in pv$ which is the first plan in the clockwise ordering. Consider the p_1, q iso-line for any $q \in pl$, $q \neq p_1$. This line makes an angle θ with **d** measured clockwise from **d**. The plan p_2 that makes the least angle is the second plan in the clockwise ordering. The procedure can be repeated with p_2 assuming the role of p_1 and considering plans in the set $pl - \{p_1, p_2\}$ and so on.

The angle that an iso-line makes with a direction vector **d** is computed using simple vector algebra. First, the equation of the iso-line is found. Let p and q be plans with cost coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2) respectively. The equation of the p, q iso-line is $x_1 - x_2 + s \cdot (y_1 - y_2) + t \cdot (z_1 - z_2) = 0$. We then find the angle that the line makes with the given direction vector **d** measured clockwise from **d**, using the following procedure.

For any line with equation $x + y \cdot s + z \cdot t = 0$, the direction of the line is given by the vector $\mathbf{e} = (z, -y)$ or its negative $(-z, y)$. If **e** is counterclockwise from **d**, then the angle between the line $x + y \cdot s + z \cdot t = 0$ and the vector d is quantified by $\mathbf{e} \cdot \mathbf{d}/|\mathbf{d}|$. Otherwise, the angle is quantified as $-\mathbf{e} \cdot \mathbf{d}/|\mathbf{d}|$. Here, $\times$ and $\cdot$ should be interpreted as vector cross and dot product operators respectively.

The procedure for angular (clockwise) ordering of iso-optimal plans outlined above also produces the directions of the edges of the polygonal regions of optimality emanating at v. This procedure is called $csort()$, although we do not show the pseudo-code as that would essentially contain elementary vector algebra. Actually, it is not necessary to assume that the first plan p_1 in the clockwise ordering is known. However, we do not discuss this issue any further.

3 Parametric query optimization for unary linear cost functions

In this section, we present an optimization procedure for unary cost functions, that is, when the cost of a plan p is of the form $C(p, s) = x(p) + s \cdot y(p)$. We assume that $a \leq s \leq b$, where a and b are given constants. of a single predicate is a parameter etc. We assume that a traditional query optimizer function $optimize(s)$ returns the set of iso-optimal plans at s.

The procedure used is the following. Let pa and pb denote the set of iso-optimal plans at a and b respectively. If $pa \cap pb$ is non-empty, say it contains p, then p is optimal at points a and b and therefore for each point in the interval $[a, b]$, allowing us to terminate. Otherwise, let p be the plan $lowest(2, pa)$ and q be the plan $lowest(1, pb)$. Let s be the p, q iso-point and pc be the set of iso-optimal plans at s. Now there are two cases, (i) either p and q are both in pc or (ii) neither p nor q is in pc. If case (i) holds, we terminate the search, since p is optimal at a and s and therefore in the interval $[a, s]$ and analogously, q is optimal in the interval $[s, b]$. If case (ii) holds, we partition $[a, b]$ into intervals $[a, s]$ and $[s, b]$ and we recursively use this procedure on these intervals. The procedure is presented in Figure 2, where it is called as *find-all-segments*. The algorithm is an alternative and easier formulation of the algorithm presented in [GK94].

The complexity of the algorithm is dominated by the calls to the *optimize* routine. If the number of parametric optimal plans is h, then the number of calls

procedure find-all-segments(a, b) {
Input: Parameter s lies in $[a, b]$.
Output: Parametric optimal plans in $[a, b]$ and their regions of optimality expressed as segments $[u, v]$.

1. $pa := optimize(a)$;
   /* find optimal plan at left end */
2. $pb := optimize(b)$;
   /* find optimal plan at right end */
3. **if** $(pa \cap pb \neq \phi)$
4. **return** $[a, b], pa \cap pb$;
   /* every plan in $pa \cap pb$ is optimal for the entire range $[a, b]$
5. $p := lowest(2, pa)$;
   /* if $cost = x + s.y$, then p is the plan in pa with lowest cost on y */
6. $q := lowest(1, pb)$;
   /* q is the plan in pa with lowest cost on x */
7. **return** *find-segments* (a, p, b, q);
8. } /* end of *procedure find-all-segments* */

procedure find-segments (u, p, v, q) {
/* $u \leq v$, $p = lowest(2, optimize(u))$ and $q = lowest(1, optimize(v))$ */

1. $s := iso - point(p, q)$;
   /* find the p, q isopoint */
2. $pc := optimize(s)$;
3. **if** $p \in pc$
4. **return** $([a, s], p) \circ ([s, b], q)$;
   /* $\circ$ is the list concatenation operator*/
5. $rx := lowest(1, pc)$;
6. $ry := lowest(2, pc)$;
7. **return**
   find-segments $(u, p, s, rx) \circ$
   find-segments (s, ry, v, q);
8. } /* end of *procedure find-segments* */

Figure 2: Parametric query optimization algorithm for unary linear cost functions

to optimize is $max(2h - 1, 2)$ [GK94].

