Hardness of Approximate Query Optimization

[Extended Abstract]

S. Chatterji
University of California
Berkeley
souravc@eecs.berkeley.edu

S.S.K. Evani
Sun Microsystems
Bangalore, India
saikiran.surya@sun.com

S. Ganguly
Bell Labs
Lucent Technologies
Murray Hill, NJ
sumit@lucent.com

M.D. Yemmanuru
Sun Microsystems
Bangalore, India
mahesh.datt@sun.com

ABSTRACT

We study the complexity of the problem of approximate query optimization, namely, the hardness of finding approximately optimal plans for a given query. We show that for any $\delta > 0$, the problem of finding a join order sequence whose cost is within a factor $2^{\Theta(log^{1-\delta}(K))}$ of K, where K is the cost of the optimal join order sequence is NP-hard. Similar approximation gaps are obtained for different variants of the query optimization problem. Further, these results remain true if the number of edges in the query graph is constrained to be some given function $3n \leq e(n) \leq n^2/3$, where n is the number of vertices in the query graph. These results argue that, unless $P = NP$, the query optimization problem is inherently non-approximable to within any polylogarithmic factor of the optimal cost in polynomial time.

General Terms

Keywords

Databases, Query Optimization, Complexity, Approximation

1. INTRODUCTION

The problem of finding an optimal join order sequence for Select Project Join (SPJ) Queries is a classical problem in Query Optimization. A solution to this problem forms the core of any database query optimizer, and therefore, this problem has evoked considerable attention from researchers, from both a practical and theoretical standpoint.

It is well-known that this problem is NP-complete in general [5], and continues to be so in many variant formulations [2, ?]. Despite the set of results on NP-completeness, there may be a justifiable optimism about the possibility of polynomial time approximation algorithms; algorithms that, for a given query on a database, will return a plan whose cost is within a small factor of the cost of the optimal plan. Given the fundamental importance of the query optimization problem, the design of such approximation algorithms would be both practically relevant and theoretically interesting. This paper studies the complexity of efficiently computing approximate solutions to the query optimization problem. The results of this paper imply that designing an efficient approximation algorithm with a competitive ratio that is within a polylogarithmic factor of the cost of the optimal plan is NP-Hard.

We consider several variants of the query optimization problem. In one variant of the problem, we restrict all joins to computed using the nested-loops join method only, by following a cost model virtually identical to that of [5]. We then consider a variant where joins are restructed to be computed using the hash-joins method only. In either of these variants, our execution space allows the use of cartesian products in the join sequences. For both these cases, we show that, for any constant value of $\delta > 0$, the problem of computing a join order sequence whose cost is within a factor $2^{\Theta(log^{1-\delta}(K))}$ of K, where K is the cost of the optimal join order sequence, is NP-hard. We then restrict the problem by considering the complexity of approximate query optimization when the number of edges in the query graph matches a given function $2n \leq e(n) \leq n^2/3$, where n is the number of vertices of the query graph. Given this restriction on the space of queries, and for each of the two variants of the execution space, the complexity of approximation remains equally difficult. The results of this paper also hold for the cost models considered in [2].

The rest of the paper is organized as follows. In Section 2, we present an overview of related work in the area of complexity of query optimization. Section 3 presents the problem definitions for two variants of the query optimization problem - QO_N and QO_H, where QO_N and QO_H refer to the problem of finding approximate solutions to the query optimization problem when the join method is restricted to nested loop joins and hash joins, respectively. Section 4 presents the proof techniques used in proving the hardness of QO_N and QO_H. Section 5 presents the actual proofs for QO_N and QO_H. In Section 6, we summarize our work on the different variants of the query optimization problem including the problem SQO−CP that we study in detail in the appendix. Finally we conclude in Section 7.

2. RELATED WORK

In this section, we briefly describe and classify previous work in the area of complexity of query optimization.

One perspective on the problem of query optimization is

Principles of Database Systems 2002, Madison, Wisconsin USA

to view it as a family of optimization problems. A member of this family is chosen by specifying three parameters, namely, the query graph space, the execution space and the cost model.

A query graph is an undirected graph where the vertices are the query relations of the input instance and the edges are placed between pairs of vertices if the query relations corresponding to these vertices are related by a query predicate. Various classes of query graphs been considered in the literature, for example, general query graphs [5], tree queries [5], [7], [2] and star queries [2]. Examples of execution spaces considered are, the space of linear trees (i.e., join sequences), the space of bushy trees, space of allowing only nested loop join methods [5] etc.

The third parameter needed to specify a query optimization problem is a cost model that is used to estimate the cost of an execution tree. Cost models with varying levels of abstractions have been used; for examples, cost as the sum of the intermediate sizes [2] and a detailed cost model of Volcano [6].

Given an execution space and a cost model, the query optimization problem, which we call QO, is to compute the optimal (i.e., least) cost execution tree for each input instance. The approximate version of the QO problem can be stated as follows. Given an input instance, compute the execution tree, whose cost is within a factor p of the optimal cost. The term p can be a constant, or a function of the size of the input instance or a function of the optimal cost.

Ibaraki and Kameda [5], prove that the problem is NP-complete by (a) allowing the use of nested loop join methods and (b) disallowing the use of cartesian products between relations. Cluet and Moerkotte in [2], show that the problem is NP-Complete when (a) the query graph is a star graph and (b) cartesian products are allowed in join sequences.

3. PROBLEM DEFINITIONS

In this section, we give the definitions for the problems QO_N, QO_H. QO_N and QO_H stand for the approximate versions of the QO problem when the join method is restricted to nested loop joins and hash joins respectively.

3.1 Problem Specification for $\mathbf{QO_N}$

3.1.1 Cost model of $\mathbf{QO_N}$

In this section, we define the cost of sequences of joins for an instance Π of QO_N.

Let Z be any permutation of vertices in V and X be any prefix of Z. The number of tuples in the output of X, denoted by $N(X)$ is estimated as a product of relation sizes and the relevant selectivities. More formally

$$\begin{array}{lcl} N(X) & = & 1 \textit{ if } X = \phi \\ N(Xv_j) & = & N(X) \cdot n(v_j) \cdot \prod_{v_i \textit{ appears in } X}(s_{ij}). \end{array}$$

Here, $n(v_j)$ denotes the number of tuples in the relation R_j and s_{ij} denotes the selectivity of the join predicate between the relations R_i and R_j. Let $Z = Xv_jY$ be a sequence such that v_j occurs in position $i+1$ (i.e, $|X| = i$) for some $i \in \{1, 2, \ldots, n-1\}$. The cost of the nested loops join corresponding to v_j is denoted by $H_i(Z)$ and defined as

$$H_i(Z) \ = \ N(X) \cdot min\{w_{jk} \mid v_k \textit{ appears in } X\}$$

The cost of a sequence Z is defined as the sum of the cost of joins corresponding to each v_j occurring in Z, except the first one. That is,

$$C(Z) = \sum_{i=1}^{n-1} H_i(Z)$$

3.1.2 Instance Cost and Decision Problem

INSTANCE

1. The query graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_m\}$ and $E = \{e_1, e_2, \ldots, e_p\}$. For each v_i, there exists a relation R_i, $1 \leq i \leq m$. An edge e_k between two vertices v_i and v_j means that the relations R_i and R_j have a join predicate between them.
2. Selectivity of a predicate s_i, between 0 and 1 for $1 \leq$ i $\leq$ p.
3. T_i, the size of the relation R_i in terms of number of pages.
4. The page size PS.
5. Two values w_{jk} and w_{kj} corresponding to each edge e_i = { v_j, v_k }. The meaning of them is as follows. Let P_i be the predicate associated with the edge $e_i = \{v_j, v_k\}$. The value w_{jk} denotes the least cost of solving P_i for a given tuple of R_k(containing join attributes from R_k that are relevant to P_i) among all possible choices of access paths for R_j. The value w_{kj} is defined likewise with the roles of R_k and R_j reversed. If $\{v_j, v_k\} \notin E$, then w_{jk} is defined to be T_j.
6. A number K.