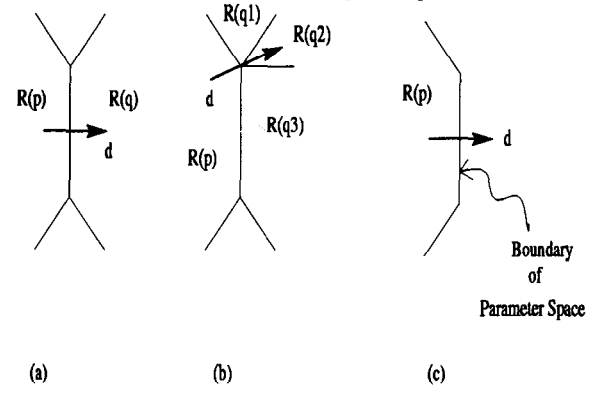

Figure 3: Examples of neighbors

4 Computing neighboring plans along a direction for binary linear cost functions

Suppose that cost function for a plan p has the form $C(p) = x(p) + s \cdot y(p) + t \cdot z(p)$. We first define the set of neighboring plans along a direction. We then present an algorithm to compute this set.

4.1 Neighbors along a direction

Consider the convex polygonal partitioning of the parameter space induced by the parametric optimal set. Let p be a parametric optimal plan and let $R(p)$ denote the polygon which is the region of optimality for p. Let $\mathbf{u}$ be a point in $R(p)$ and let $\mathbf{d}$ be a direction vector designating a ray emanating from u, as shown in Figure 3.

The *neighbors of p along* $\mathbf{d}$ starting at $\mathbf{u}$ is the set of plans q such that the regions of optimality $R(p)$, $R(q)$ and the ray $\mathbf{u} + \alpha \cdot \mathbf{d}$, for scalar $\alpha > 0$ intersect at a single point. By definition, neighbor of p also includes p. Figure 3 illustrates the three types of neighbors possible. In (a), the neighbor of p along $\mathbf{d}$ from $\mathbf{u}$ is the set $\{p, q\}$. In (b), the neighbor is the set of plans $\{p, q1, q2, q3\}$. In (c), neighbor is the singleton set $\{p\}$. The function $neighbor(p, \mathbf{u}, \mathbf{d}, \beta)$ is used to denote the set of plans q such that $R(p), R(q)$ and the directed line segment $\mathbf{u} + \alpha \cdot \mathbf{d}$, for $0 < \alpha \leq \beta$ intersects at a single point. Again, by definition, $neighbor(p, \mathbf{u}, \mathbf{d}, \beta)$ includes p.

An alternative equivalent formulation of neighbor is the following. The line segment $\mathbf{u} + \alpha \cdot \mathbf{d}$, for $0 \leq \alpha \leq \beta$ is a segment of the affine set. The parametric optimal set for the unary linear cost function $C_{\mathbf{u},\mathbf{d}}$, defined in Section 2, is a sequence of plans $p = p_1, p_2, \ldots, p_h$ and a sequence of values $s_0 = 0 < s_1 < s_2 < \cdots < s_h = \beta$ such that p_i is optimal for the interval $[s_{i-1}, s_i]$. In this interpretation, $neighbor(p, u, d, \beta)$ is the set of plans

that are optimal at $\alpha = s_1$ in the affine set or equivalently, at $\mathbf{u} + s_1 \cdot \mathbf{d}$ in the 2-dimensional parameter space. This formulation is used to design an algorithm for computing the neighbor function in Section 5.2.