An instance of QO_N can be summarized as a tuple $(Q = (V, E), \overline{S}, \overline{T}, W)$, where $\overline{S}$ is the vector of edge selectivities $(s_1, s_2, \ldots, s_p)$ $(p = |E|)$, $\overline{T}$ is the vector of relation sizes $(T_1, T_2, \ldots, T_n)$ $(n = |V|)$ and W is the matrix $\{w_{ij}\}_{1 \leq i,j \leq n}$.

QUESTION
Does there exist a sequence X such that $C(X) \leq K$? □

3.2 Problem sepecification for QO_H

We now specify the problem of query optimization QO in which joins are computed using the hash joins method.

3.2.1 Cost Model for Hash Join Sequences

Consider the join $R \bowtie S$ to be performed using the *hash join algorithm*. Let b_R and b_S be the number of pages of relations R and S, respectively, m be the number of pages allocated to the join and m_{min} be the minimum number of pages that should be allocated for the join to take place. We assume here that S is the inner relation. The parameter m_{min} can be any function of b_S such that it is $> \sqrt{(}b_S)$. The cost of performing the *hash join* $R \bowtie S$, can be of the general form

$$\begin{cases} b_S & m \geq b_S \\ \beta \cdot b_R \cdot g(m, b_S) + h(m, b_S) & m_{min} \geq m < b_S \end{cases}$$

where, $h(m, b_S)$ and $g(m, b_S)$ can be any general functions decreasing in m and increasing in b_S with

1. $g(b_S, b_S) = 0$,

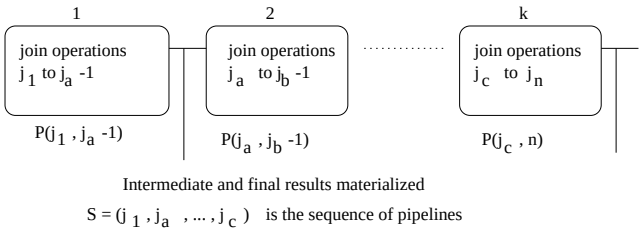

Figure 1: Pipelined execution of a query

2. $g(m, b_S) \geq b_S^{\frac{-\epsilon}{3}}$, for all values of $m < b_S$, where ϵ is a small constant $\ll 1$ mentioned later in Lemma 4,

3. $h(b_S, b_S) = b_S$,

4. $h(m_{min}, b_S) = \gamma \cdot b_S$, where γ is a constant,

5. β is a constant ≥ 2 and

The term $h(m, b_S)$ represents the cost of partitioning(hashing) the relation S and the term $\beta \cdot b_R \cdot g(m, b_S)$ represents the cost of partitioning the relation R which is streaming into memory and joining it with the relation S.

A query is executed by ordering the join operations into a sequence called the *join order sequence*. A *join order sequence* is decomposed into a series of *pipelines*. All the joins among the relations that fall into a pipeline are executed concurrently with the available main memory distributed among the joins. The intermediate result at the end of the pipeline is written to disk. The next pipeline reads the intermediate result from the disk and uses it as its outermost relation.

The cost of executing a given query is the sum of the costs of executing each of the pipelines. The cost of executing a pipeline is the sum of the following components:

1. the cost of reading from disk the intermediate relation materialized by the previous pipeline, if any,

2. the cost of executing each of the joins in the pipeline, and,

3. the cost of materializing the result of the joins executed in the pipeline.

Let n be the number of relations involved in a given query and M be the available main memory. And let $R_1, R_2, \ldots, R_n$ be the relations involved in the query. The notation j_i denotes the i^{th} join operation and m_i the memory allocated to the join. $P(j_i, j_k)$ means that the join operations $j_i, \ldots, j_k$ are performed in a single pipeline. A sequence of pipelines, S' is described by listing the number of the join each pipeline starts with, i.e., $S' = (j_{i_1}, j_{i_2}, \ldots, j_{i_k})$ means that there are k pipelines and the p^{th} pipeline starts with the join j_{i_p} and ends with the join $j_{i_{p+1}-1}$. Clearly, $j_{i_1} = 1$ and the last pipeline ends with the n^{th} join. The cost of executing the pipeline $P(j_i, j_k)$ is denoted by $C_{pipe}(j_i, j_k)$. The cost of executing the query is denoted by $C_{seq}(S', m', M)$, where $m' = (m_1, m_2, \ldots, m_n)$. It is simply the the sum of the costs of executing each of the pipelines in the sequence of pipelines S', as shown in Figure 1.

3.2.2 Instance Cost and Decision Problem

INSTANCE

1. The query graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_m\}$ and $E = \{e_1, e_2, \ldots, e_p\}$. For each v_i, there exists a relation R_i, $1 \leq i \leq m$. An edge e_k between two vertices v_i and v_j means that the relations R_i and R_j have a join predicate between them.

2. The page size PS.

3. m-dimensional vector $(n_1, n_2, \ldots n_m)$, where n_i is the number of tuples in R_i.

4. m-dimensional vector $(b_1, b_2, \ldots b_m)$ where b_i is the size of R_i in pages.

5. p-dimensional vector $(s_1, s_2, \ldots s_p)$ of non-negative real numbers where s_i is the selectivity of the predicate between the relations which the edge e_i represents.

6. The available main memory, M.

7. A number K.

QUESTION
Does there exist a sequence of pipelines, S' and a memory allocation vector, m' such that $C_{seq}(S', m', M) \leq K$?

4. PROOF TECHNIQUES AND PRELIMINARIES

In this section, we review the proof techniques that are used and establish notation that are used subsequently in the paper.

Our proof techniques mainly rely on the concept of *gap preserving reductions* [8]. The initial point is a result from [8], which exhibits a certain gap property in the MAX-3SAT problem, and shows that that the problem of closing this gap is NP-hard. We design a series of gap-preserving reductions through intermediate problems finally terminating in the QO_N problem. At each step of the reduction, we attempt to amplify the gap as much as possible. The end result is a proof that shows that designing an approximation algorithm that closes this final gap as determined for the QO_N problem is as difficult as closing the gap for the MAX-3SAT problem. Since closing the gap for the latter problem is known to be NP-hard, we conclude that approximating the QO_N problem to be better than the final gap determined is also NP-hard.

We review the concept of gap preserving reductions and establish our notation, and in particular, define the MAX-3SAT problem that was referred to earlier.

Definition 1: Let Π and Π' be two maximization problems, $OPT(I)$ and $OPT(I')$ be the cost of optimal solutions to instances I of Π and I' of Π' and ρ and ρ' be greater than 1. A gap preserving reduction with parameters $(c, \rho), (c', \rho')$ from Π to Π' is a polynomial time reduction of Π to Π' satisfying the following property.

$$\begin{aligned} OPT(I) \geq c &\Rightarrow OPT(I') \geq c' \\ OPT(I) \leq \tfrac{c}{\rho} &\Rightarrow OPT(I') \leq \tfrac{c'}{\rho'} \end{aligned}$$ □

Given a boolean formula ϕ in 3-CNF form and an assignment A of truth values to the variables, the expression $val(A/\phi)$

denotes the number of clauses in ϕ satisfied by A. We now define the problem MAX-3SAT(13), which is the specific version of the MAX-3SAT problem that we use as our starting point in the reductions.
INSTANCE of MAX-3SAT(13): A 3CNF formula ϕ, in which each variable occurs in at most 13 clauses and an integer k.
QUESTION: Is there an assignment A of truth values to the variables such that $val(A/\phi) \geq k$?

The proofs of non approximability to follow in later sections use the following maximization variants of the CLIQUE problem - MAX-CLIQUE and $\frac{2}{3}$CLIQUE, in which the degree of each vertex is constrained to be at least $|V| - 14$.
INSTANCE of MAX-CLIQUE: An undirected graph $G = (V, E)$ and an integer k and the degree of each vertex is $\geq |V| - 14$.
QUESTION: Is there a clique of size k ?
INSTANCE of $\frac{2}{3}$CLIQUE: An undirected graph $G = (V, E)$, where $|V| = n$ and $|E| > \frac{8 \cdot n^2 - 18 \cdot n}{18}$.
QUESTION: Is there a clique of size $\frac{2 \cdot n}{3}$?
Given a graph G, the size of the largest clique in G is denoted by $w(G)$. We now prove that there exist gap preserving reductions from MAX-3SAT(13) to MAX-CLIQUE and $\frac{2}{3}$CLIQUE. The proofs are similar to the reduction of MAX-3SAT(13) to CLIQUE in [8]. We state the following theorems first.

THEOREM 1. *There exists a fixed positive constant ϵ for which there is polynomial time reduction f from any NP problem to MAX-3SAT(13) s.t. YES instances map to satisfiable formulae and NO instances map to formulae in which less than $1 - \epsilon$ fraction of clauses can be satisfied.*

Proof: See the proof of Theorem 6.2 of [8]. □

THEOREM 2. *There is a reduction f from instances of MAX-3SAT(13) to instances of Vertex Cover with the the following properties:*

1. *Satisfiable formulae map to graph G (with n vertices) in which vertex cover has size $\leq cn$.*
2. *There exists a positive constant ϵ such that unsatisfiable formula map to graph G (with n vertices) in which vertex cover has size $\geq c(1+\epsilon)n$.*
3. *c is a constant such that $c > \frac{1}{2}$, $c \cdot (1+\epsilon) > \frac{1}{2}$.*

Proof: The proof follows from the reduction in Theorem 6.22 of [8]. □

LEMMA 3. *There exist fixed positive constants c, d and a polynomial time reduction f from SAT to MAX-CLIQUE such that (1) $c > 2/3$ and $c - d > 2/3$, (2) f maps satisfiable formulae map to graphs $G = (V, E)$ where $w(G) \geq c|V|$, (3) f maps unsatisfiable formula to graphs $G = (V, E)$ where $w(G) \leq (c - d)|V|$, and (4) each vertex in G has degree $\geq |V| - 14$.*

Proof of Lemma 3 : Consider the reduction f of 3-SAT to VERTEX COVER as given in [4]. Let F be a formula which is an instance of 3-SAT. By construction of f, it is clear that F is mapped to an instance $G = f(F)$ of VERTEX COVER such that the degree of each vertex in G is ≤ 14. Now consider $G^c = (V, E^c)$. If F is satisfiable, then G has a vertex cover of size $c'n$. Equivalently, G^c has a clique of size $n - c'n = n(1 - c')$. Thus $w(G^c) \geq n(1 - c') = cn$(say). If F is not satisfiable, then by Theorem 2, there exists a constant ϵ such that the minimum vertex cover of G has size $\geq c'(1+\epsilon)n$. Thus every clique of G^c has size $\leq n - c'(1+\epsilon)n = n(c - c'\epsilon)$. □

LEMMA 4. *There is a reduction f from MAX-3SAT(13) to $\frac{2}{3}$CLIQUE with the property that*

$$\begin{array}{rcl} Instance\, I \in MAX-3SAT(13) & \Rightarrow & OPT(f(I)) \geq \frac{2n}{3} \\ Instance\, I \notin MAX-3SAT(13) & \Rightarrow & OPT(f(I)) < \frac{(2-\epsilon)n}{3} \end{array}$$

where, $f(I)$ is a graph $G = (V, E)$ with $|V| = n$ and $|E| > \frac{8 \cdot n^2 - 18 \cdot n}{18}$ and ϵ is a small constant > 0.

Proof of Lemma 4: The reduction is almost the same as in Lemma 3. Let $G^c = (V^c, E^c)$ be the complement of the instance of the VERTEX COVER problem in Theorem 2. We obtain $G = (V, E)$ by adding $n' = (3 \cdot c - 1) \cdot |V^c|$ (c is the constant mentioned in Theorem 2). The new vertices are connected to each other and to every vertex in V^c. $n = |V| = |V^c| + n' = 3 \cdot |V^c|$. Now,

$$\begin{array}{rcl} |E| & \geq & \frac{|V^c|(|V^c|-14)}{2} + \frac{n'(n'-1)}{2} + n' \cdot |V^c| \\ & > & \frac{8 \cdot n^2 - 18 \cdot n}{18} \end{array}$$

From the construction, it is obvious that $w(G) = w(G^c) + (3 \cdot c - 1) \cdot |V^c|$.

1. Let $I \in MAX - 3SAT(13)$.

$$\begin{array}{lll} w(G^c) \geq & |V^c| - c \cdot |V^c|. & \\ \Rightarrow w(G) = & \frac{2}{3} \cdot c \cdot |V^c| = & \frac{2}{3} \cdot |V| \end{array}$$

Hence

$$I \in MAX - 3SAT(13) \quad \Rightarrow \quad OPT(f(I)) \geq \frac{2n}{3}$$

2. Let $I \notin MAX - 3SAT(13)$.

$$\begin{array}{lll} w(G^c) \leq & |V^c| - c(1+\epsilon) \cdot |V^c|. & \\ \Rightarrow w(G) = & \frac{2}{3} \cdot c \cdot |V^c| = & \frac{2}{3} \cdot |V| \end{array}$$

$$\Rightarrow I \notin MAX - 3SAT(13) \quad \Rightarrow \quad OPT(f(I)) < \frac{(2-\epsilon)n}{3}$$

□

5. HARDNESS OF APPROXIMATING QO_N AND QO_H

5.1 Hardness of Approximating QO_N

In this section, we present a gap preserving reduction from instances of MAX-CLIQUE to QO_N with a gap of $2^{log^f K}$, where K is the optimal cost of the mapped instance of QO_N.

The reduction from an instance of the MAX-CLIQUE problem to the QO_N is defined as follows. Let $G = (V, E)$ be an instance of MAX-CLIQUE. We construct an instance $Q = (Y, \overline{S}, \overline{T}, W)$ with the following properties. The query graph Y is set to be identical to the input instance G. The selectivity of each edge is set to a constant, that is, $s_i = \frac{1}{\alpha}$, where $\alpha \geq 4$ is some constant. For every vertex $v \in V$, the size of the relation corresponding to that vertex which is denoted by T_i is set equal to $\alpha^{(c - \frac{d}{2})n}$, where c and d are constants obtained from Lemma 3. For each edge $\{j, k\} \in E$, w_{jk} is set to $\omega(n)$, which is some polynomially computable

function of n, the number of vertices of G. Note that this construction implies that for every edge $\{j,k\} \in E$, w_{jk} is independent of j and k. For vertex pairs $\{j,k\}$ that are not edges we set $w_{jk} = T_j$.

We now establish some simple properties of the reduction and present some notation that helps us in doing so. Let Z be any sequence for an instance of QO_N whose query graph $Y = (V, E)$. The term $D_i(Z)$ denotes the number of edges in the subgraph of the query graph Y of Q formed by the vertices that occupy the first i positions of Z. We let the term $K_{c,d}(n)$ to be $\omega(n) \cdot \alpha^{\frac{[(c-\frac{d}{2})n][(c-\frac{d}{2})n+1]}{2}+1}$.