4.2 Algorithm to compute neighbors

In this section, we present an algorithm to compute *neighbor*$(p, \mathbf{u}, \mathbf{d}, \beta)$. We modify the procedure for finding the parametric optimal set for linear unary cost functions with two basic differences. First, the cost function for a plan p is given by $C_{\mathrm{u,d}}(p, \alpha)$. Thus, the notion of iso-point is defined with respect to the affine cost function. Secondly, in the *find-segments* procedure (Figure 2) the input interval $[u, v]$ is partitioned into two segments $[u, s]$ and $[s, v]$, and the procedure is recursively invoked on both segments. For the neighbor function, it is only necessary to recursively descend along the left partition $[u, s]$.

The algorithm for *neighbor*$(p, \mathbf{u}, \mathbf{d}, \beta)$ is given in Figure 4. The procedure uses a traditional optimizer function *optimize*$(\mathbf{v})$, where $\mathbf{v}$ is a two-dimensional point, to return the plan(s) that are equally optimal at $\mathbf{v}$. The function *isopoint*$(\mathbf{u}, \mathbf{d}, p, q)$ returns the point $\mathbf{u} + \mathbf{d} \cdot \alpha$ where $C_{\mathrm{u,d}}(p, \alpha) = C_{\mathrm{u,d}}(q, \alpha)$.

procedure neighbor(p, $\mathbf{u}$, $\mathbf{d}$, β)
Input: $\mathbf{u}$ and $\mathbf{d}$ are 2 dimensional vectors. p = *lowest*$(2, \mathbf{u}, \mathbf{d}, $*optimize*$(\mathbf{u}))$.
Output: Find neighbor along $\mathbf{u} + \mathbf{d} \cdot \alpha$, $0 < \alpha \leq \beta$. It returns the neighboring set of plans and the neighboring point.

1. $pb :=$ *optimize*$(\mathbf{u} + \mathbf{d}\beta)$;
2. **if** $p \in pa$ return $(pa, \mathbf{u} + \mathbf{d} \cdot \beta)$;
3. $q :=$ *lowest*$(1, \mathbf{u}, \mathbf{d}, pb)$;
4. **return** *neighbor-segment*$(p, \mathbf{u}, \mathbf{d}, q)$;
5. }

procedure neighbor-segment$(p, \mathbf{u}, \mathbf{d}, q)$ {
1. $\mathbf{v} :=$ *isopoint*$(\mathbf{u}, \mathbf{d}, p, q)$;
2. $pc :=$ *optimize*$(\mathbf{v})$;
3. **if** $p \in pc$ **return** (pc, v);
4. $rx :=$ *lowest*$(1, \mathbf{u}, \mathbf{d}, pc)$;
5. **return** *neighbor-segment* $(p, \mathbf{u}, \mathbf{d}, rx)$;
6. }

Figure 4: An algorithm to compute the neighbor function

The complexity of *neighbor*$(p, \mathbf{u}, \mathbf{d}, \beta)$ is derived from calls to the *optimize* function. Suppose that there are h optimal regions in the parametric optimal set for $C_{\mathrm{u,d}}$ for $0 \leq \alpha \leq \beta$, corresponding to the segment boundary points $0 = s_0 < s_1 < s_2 < \cdots < s_h = \beta$. At each call to *neighbor-segment*, a segment $[s_i\Gamma_1, s_i]$ is chosen. If we assume that the probability of choosing any of the h segments is identical, then the expected number of calls to optimize is $H_h\Gamma_2 + 2$, where H_k is the k^{th} harmonic number. The expected value of h is $O(ln\ N)$ (see Section 9), where N is the number of plans in the plan space. In this case, it can be proved that the expected number of calls that the routine *neighbor* makes to the routine *optimize* is $O(ln(ln\ N))$.

5 Computing parametric optimal plans for binary linear cost functions

In this section, we describe an algorithm that solves the parametric query optimization problem when any plan p has the cost function of the form $x(p) + y(p) \cdot s + z(p) \cdot t$. The set of values that s and t can take is assumed to be defined by a convex polygon whose boundary B is specified by a list of edges in counterclockwise order. For example, the boundary B could define a rectangle in many practical cases. The two main subroutines used by the algorithm are *csort* and *neighbor* discussed in Sections 3 and 5 respectively.