Let Z be the sequence Xv_jY, where $|X| = i$. Recall that $H_i(Z) = N(X) \cdot min\{w_{jk} | v_k \text{ appears in } X\}$. By construction of the reduction mapping, $H_i(Z) = \omega(n) \cdot N(X)$. With respect to the sequence Z, call an edge $\{v_j, v_k\} \in E$ a *back edge* of vertex v_j with respect to the sequence Z if v_k appears earlier than v_j in Z. If the i^{th} position of Z is occupied by vertex v_j, then $B_i(Z)$ denotes the number of back-edges of v_j.

For $1 \leq i \leq n-1$, $H_{i+1}(Z) = N(Xv_j) \cdot \omega(n) = N(X) \cdot \omega(n) \cdot \alpha^{(c-\frac{d}{2}) \cdot n - B_{i+1}(Z)}$. It therefore follows that $H_i(Z) = N(X) \cdot \omega(n) = \alpha^{(c-\frac{d}{2}) \cdot n \cdot i - D_i(Z)}$. Now, let $H_i(Z) < H_j(Z)$. Then $H_i(Z) \leq (\frac{1}{\alpha}) \cdot H_j(Z), 1 \leq i \leq n$; since, $H_i(Z) = \omega(n) \cdot \alpha^{z_i}$ for some integer z_i.

LEMMA 5. *For $i \geq cn, H_{i+1}(Z) \leq (\frac{1}{\alpha}) \cdot H_i(Z)$.*

Proof: As noted in the discussion above, $H_{i+1}(Z) = H_i(Z) \cdot \alpha^{(c-\frac{d}{2}) \cdot n} \cdot (\frac{1}{\alpha})^{B_{i+1}(Z)}$. Since the number of nodes with which v_i cannot have edges is at most 14, it follows that

$$\begin{aligned} B_{i+1}(Z) &\geq i - 14 \\ &\geq cn - 14 \\ &= (c - \tfrac{d}{2})n + 1 + (\tfrac{d}{2})n - 15 \\ &> (c - \tfrac{d}{2})n + 1 \text{ assuming that } \tfrac{d}{2})n - 15 > 0. \end{aligned}$$

It follows that

$$H_{i+1}(Z) < H_i(Z) \cdot (\frac{1}{\alpha})$$

□

LEMMA 6. *Let G be an instance of MAX-CLIQUE s.t. $w(G) \geq c \cdot n$. and suppose that the reduction function maps the graph G to the instance Q of QO_N. Then $OPT(Q) \leq K_{c,d}(n)$.*

Proof: We consider a sequence Z, in which the first $c \cdot n$ nodes are the nodes of the clique, which is of size $c \cdot n$. Since $B_{i+1} = i$ for $1 \leq i \leq c \cdot n - 1$ the following is follows from the Observation 2.

$$\begin{aligned} H_{(c-\frac{d}{2})n+1} &= H_{(c-\frac{d}{2})n} > \ldots > H_1 \quad (1) \\ H_{cn} &< \ldots < H_{(c-\frac{d}{2})n+1}. \quad (2) \end{aligned}$$

¿From equation (2) above and Lemma 5, we can deduce the following

$$H_{n-1} < \ldots < H_{cn} \ldots < H_{(c-\frac{d}{2})n+1} \quad (3)$$

$$\begin{aligned} C(Z) &= \textstyle\sum_{i=1}^{(c-\frac{d}{2})n} H_i + \sum_{i=(c-\frac{d}{2})n+1}^{n-1} H_i \\ &\leq H_{(c-\frac{d}{2})n}[1 + (\tfrac{1}{\alpha}) + (\tfrac{1}{\alpha})^2 + \ldots] + \\ &\quad H_{(c-\frac{d}{2})n+1}[1 + (\tfrac{1}{\alpha}) + (\tfrac{1}{\alpha})^2 + \ldots] \\ &\leq 2\tfrac{\alpha}{\alpha-1} \cdot H_{(c-\frac{d}{2})n}, \text{ since } H_{(c-\frac{d}{2})n} = H_{(c-\frac{d}{2})n+1} \\ &\leq \alpha \cdot H_{(c-\frac{d}{2})n}, \textit{ since } \alpha \geq 4 \\ &\leq \omega(n) \cdot \alpha^{\frac{[(c-\frac{d}{2})n][(c-\frac{d}{2})n+1]}{2}+1} = K_{c,d}(n). \end{aligned}$$

□

LEMMA 7. *Let $G = (V, E)$ be an undirected graph such that $|V| = n$. Then $|E| \leq \frac{n \cdot (n-1)}{2} - n + w(G)$.*

Proof: Let $V_1 = \{v_{i,1}, v_{i,2}, \ldots, v_{i,w(G)}\}$ be a clique of size $w(G)$ in G. For each vertex $v_k \in V - V_1$, there exists an edge $\{v_j, v_k\} \notin E$ for some $v_j \in V_1$ (otherwise $V_1 \cup \{v_k\}$ will form a clique). Thus, $|E| \leq \frac{n \cdot (n-1)}{2} - |V - V_1| \leq \frac{n \cdot (n-1)}{2} - n + w(G)$.
□

LEMMA 8. *If G is a graph with the size of the maximum clique in $G \leq (c - d) \cdot n$ and Q be the instance of QO_N obtained by applying the reduction f to G. Then $OPT(Q) \geq K_{c,d}(n) \cdot \alpha^{(\frac{d}{2})n-1}$.*

Proof: Consider an optimal sequence $Z = R_1, R_2, ...R_n$ of the nodes. Consider the subgraph G_1 induced by the nodes $R_1, R_2, \ldots, R_{(c-\frac{d}{2})n}$ and let m be the size of the largest clique in G_1. By hypothesis about clique size, $m \leq (c-d)n$. Therefore it follows from Lemma 7 that

$$D_{(c-\frac{d}{2})n}(Z) \leq (c - \tfrac{d}{2})n][(c - \tfrac{d}{2})n - 1]/2 - (\tfrac{d}{2})n$$

It therefore follows that

$$\begin{aligned} C(Z) &\geq H_{(c-\frac{d}{2})n} \cdot \alpha^{[(c-\frac{d}{2})n][(c-\frac{d}{2})n] - |D_{(c-\frac{d}{2})n}(Z)|} \\ &\geq \alpha^{\frac{[(c-\frac{d}{2})n][(c-\frac{d}{2})n+1]}{2} + (\frac{d}{2})n} \\ &\geq K \cdot \alpha^{(\frac{d}{2})n-1}. \end{aligned}$$

□

Lemmas 6 and Lemma 8 imply the following theorem, which formally states the complexity of the approximation problem.

THEOREM 9. *There is a positive constant ϵ such that there is a gap preserving reduction from MAX-3SAT(13) to QO_N such that*
(a) Satisfiable formulas map to instances of QO_N where optimal cost $\leq K_{c,d}(n)$
(b) Unsatisfiable formulas map to instances of QO_N where optimal cost $\geq K_{c,d}(n) \cdot \alpha^{n \cdot \epsilon}$,
where n is the size of the query graph produced by the reduction. □

Note that α can be any number ≥ 4 whose size (in number of bits) is some polylogarithmic function of the size of the input instance. By choosing α to be 4, we obtain $K_{c,d}(n)$ to be approximately $4^{\theta(n^2)}$. Thus, the gap produced is $4^{n \cdot \epsilon}$ which is $2^{\theta(log^{0.5}K)}$, that is, solving the QO_N problem approximately with a better competitive ratio than $2^{\theta(log^{0.5}K)}$ using a polynomial time algorithm would mean that P=NP. By increasing α, the gap expression remains of the form $2^{\theta(log^{h}K)}$, with h tending to 1, or equivalently, the gap expression can be made to equal $2^{\theta(log^{1-\delta}K)}$ for any constant $\delta > 0$.