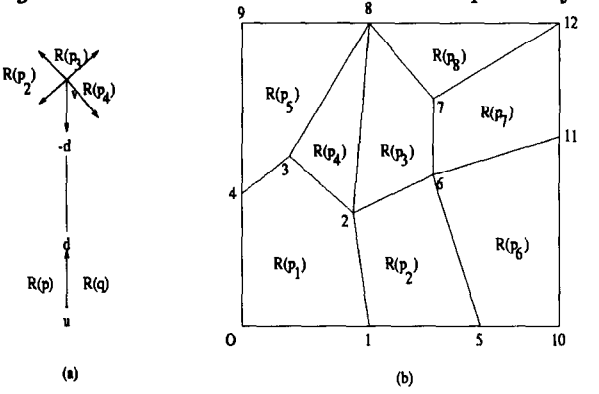

Figure 5: Illustrating the basic ideas in the *find-all-regions* procedure

5.1 Basic idea of the algorithm

The algorithm starts by traversing along the first edge of the boundary B. In general, it traverses the edges of the regions of optimality in a counterclockwise direction. Each region of optimality is completely traversed before another region is considered. During this traversal, new parametric optimal plans may be discovered and their regions are subsequently traversed. The regions of plans are considered in *breadth-first* order and the procedure terminates when the queue of plans corresponding to the breadth-first traversal is empty.

Suppose we are given a vertex u of the polygonal partitioning and a unit vector $\mathbf{d}$ parallel to an edge (u, v) of a polygon. Further, we are given plans p and

procedure find-all-regions(B)
Input: B defines the boundary of the 2 dimensional legal parameter space.
Output: Parametric optimal plans and their regions of optimality.
Initial State: Queue is empty. Edgelists of all plans are empty.

```
1.    v := first vertex of B;
2.    d := direction of first edge of B;
3.    p := least(2, v, d, optimize(v)) :
4.    insert(p, Visited);
5.    enqueue(p);
6.    insert_path(p.edgelist,
            new edge(v, UNKNOWN,
            d, NULLPLAN)) ;
7.    q := dequeue();
8.    while (q ≠ NULL) {
9.          find-region(q, B);
10.         q := dequeue();
11.   }     /* end while */
12. } /* end of procedure find-all-regions */
```

Figure 6: Algorithm for the parametric query optimization problem for binary linear cost functions

q, such that their regions of optimality, $R(p)$ and $R(q)$ lie to the left and right respectively as we traverse from u along direction d (See Figure 5(a)). Using *neighbor*, we can find the coordinates of v and the plans that are iso-optimal at v, that is, plans whose regions of optimality share a vertex at v. Using *csort*, we can find the direction vectors of each of the edges at v. The leftmost turn at v with respect to the incoming direction $\mathbf{d}$ or equivalently, the first edge in clockwise order with respect to the direction $-\mathbf{d}$ gives us the an edge direction of $R(p)$ which follows the edge direction $\mathbf{d}$ in the counterclockwise traversal of the edges of $R(p)$.

Therefore, given a vertex u of an optimal region $R(p)$ and an edge direction, we can traverse the edges of $R(p)$ in a counterclockwise order (i.e., by keeping the interior of $R(p)$ to the left). Each vertex is visited exactly once. The algorithm when applied to the polygonal partitioning shown in Figure 5(b) discovers vertices in the breadth-first order as numbered in the figure. It is assumed that we start at vertex numbered 0 and initially move along the edge 0—10.

The following property of breadth-first search is used by the algorithm to avoid visiting a vertex twice. Suppose a set of edges of the boundary of $R(p)$ $e_1, e_2, \ldots, e_k$ is visited before we start traversing $R(p)$ (i.e., before p comes to the head of the queue). Then, this set of edges forms a path although the sequence of edges in the path may not be identical to the sequence in which the edges have been visited. The algorithm is presented in Figure 6. This procedure uses an auxiliary procedure *find-region* that traverses the region of optimality of a given plan. This procedure is presented in Figure 7.

5.2 Detailed Description of the Algorithm

A plan p is represented by its cost coordinates $(x(p), y(p), z(p))$. For each parametric plan p, a list of edges *el* is kept such that *el* traversed in sequence is a path and when traversed keeps the interior of $R(p)$ to its left. The (p, el) pairs are kept in a global data structure. Each member of *el* models an edge and is a record with schema (*FromVertex, ToVertex, Direction, Rplan*). When a vertex is discovered, the *ToVertex* field is unknown, although, the *Direction* is known (this information is obtained from *csort*). *Rplan* is the plan whose region of optimality lies to the right if we traverse the edge. The plan to the left of the edge is p itself and so is not stored. *Rplan* defaults to *NULLPLAN* if the edge is on the admissible space boundary B.

The second global data structure kept is that of a queue of plans to implement breadth-first traversal. A set *Visited* of plans is also kept to remember if a plan was discovered earlier.

The function *find-all-regions* returns the set of (p, el) pairs that identifies the parametric optimal plans and their optimal regions. The auxiliary function *find-region*(p, B) returns the sequence of edges that define the region of optimality $R(p)$.

The following simple functions are used by the procedure. Given a list l, $l.first()$ and $l.last()$ return the first and last member of l. The function *insert_path*(el, e) inserts an edge e either at the beginning of the path *el* or at the end of the path *el*, such that after insertion, *el* still remains a path. Due to a property of breadth-first traversal as mentioned in the previous section, the set of edges traversed for any plan is always a path.

Given points $\mathbf{f}, \mathbf{t}$ and a direction $\mathbf{d}$, the function *collinear*$(\mathbf{f}, \mathbf{t}, \mathbf{d})$ is true if the vector $(\mathbf{t} - \mathbf{f})$ is parallel to $\mathbf{d}$ (i.e., $(\mathbf{t} - \mathbf{f}) \times \mathbf{d} = 0$ and $(\mathbf{t} - \mathbf{f}) \cdot \mathbf{d} = 1$). Given a sequence of boundary edges B and a ray emanating from v in the direction $\mathbf{d}$, *find-intersection* (B, v, d, k) finds the index of the edge in B that the ray intersects using a sequential search algorithm. The parameter k specifies that the sequential search starts the check at edge k, proceeds along the list by wrapping around if necessary.

The following invariant is maintained when *find-region*(p, B) is called. Let (**cv**, **nextv**, **d**,*rp*) be the last edge in the edge list of p. Then **nextv** is UNKNOWN and **cv** and **d** are known. For every other

procedure find-region(p, B)
Input: p is a parametric optimal plan. B defines the boundary of the legal parameter space. A list of edges called *p.edgelist* is associated with p.
Output: Finds the boundary of the convex polygonal region for which p is optimal. This boundary is traced in counterclockwise order.

```
1.    el = p.edgelist;
2.    (iv, −, −, −) = el.first();
      /* − stands for don't care value;
      iv is initial vertex */
3.    (cv, −, d, rplan) = el.last();
/* cv is current (i.e., last known ) vertex */
4.    eno := 0;
5.    while (collinear(cv, iv, d) = FALSE) {
      /* iv is not along d from cv*/
6.        (eno, dist) :=
          find-intersection(B, cv, d, eno);
7.        (nv, pl) :=
          neighbor(p, cv, d, dist);
          /* new vertex nv is discovered */
8.        update el set
          el.last().ToVertex = nv;
9.        if (rplan ≠ NULLPLAN)
10.           insert_path(
              rplan.edgelist, new edge (
nv, cv, NOTNEEDED, p);
11.       m := pl.length();
          /* number of plans in pl */
12.       pa := pl.planarray();
          /* list of plans in pl */
13.       csort(m, pa, dl, −d, p);
14.       pa[m] := NULLPLAN;
15.       for i := 1 to m − 1 do {
16.           if (IsVisited(pa[i]) = FALSE) {
              /* pa[i] not yet visited */
17.               enqueue(pa[i]);
                  /* put in rear of queue */
18.               initialize p.edgelist to
(nv, UNKNOWN, dl[i], pa[i + 1]);
19.               insert(pa[i], Visited);
                  /* mark the plan as visited */
20.           }   /* end of if ... */
21.       } /* end of for ... */
22.       cv := nv;
23.       rplan := pa[1];
24.   }   /* end of while .... */
25. } /* end of procedure find-region */
```

Figure 7: Algorithm to find the region of a parametric optimal plan for binary linear cost functions

edge $(\mathbf{u}, \mathbf{v}, \mathbf{d}^0, p^0)$ in the edge list of p, all the above four attributes are known. Of course, if the direction $\mathbf{d}^0$ defines an edge along the boundary, then p^0 is set to NULLPLAN.