5.2 Hardness of Approximating QO_H

In this section, we prove that the problem of approximating QO_H within a polylogarithmic factor of the optimal cost is NP-Hard by giving a polynomial time reduction, f from of the $\frac{2}{3}$CLIQUE problem to the QO_H problem.

We now give the reduction from $\frac{2}{3}$CLIQUE to QO_H. Let $G = (V, E)$ be an instance, I of $\frac{2}{3}$CLIQUE, where $V = \{v_1, v_2, \ldots, v_m\}$ and $E = \{e_1, e_2, \ldots, e_p\}$. The instance, $f(I)$ of QO_H is constructed as follows.

1. The query graph is $G' = (V', E')$, where $V' = V \cup \{v_0\}$ and $E' = E \cup \{e_{p+1}, e_{p+2}, \ldots, e_{p+m}\}$. The vertex v_0 corresponds to the relation R_0. Edge e_{p+i} means that there is a join predicate between relation R_0 and relation R_i.

2. The page size $PS = (2^m + p + m) \cdot d$, where d is any even number.

3. Let $B = 2^{(m-1)}$. The number of tuples in relation R_0, n_0 is $m^2 \cdot B^2$. The number of tuples in relation R_i, that is, n_i is B, for $1 \leq i \leq m$.

4. The number of pages in each relation R_i, namely, b_i is the same as n_i, for $0 \leq i \leq m$.

5. The selectivity of the join predicate between any two relations R_i and R_j is $s = B^{-\frac{2}{m-1}} = \frac{1}{4}$, for $1 \leq i, j \leq m$ and $i \neq j$. The selectivity of the join predicate between relations R_0 and R_i, for $1 \leq i \leq m$ is 1 (i.e. a cartesian product).

6. The available main memory $M = (\frac{m}{3} - 1) \cdot B$ pages.

7. $K = 16 \cdot m^2 \cdot B^2 \cdot B^{(\frac{2m}{3})} \cdot s^{\frac{(\frac{2m}{3}) \cdot (\frac{2m}{3} - 1)}{2}}$.

Some Comments and Observations

1. We assume here that the query is of the following form.

```
Select R_0.a
from R_0, R_1, R_2, ..., R_m
where
predicates between R_0 and each of R_1 to R_m
and predicates among R_1 to R_m.
```

   The attribute a of R_0 is of size $(2^m) \cdot d$. Every other attribute involved in join predicates is of size d. A consequence of this is that, once a join is performed with R_0, every tuple in all subsequent intermediate results and the final result occupies one page(no tuple can be split across a page).

2. The size, in pages, of the relation R_0 has been set to $m^2 \cdot B^2$. The consequence of this is that, R_0 can never be an inner relation or the build relation during the execution of a hash join operation. This is because the minimum amount of memory required for R_0 to be an inner relation is $\geq \sqrt{b_0} = \sqrt{m^2 \cdot B^2} = m \cdot B > (\frac{m}{3} - 1) \cdot B = M$. So, R_0 is the outermost most relation, i.e. before the first pipeline, since we are considering only linear trees of execution here.

3. The basic idea used here is that we get the least cost sequence when we prefix the sequence with relations such that the vertices corresponding to those relations form a clique in the instance I of the $\frac{2}{3}$CLIQUE problem.

LEMMA 10. *If the instance I of $\frac{2}{3}$CLIQUE has a clique C of size $\geq \frac{2m}{3}$ then the instance $f(I)$ of QO_H has a sequence of pipelines S' and a memory allocation vector m' such that $C_{seq}(S', m', M) \leq K$.*

Proof: The solution for $f(I)$ has three pipelines. R_0 is the outermost relation. There are $\frac{m}{3}$ relations in the first pipeline, $\frac{m}{3} + 1$ relations in the second pipeline and $\frac{m}{3} - 1$ relations in the last pipeline - $P(1, \frac{m}{3})$, $P(\frac{m}{3}+1, \frac{2m}{3}+1)$ and $P(\frac{2m}{3} + 2, m)$ is the sequence of pipelines. The parameter min is the minimum memory that should be allocated to the join.

The last join in the pipeline $P(\frac{m}{3} + 1, \frac{2m}{3} + 1)$ is with a relation R_q which is not present in C. Consider the number of edges between the vertex v_q and the vertices corresponding to the relations involved in the joins preceding the join with R_q. The number of edges among vertices $v_l \in C$ is $\frac{(\frac{2m}{3}) \cdot (\frac{2m}{3} - 1)}{2}$. The total number of edges is $> \frac{8 \cdot m^2 - 18 \cdot m}{18}$. So, there are $> \frac{4 \cdot m^2 - 12 \cdot m}{18}$ edges remaining. Let the vertices corresponding to the last $\frac{m}{3}$ relations i.e. the vertex v_q and the $\frac{m}{3} - 1$ vertices corresponding to the relations involved in joins in the last pipeline be a part of a clique, say Q. The number of edges in $Q = \frac{(\frac{m}{3}) \cdot (\frac{m}{3} - 1)}{2}$. Now the number edges remaining is $> \frac{(3 \cdot m^2 - 9 \cdot m)}{18}$, which are the edges between vertices in C and vertices in Q. Since the number of vertices in Q is $\frac{m}{3}$, on an average, each vertex in Q should be connected to $> \frac{\frac{3 \cdot m^2 - 9 \cdot m}{18}}{\frac{m}{3}} > \frac{m-3}{2}$ vertices in C. Or in other words there should exist at least one vertex which is connected to $\geq \frac{m-3}{2} + 1 = \frac{m-1}{2}$ vertices in C. The relation R_q is chosen to correspond to such a vertex v_q. Now it can be verified that the number of pages materialized at the end of pipeline is

$$\begin{aligned} &\leq m^2 \cdot B^2 \cdot B^{(\frac{2m}{3}+1)} \cdot s^{\frac{(\frac{2m}{3})(\frac{2m}{3}-1)}{2}} \cdot s^{\frac{m-1}{2}} \\ &= m^2 \cdot B^2 \cdot B^{(\frac{2m}{3})} \cdot s^{\frac{(\frac{2m}{3}) \cdot (\frac{2m}{3} - 1)}{2}}. \end{aligned}$$

We now briefly describe the total cost of executing the three pipelines by giving the dominating costs of executing each of the pipelines.

1. The cost of executing the pipeline $P(1, \frac{m}{3})$ is dominated by the cost of writing the intermediate result to the disk, which is $m^2 \cdot B^2 \cdot B^{(\frac{2m}{3})} \cdot s^{\frac{(\frac{2m}{3})(\frac{2m}{3}-1)}{2}}$.

2. The cost of executing the pipeline $P(\frac{m}{3}, \frac{2m}{3} + 1)$ consists of the cost of reading the intermediate result + the cost of executing the join $j_{\frac{m}{3}+1}$ + the cost of executing the join $j_{\frac{2m}{3}+1}$ + the cost of writing the intermediate result to the disk, which amounts to $6 \cdot m^2 \cdot B^2 \cdot B^{(\frac{2m}{3})} \cdot s^{\frac{(\frac{2m}{3})(\frac{2m}{3}-1)}{2}}$.