6 Structure of parametric optimal set for non-linear cost functions

Non-linear cost functions arise naturally in parametric query optimization. For example, let s and t be the unknown selectivities of two predicates in a conjunctive query. Then, a plan p has a cost function of the form

$$C(p, s, t) = x(p) + y(p) \cdot s + z(p) \cdot t + w(p) \cdot s \cdot t$$

where $x(p)$ is the cost of that portion of p that is does not depend on either selectivities, $y(p) \cdot s$ is the cost of that portion of p that depends only on s, $z(p) \cdot t$ is the cost of that portion of p that depends only on t and $w(p) \cdot s \cdot t$ is the cost of that portion of the plan that depends on both s and t.

In order to avoid working in three or higher dimensions, we study the structure of parametric optimal set of plans when the cost function of a plan p is of the form:

$$C(p, s) = x(p) + y(p) \cdot s + z(p) \cdot f(s)$$

where f is any smooth function of s. Generalizations to higher dimensions will be easy to see.

The definition of the region of optimality for a parametric optimal plan p is $R(p) = \{s |\ p \text{ is optimal at } s\}$. However, $R(p)$ *may not be a convex set.* We adopt the following approach for non-linear cost functions. Let t be an artificial parameter. Consider the cost function

$$C^{(2)}(p, s, t) = x(p) + y(p) \cdot s + z(p) \cdot t$$

Since $C^{(2)}$ is a binary linear function, the regions of optimality for parametric optimal plans for $C^{(2)}$ are convex polygons. The *polygons for* $C^{(2)}$ in the s, t space that *intersect with the curve* $t = f(s)$ *correspond to the parametric optimal regions for the function* C.

This is illustrated in Figure 8. In this figure, p_1 is optimal for the segment of the curve from 0 to a, p_2 is optimal for the segment from a to b, p_3 is optimal from b to c, p_4 from c to d and finally p_1 again (due to non-convexity) is optimal from d to e.

The structure of the parametric optimal set for piece-wise linear functions which arises when memory is a parameter is not explored in the paper and is left for future work.

7 Computing parametric optimal set for non-linear cost functions

In this section, we sketch an algorithm that computes the parametric optimal set and regions for unary cost

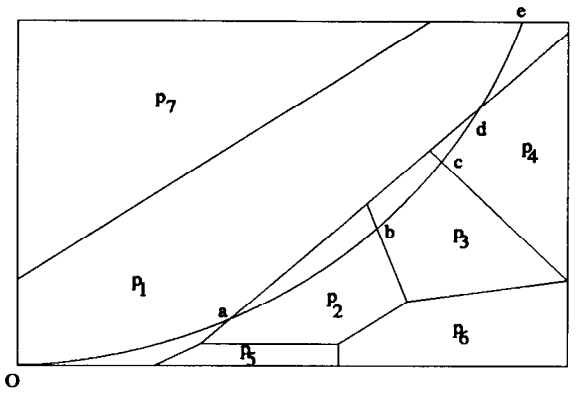

Figure 8: Embedding non-linear cost functions in a larger linear space

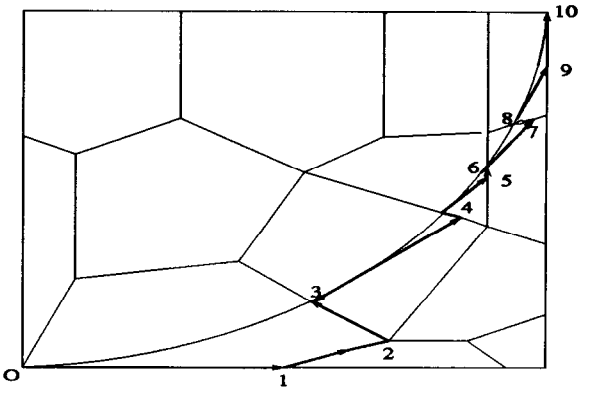

Figure 9: Illustration of algorithm used to compute the parametric optimal set for non-linear functions

functions of the form $C(p,s) = x(p) + s \cdot y(p) + f(s) \cdot z(p)$. We assume that f is a smooth convex curve (e.g., $f(s) = s^2$). We first conceptually embed the curve $t = f(s)$ in the polygonal partitioning induced by the parametric optimal set for the generalized linear function $C^{(2)}(p,s,t) = x(p) + s \cdot y(p) + t \cdot z(p)$ (for e.g., see Figure 9).

We will illustrate the idea informally by using Figure 9. Suppose we start at origin 0 in Figure 9. Let p be the optimal plan at 0 along the direction of the tangent at 0. If we now invoke the subroutine *neighbor* along the direction of the tangent to the curve at 0, then we arrive at point numbered 1, which is the first cross over point along the tangential direction. We take the leftmost turn at 1, using a variant of the subroutine *csort* and call *neighbor* again to obtain point 2. We repeat this process until for some i, the line segment joining points i and $i-1$ intersect the curve. Since the curve is assumed to be convex, an intersection point must exist. This gives us the intersection point 3 in Figure 9. We then restart the process by navigating along the tangential direction at 3. The path traced out by successive iterations of this procedure is shown in Figure 9.

The following details have been not been specified for sake of brevity. The line segment joining points i and $i-1$ may intersect the curve in more than one point. Secondly, a point on the curve may also be a vertex of the polygonal partitioning of $C^{(2)}$. All these issues can be satisfactorily handled.

8 Expected Sizes

In this section, we analyze the expected sizes of the parametric optimal set of plans under simple probability distributions. The following notation is used in this section. The size of the set of all the plans for a given query is denoted by N. For any natural number n, H_n denotes the n^{th} harmonic number defined as $\sum_{i=1}^{n} 1/i$. For large n, $H_n \approx ln\ n + \gamma + O(1/n)$, where γ is Euler's constant and is equal to 0.577.... The size of the parametric optimal set is denoted by h. The expected size of the parametric optimal set is denoted by $E(h)$.

We first consider the problem of determining $E(h)$ under simple probability distributions for the case of unary linear functions. Most results in this area are either known from mathematics or from computational geometry. We then consider the same problem for binary non-linear functions, of the kind that would be expected when the parameters are unknown predicate selectivities. The results in this area are not available in the literature (to the best of the knowledge of this author). We then consider the problem of determining the expected size of set of plans produced by the Cole and Graefe algorithm. The section concludes with a discussion.

8.1 Unary Linear cost functions

The cost of a plan p is given by a function $C(p,s) = x(p) + s \cdot y(p)$, and equivalently, the cost of a plan may be thought of as a point $(x(p), y(p))$ in two-dimensional space.

Theorem 3 *If the cost of N plans are chosen uniformly and randomly from a rectangle, then $E(h) = \Theta(ln\ N)$*

$$E(h) < \frac{2}{3} H_N + 2 + O\left(\frac{1}{N}\right)$$

The above result is attributed to Rényi and Sulanke [RS63].

Theorem 4 *If the cost of N plans are chosen independently from a 2-dimensional normal distribution, then $E(h) = \Theta(\sqrt{ln\ N})$. Furthermore*

$$E(h) < 2 + \sqrt{ln\ N} + O\left(\frac{ln\ N}{N}\right)$$

This result is originally due to Raynaud [Ray70].

Theorem 5 *If the cost of N plans are chosen independently from any set of continuous distributions, then $E(h) < H_N/2$.*

The above result is a specific case of a general theorem by Bentley, Kung, Schkolnick and Thompson [BKS+78].

8.2 Non-linear cost function in two variables

Assume that the cost function for a plan p is of the form

$$C(p,s,t) = x(p) + s \cdot y(p) + t \cdot z(p) + s \cdot t \cdot w(p)$$

The cost of each plan may be thought of as a point $(x(p), y(p), z(p), w(p))$ in a four-dimensional space.

Theorem 6 *If the cost of N plans are chosen independently from a four dimensional hypercube, then $E(h) = \Theta((ln\ N)^2)$.*

Theorem 7 *If the cost of N plans are chosen independently from a four dimensional normal distribution, then $E(h) = \Theta(ln\ N)$.*

8.3 Analysis of the Cole and Graefe algorithm

Let g represent the number of plans produced by the Cole and Graefe algorithm, that is the number of least plans with respect to the partial order $<_{CG}$. The expected value of g is denoted by $E(g)$. Consider unary linear cost functions (i.e., $C(p,s) = x(p) + s \cdot y(p)$). The cost of each plan is described as a point $(x(p), y(p))$ in two dimensional space.