3. The cost of executing the pipeline $P(\frac{2m}{3} + 2, m)$ is dominated by the cost of reading the intermediate re-

sult materialized by $P(\frac{m}{3}, \frac{2m}{3}+1)$. This cost can be verified to be $\leq m^2 \cdot B^2 \cdot B^{(\frac{2m}{3})} \cdot s^{\frac{(\frac{2m}{3})(\frac{2m}{3}-1)}{2}}$.

The total cost of executing the three pipelines is $\leq K$. Hence the proof. □

LEMMA 11. *If the instance I of the $\frac{2}{3}$CLIQUE problem has no clique of size $\geq \frac{2-\epsilon}{3} \cdot m$ (ϵ is a very small constant < 1) then for the instance $f(I)$ of QO_H there exist no sequence of pipelines S' and memory allocation vector m' such that $C_{seq}(S', m', M) \leq K \cdot 2^{\frac{\epsilon}{3} \cdot (\frac{9}{2})^{\frac{1}{2}} \cdot (logK-4)^{\frac{1}{2}} - 4}$.*

Proof: We first make the following observation - any intermediate result which is the product of $> \frac{m}{3}$ and $< \frac{2m}{3}$ joins should not be materialized. If there exists a pipeline which ends with the join j_k, where $\frac{m}{3} < k < \frac{2m}{3}$, the cost of reading and writing the intermediate result is

$$\begin{aligned}
&\geq 2 \cdot m^2 \cdot B^2 \cdot B^{(\frac{m}{3}+1)} \cdot s^{\frac{(\frac{m}{3}) \cdot (\frac{m}{3}+1)}{2}} \\
&= 2 \cdot m^2 \cdot B^2 \cdot B^{(\frac{2m}{3}-1)} \cdot s^{\frac{(\frac{2m}{3}-1) \cdot (\frac{2m}{3}-2)}{2}} \\
&= 2 \cdot m^2 \cdot B^2 \cdot B^{(\frac{m}{3})} \cdot s^{\frac{(\frac{m}{3}) \cdot (\frac{m}{3}-1)}{2}} \cdot B \cdot s^{(\frac{m}{3})} \\
&= 2 \cdot m^2 \cdot B^2 \cdot B^{(\frac{m}{3})} \cdot s^{\frac{(\frac{m}{3}) \cdot (\frac{m}{3}-1)}{2}} \cdot 2^{\frac{(m-3)}{3}} \\
&= K \cdot 2^{\frac{(m-3)}{3} - 3} \\
&\gg K \cdot 2^{\frac{\epsilon}{3} \cdot (\frac{9}{2})^{\frac{1}{2}} \cdot (logK-4)^{\frac{1}{2}} - 4}.
\end{aligned}$$

The consequence of this is that, in any join sequence, the joins starting with the join $j_{\frac{m}{3}+1}$ till the join $j_{\frac{2m}{3}-1}$ should be performed in a single pipeline, say L. Otherwise, we have a total cost which is $\gg K \cdot 2^{\frac{\epsilon}{3} \cdot (\frac{9}{2})^{\frac{1}{2}} \cdot (logK-4)^{\frac{1}{2}} - 4}$. Let the pipeline L end with the join j_i. Now, $i \geq \frac{2m}{3}$.

1. $i = \frac{2m}{3}$. We have no clique of size $\geq \frac{2-\epsilon}{3} \cdot$ in the instance I. It has been shown that the number of edges in the subgraph comprising of the any subset of vertices , $\frac{2}{3}$ in number is $< \frac{\frac{2m}{3} \cdot \frac{2m}{3}-1}{2} - \frac{\epsilon}{3} \cdot m$ i.e. the number of edges is $\frac{\epsilon}{3} \cdot m$ less than the number of edges in a clique of size $\frac{2m}{3}$. The materialization cost is

$$\begin{aligned}
&\geq m^2 \cdot B^2 \cdot B^{(\frac{2m}{3})} \cdot s^{\frac{(\frac{2m}{3}) \cdot (\frac{2m}{3}-1)}{2}} \cdot s^{-\frac{\epsilon}{3} \cdot m} \\
&= K \cdot 2^{2 \cdot \frac{\epsilon}{3} \cdot m - 4} \\
&> K \cdot 2^{\frac{\epsilon}{3} \cdot (\frac{9}{2})^{\frac{1}{2}} \cdot (logK-4)^{\frac{1}{2}} - 4}.
\end{aligned}$$

2. $i > \frac{2m}{3}$. Consider the input number of pages to the join $j_{\frac{2m}{3}+1}$ and the cost of executing it. Since there is no clique of size $\geq \frac{2m}{3}$ in the instance I, the cost of performing the join is

$$\begin{aligned}
&\geq \beta \cdot m^2 \cdot B^2 \cdot B^{(\frac{2m}{3})} \cdot s^{\frac{(\frac{2m}{3}) \cdot (\frac{2m}{3}-1)}{2}} \cdot s^{-\frac{\epsilon}{3} \cdot m} \cdot g(min, B) \\
&\quad + h(min, B) \\
&> K \cdot 2^{\frac{\epsilon}{3} \cdot (\frac{9}{2})^{\frac{1}{2}} \cdot (logK-4)^{\frac{1}{2}} - 4}.
\end{aligned}$$

Since $\leq B$ pages are to shared between the joins $j_{\frac{2m}{3}}$ and $j_{\frac{2m}{3}+1}$, the best allocation gives min pages to the join $j_{\frac{2m}{3}+1}$ and $\leq B - min$ pages to the join $j_{\frac{2m}{3}}$. It should be noted that $g(min, B) \geq B^{-\frac{\epsilon}{3}}$.

Hence the proof. □

THEOREM 12. *Approximating QO_H to within a factor of $2^{\alpha \cdot (logK-4)^{\frac{1}{2}} - 4}$ is NP-Hard. Here K is the optimal cost of executing the query and α is a constant.* □

Proof: Follows from Lemmas 10 and 11 that

$$\begin{aligned}
Instance\, I \in \tfrac{2}{3}CLIQUE &\Rightarrow OPT(f(I)) \leq K \\
Instance\, I \notin \tfrac{2}{3}CLIQUE &\Rightarrow OPT(f(I)) > \\
&\quad K \cdot 2^{\frac{\epsilon}{3} \cdot (\frac{9}{2})^{\frac{1}{2}} \cdot (logK-4)^{\frac{1}{2}} - 4}.
\end{aligned}$$

The constant α mentioned in the statement of Theorem 12 $= \frac{\epsilon}{3} \cdot (\frac{9}{2})^{\frac{1}{2}}$. □

5.3 Hardness of Approximability under further restrictions on QO_N problem

The previous section proves a basic hardness result concerning finding approximately optimal plans. However, the reductions in the previous section result in dense graphs, which raises the question that perhaps approximation algorithms with better competitive ratios could be constructed for sparse query graphs. We consider this question in this section, and answer it negatively, that is, we prove that for all query graphs where the number of edges $e(n)$ is a given function of the number of vertices n, where $3n \leq e(n) \leq n^2/3$, the approximation problem for the QO_N problem remains equally difficult.

THEOREM 13. *Given a function $e(m)$, $3m \leq e(m) \leq \frac{m^2}{3}$ such that we can find a graph with $e(m)$ edges in polynomial time. Then there is a gap preserving reduction from instances of MAX-3SAT(13) to instances of QO_N where the query graph has m vertices and $e(m)$ edges such that*
(a) Satisfiable formulae map to instances of QO_N where optimal cost $\leq K_{c,d}(n)$, and
(b) Unsatisfiable formulas map to instances of QO_N where optimal cost $\geq K_{c,d}(n) \cdot \alpha^{n \cdot \epsilon}$,
where $n = \sqrt{m}$.

Proof: Let ($G' = V'$,E') be our original query graph formed in the reduction from MAX-3SAT(13) to MAX-CLIQUE and let $|V'| = n$. Let $G'' = (V'', E'')$ be a graph with $n^2 - n$ vertices and $e(n^2) - |E'| - 1$ edges. Clearly, for any value of $e(m)$, we can construct a graph with $e(m)$ edges which is connected. This follows since the number of edges is quite large, namely, $2(n^2 - n)$.

1. The query graph is $G = (V, E)$, where $V = V'' \cup V'$ and $E = E'' \cup E' \cup e'$. Here e' is an edge added to connect the components V'' and V'. The number of edges in V is n^2 and edges is $e(n^2)$.

2. Consider any constant $\beta \geq 4$ such that β is bounded by a polynomial in n. Let $\alpha = \beta^{n^4}$. The size of each relation corresponding to nodes in V'' is same as in our reduction above and the size of each relation in V'' is β^{n^2}.

3. The selectivity factor of each edge is $\frac{1}{\alpha}$ for each edge in E' and $(\frac{1}{\beta})$ for each edge in E''. The selectivity factor of e' is also $(\frac{1}{\beta})$.

4. $w_{jk} = \omega(n)$ for each edge.

All the the new nodes can account for a factor of at most $\beta^{n^4} = \alpha$ in the cost. Further, all the edges can account for a factor of at most $(\frac{1}{\beta})^{e(n^2)} \leq \beta^{n^4} = \alpha$ in the cost. By extending the graph, we have introduced a factor of at most α^2 to the cost. The gap thus becomes $\alpha^{(c-\frac{d}{2})\cdot n-2}$, which is $\alpha^{\epsilon \cdot n}$, for some ϵ. . □

5.4 QO_H and Sparse Graphs

It can be observed that the constructed instance of QO_H is dense. In this section we extend the proof to sparse graphs. The reduction is almost the same as the previous reduction. It is as follows.

1. The query graph is $G' = (V', E')$, where the vertex set $V' = V \cup \{v_0\} \cup \{v'_1, v'_2, \ldots, v'_{m^2-m}\}$ and the edge set $E' = E \cup \{e_{p+1}, e_{p+2}, \ldots, e_{p+m}\} \cup \{e'_1, e'_2, \ldots, e'_{m^2-m}\}$. The vertex v_0 corresponds to the relation R_0. Edge e_{p+i} means that there is a join predicate between relation R_0 and relation R_i. The vertex v'_i corresponds to the relation R'_i. Edge e'_i means that there is a join predicate between relation R_0 and relation R'_i.

2. The page size $PS = (2^m + p + m) \cdot d$, where d is any even number.

3. Let $B = 2^{(m-1)}$. The number of tuples in relation R_0, n_0 is $m^2 \cdot B^2$. The number of tuples in relation R_i, n_i is B, for $1 \leq i \leq m$. The number of tuples in relation R'_j is $\frac{(\frac{m}{3}-1)\cdot B}{m^2-m}$, for $1 \leq j \leq m^2 - m$.

4. The number of pages in each relation R_i, b_i is the same as n_i, for $0 \leq i \leq m$. The number of pages in each relation R'_j is the same as the number of tuples, for $1 \leq j \leq m^2 - m$.

5. The selectivity of the join predicate between any two relations R_i and R_j is $s = B^{-\frac{2}{m-1}} = \frac{1}{4}$, for $1 \leq i, j \leq m$ and $i \neq j$. The selectivity of the join predicate between relations R_0 and R_i, for $1 \leq i \leq m$ is 1. The selectivity of the join predicate between relations R_0 and R'_j is $\frac{m^2-m}{(\frac{m}{3}-1)\cdot B}$, for $1 \leq j \leq m^2 - m$.

6. The available main memory $M = (\frac{m}{3} - 1) \cdot B$ pages.

7. $K = 16 \cdot m^2 \cdot B^2 \cdot B^{(\frac{2m}{3})} \cdot s^{\frac{(\frac{2m}{3})\cdot(\frac{2m}{3}-1)}{2}}$.

We now sketch the proofs very briefly and give a few observations.

1. We now have the following relation - $|E'| \leq \frac{3}{2} \cdot |V'|$.

2. The only change in Lemma 10 is that, instead of three pipelines we have four pipelines. Again, R_0 is the outermost relation. The first pipeline consists of the joins between relations R_0 and R'_i, $1 \leq j \leq m^2 - m$, with each join allocated the maximum requirement. The last three pipelines are the same as in Lemma 10. The number of pages materialized at the end of the first pipeline can be evaluated to be $m^2 \cdot B^2$. The cost of executing the four pipelined sequence is $\leq K$.

3. Lemma 11 remains the same. Any mention of a join in Lemma 11 should be taken to be a join among relations R_0 and R_i, $1 \leq i \leq m$.

4. The proof of Theorem 12 also remains the same. This idea can be extended to have the following relation

$$|E'| \leq fn(|V'|)$$

where, $fn(|V'|)$ is a given function of $|V'|$, except for very sparse or very dense graphs.

6. SUMMARY OF THE COMPLEXITY OF VARIANTS OF QO PROBLEM

In this section, we present a summary of the proofs for different variants of the query optimization problem we considered.

In the previous sections, we showed that approximating QO_N and QO_H to within a polylogarithmic factor of the optimal cost is NP-Hard. However, it should be noted that the query graphs of the constructed instances are dense. In the appendix, we show that the problems are equally hard even when the number of edges in the query graph is a given function of the number of vertices.

The final problem we consider, namely SQO−CP is derived from a long standing problem first posed by Ibaraki and Kameda [5]. They considered the following QO problem, in which (a) the query graphs are restricted to be tree queries, and (b) execution space consists of join sequences where a join could be computed by either the nested loops method or the merge-sort method. The complexity of this problem is open. In this paper, we show that this problem is NP-complete by considering a restriction of this problem, called SQO−CP(Star Query Optimization without Cartesian Products) in which the query graph space consists of star shaped graphs. This implies that Ibaraki and Kameda's conjecture is true unless P = NP. Variants of the problem, which take memory sensitivity of the cost function into account is also proved to be NP-complete[3].

7. CONCLUSIONS

In this paper, we study the complexity of the query optimization problem, primarily from the perspective of finding approximately optimal plans.

We show that for any $\delta > 0$, the problem of finding a join order sequence whose cost is within a factor $2^{log^{1-\delta}(K)}$ of K, where K is the cost of the optimal join order sequence is NP-hard. Results of a very similar nature are obtained for variants of the problem in which joins are executed using hash join methods, or both nested loops and hash join methods. Further, the result remains true, when the number of edges in the query graph Q is constrained to be some given function $3n \leq f(n) \leq n^2/3$ of n, where n is the number of vertices in the query graph. Essentially, these results argue that the query optimization problem is inherently non-approximable to within any polylogarithmic factor of the optimal cost in polynomial time (unless $P = NP$).

We also consider the complexity of the problem of computing an optimal join order sequence for star queries, when both nested loops and merge-sort joins are allowed as join methods, and prove that this problem is NP-complete.

An avenue of future work is the design of approximation algorithms for the query optimization problem for interesting subsets of the problem. Another avenue could be to improve on the gaps of non-approximability that we obtain in this paper.

8. ADDITIONAL AUTHORS

9. REFERENCES

[1] S. Chatterji and M.D. Yemmanuru, "On finding approximate solutions to the query optimization problem", Bachelor's Thesis Report, IIT Kanpur, 2001. Available from the URL: www.cse.iitk.ac.in/research/btp2001/aqon.html.

[2] S.Cluet and G.Moerkotte. "On the complexity of generating optimal left-deep processing trees with cross products", In Proc. Int. Conf. on Database Theory (ICDT), pages 54-67, 1995.