Theorem 8 *If the cost of N plans are chosen uniformly and randomly within a rectangle, then $E(g) = (N\pi/2)^{1/2}$.*

Theorem 9 *If the cost of N plans are chosen independently from continuous distributions , then $E(g) \geq (N\pi/2)^{1/2}$.*

Theorem 10 *There exist normal distributions such that if the cost of N plans are chosen independently from these distributions, then $E(g) > N - O((N/ln\ N)^{1/2})$.*

8.4 Discussion

Theorems 3 through 7 depict the values of $E(h)$ under various assumptions and show that it is not extremely large. For comparison purposes, let N be 1 billion. This gives $H_N \approx 21$. For single parameter linear cost functions, $E(h)$ varies between 14 and 22 under the various probability distribution assumptions. Theorems 6 and 7 consider binary non-linear cost functions of the form likely to arise when the selectivities if two predicates are treated as parameters. Note that for uniform distributions, $E(h)$ is $\Theta((ln\ N)^2)$ although points are uniformly distributed in a four dimensional space. The constant for the leading order term is small (less than 1/10). For $N = 1$ billion, $E(h)$ can be calculated to be less than 40.

Theorems 8,9 and 10 discuss the expected number of plans $E(g)$ produced by the Cole and Graefe technique under assumptions similar to those made in Theorems 3 through 6. The values of $E(g)$ lie in the range from $\sqrt{N}$ to nearly N, whereas the value of $E(h)$ lies in the range from $\sqrt{ln\ N}$ to $ln\ N$. This shows that the Cole and Graefe technique could produce a large number of plans that are not members of the parametric optimal set of plans.

9 Conclusions

The paper discusses parametric query optimization algorithms for unary and binary linear cost functions and for simple unary non-linear cost functions. The approach is based on the property that the regions of optimality for linear cost functions are polyhedral. The query optimization algorithms proposed traverse along the boundaries of the polyhedron. Simple probabilistic models are used to convince the reader that the size of the parametric optimal set is not large for one or two parameters.

The paper raises several issues. Extensions of this idea to three and higher dimensions is an immediate one and may be found in [Gan98]. Another issue is to study the problem for piece-wise linear cost functions that arise naturally when available memory is considered as a parameter. A third question is to explore the approximate version of the parametric query optimization problem in which a set of approximate optimal plans are generated, such that for each value in the parameter space, there is at least one plan in the approximate set whose cost is within a given factor $(1+\epsilon)$ of the cost of the optimal plan at that value.

Acknowledgements

The author wishes to thank U. Maheshwara Rao and V.G.V. Prasad for their assistance in drawing the figures and the anonymous referees for their comments. The nomenclature of iso-cost lines and iso-optimal plans was given by Ravi Krishnamurthy, HP Labs.

References

[Ant93] G. Antoshenkov, "Dynamic Query Optimization in Rdb/VMS", *Proceedings*

of the IEEE International Conference on Data Engineering, Vienna, Austria, April 1993.

[BKS+78] J.L. Bentley, H.T. Kung, M. Schkolnick and C.D. Thompson, "On the average number of maxima in a set of vectors", *Journal of the ACM* **25**, 536-543 (1978).

[CG94] R. Cole and G. Graefe, "Optimization of Dynamic Query Evaluation Plans", *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, Minneapolis, Minnesota, USA.

[GrW89] G. Graefe and K. Ward, "Dynamic Query Evaluation Plans", *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, Portland, Oregon, USA.

[GK94] S. Ganguly and R. Krishnamurthy, "Parametric Query Optimization for Distributed Databases based on load conditions" *Proceedings of the 1994 International Conference on Management of Data*, Pune, India.

[Gan98] S. Ganguly, "Parametric Query Optimization: A tutorial" *Technical Report, IIT Kanpur* August 1998. Available from www.cse.iitk.ac.in (202.54.56.145).

[INS+92] Y.E. Ioannidis, R.T. Ng, K. Shim and T.K. Sellis, "Parametric Query Processing", *Proceedings of the 1992 International Conference on Very Large Databases*, Vancouver, British Columbia, Canada.

[Loh89] G.M. Lohman, "Is Query Optimization a 'Solved' Problem?", *Proceedings of Workshop on Database Query Optimization*, G. Graefe (editor), Oregon Graduate Center Computer Science Technical Report 89-005.

[Ray70] H. Raynaud, "Sur l'enveloppe convexe des nuages des points alèatoires dans R^n", *Journal of Applied Probability*, **7**, 35-48 (1970).

[RS63] Rényi and R. Sulanke, "Ueber die konvexe Hulle von n zufallig gewahlten Punkten I", *Z. Wahrschein*, **2**, 75-84 (1963).

[SAC+79] P.G. Selinger et.al. , "Access path selection in relational database systems", *Proceedings of 1979 ACM SIGMOD International Conference on Management of Data.*