[3] S.S.K. Evani. " On the complexity of database query optimization", Master's Thesis, IIT Kanpur, 2001. Available from the URL: www.cse.iitk.ac.in/research/mtech99/9911133.html.

[4] M.R.Garey and D.S.Johnson, "Computers and Intractability : A Guide to the Theory of NP-Completeness," W.H.Freeman, San Francisco, 1979.

[5] Toshihide Ibaraki and Tiko Kameda, "On the Optimal Nesting Order for Computing N-Relational Joins, ACM Transactions on Database Systems", Vol.9, No.3, September 1984, Pages 482-502.

[6] G. Graefe and W. McKenna, "Extensibility and Search Efficiency in the Volcano Optimizer Generator", In Proc. IEEE CS International Conference on Data Engineering, Vienna. April 1993.

[7] R. Krishnamurthy, H. Boral and C. Zaniolo, "Optimization of Non-recursive Queries", In Proceedings of the 1986 International Conference on Very Large Databases.

[8] Sanjeev Arora, "Probabilistic Checking of Proofs and Hardness of Approximation Algorithms", Technical Reports, Department of Computer Science, Princeton University.

APPENDIX

A. PROBLEM FORMULATION FOR SQO-CP

In this section, we formulate the star query optimization problem. The feasible sequences as defined by us do not allow cartesian products among relations. This problem, is referred to as the *Star Query Optimization* problem, or $SQO-CP$ (Star Query Optimization minus Cross Products) for short. We now present some notations and terminology.

A.1 Oriented Star Query Graph and Cost Parameters

A star query graph over the relations $R_0, R_1, \ldots, R_m$ has been defined as an undirected tree with no specified root. The central relation R_0 is distinguished by the property that it is the only vertex with degree > 1. An *oriented* star query graph is obtained by specifying any vertex of the query graph as the root. There are $m+1$ possible oriented trees, one each for the $m+1$ choices for the root. These $m+1$ oriented trees can be divided into two classes. The oriented tree obtained with R_r as the root is denoted by T_r. Every feasible sequence Z that begins with R_r corresponds to a traversal of T_r in which a parent node relation occurs earlier in the sequence than the child node relation, for $0 \leq r \leq m$.
Let P_i denote the predicate between R_0 and R_i, for $1 \leq i \leq m$. For any feasible sequence Z, this predicate is assumed to be solved during the join of either R_0 or R_i whichever occurs later in Z. If Z starts with R_i, then R_0 (in the form of S_0 or N_0) occurs second, and P_i gets solved during the join of R_0. Otherwise, R_i occurs later in the sequence and P_i gets solved at that point.

A.2 Cost Formula

Let A_i denote the I/O cost of sorting the relation R_i and leaving the result in the form of a stream in memory. The cost of sorting a relation R with b pages is calculated as

$$\text{sort-cost}(R) = \begin{cases} bk_s & \text{if } R \text{ is on disk} \\ b(k_s - 1) & \text{if } R \text{ is streaming into memory} \end{cases}$$

Thus, $A_i = b_i \cdot k_s$. The term k_s is the number of times a relation is read + written to disk during a 2-pass sort. Of course, if R is already sorted according to the join predicate, then R need not be sorted again. The I/O cost of a sort-merge join between relations R and S is estimated as

$$C_{sm}(R, S) = \text{sort-cost}(R) + \text{sort-cost}(S)$$

A comment on k_s being assumed constant. Since the goal in the paper is to prove the NP-completeness of the SQO−CP problem, we will sometimes restrict the scope of the problem as we proceed. One such restriction already introduced is to assume that k_s is a constant. In general, k_s is a variable that varies with the size of the relation to be sorted. We will show that in the constructed instance of the SQO−CP problem, all relation sizes (base and intermediate) are such that a 2-pass sort is required to sort them.

Let Z be a feasible sequence and X be any prefix of Z. The number of tuples in the output of a prefix X of Z, denoted by $n(X)$, is estimated in the standard way as follows. It is the product of the number of tuples in each of the relations occurring in X multiplied with the selectivities of the join predicates between all pairs of relations occurring in X.

A restriction on tuple sizes. We assume that the query is of the form

```
select R_0.{attribute list}
from R_0, R_1,...,R_m
where predicates
```

In the output attribute list, we assume that all join attributes of R_0 are included and no attribute from any other relation is included. A consequence of this is that the *tuple size of all the intermediate relations is the same once R_0 has been joined.* For ease of calculations, we assume that the tuple size of the output of the query is one page. This implies, that the tuple sizes of all intermediate sequences X in which R_0 occurs is also one page. Let $b(X)$ denote the output size of X in number of pages. This is estimated as follows.

$$\begin{array}{lcl} b(R_r) & = & b_r \\ b(X) & = & n(X) \quad \text{if } X \text{ contains at least 2 relations.} \end{array}$$

Let $Z = XS_iY$ be a feasible sequence. The **cost of the sort-merge join operator** S_i is estimated as

$$b(X) \cdot (k_s - 1) + A_i$$

Let $Z = XN_iY$ be a feasible sequence. The **cost of the nested-loops join operator** N_i is given by

$$\begin{cases} n(X) \cdot w_i & \text{if } i \neq 0 \\ n(X) \cdot w_{0,r} & \text{if } i = 0 \text{ and } Z \text{ starts with } R_r \end{cases}$$

The cost of a feasible sequence Z is denoted by $C(Z)$ and is defined as the sum of the costs of the individual join operators appearing in Z. More formally, let $Z = XY$ and let $D(X, Y)$ denote the cost of the suffix Y of Z. Thus, $C(Z) = D(\phi, Z)$. The function D is inductively defined as follows.

$$\begin{array}{lcl} D(\phi, R_rN_iY) & = & \begin{cases} b_0 + w_i \cdot n_0 + D(R_0N_i, Y) & r = 0 \\ b_r + w_{0,r} \cdot n_r + D(R_rN_0, Y) & r \neq 0 \end{cases} \\ D(\phi, R_rS_iY) & = & C_{sm}(R_r, R_i) + D(R_rS_i, Y) \\ D(W, S_iY) & = & b(W) \cdot (k_s - 1) + A_i + D(WS_i, Y) \\ D(W, N_iY) & = & n(W) \cdot w_i + D(WN_i, Y) \\ D(W, \phi) & = & 0 \end{array}$$

A.3 SQO-CP: Problem Specification

In this section, we formally specify the decision version of SQO−CP, by first specifying an instance of the problem followed by a statement of the problem.
INSTANCE

1. A number m. The star query consists of $m+1$ relations, $R_0, R_1, \ldots, R_m$, in which R_0 is the central relation.
2. A constant k_s representing the number of times a relation R is read and written for a 2-pass sort assuming that R is initially streaming into memory.
3. The page size P.
4. $(m+1)$-dimensional vector $(n_0, n_1, \ldots n_m)$, where n_i is the number of tuples in R_i.
5. $(m+1)$-dimensional vector $(b_0, b_1, \ldots b_m)$ where b_i is the size of R_i in pages.
6. $(m+1)$-dimensional vector $(A_0, A_2, \ldots A_m)$. A_i represents the cost of sorting disk resident relation R_i.
7. m-dimensional vector $(s_1, s_2, \ldots s_m)$ of non-negative real numbers where s_i represents the selectivity of the predicate between R_0 and R_i.
8. m-dimensional vector $(w_1, w_2, \ldots w_m)$. w_i represents the least cost of accessing the relation R_i to match a join predicate with R_0 in nested-loops method.
9. m-dimensional vector $(w_{0,1}, w_{0,2}, \ldots w_{0,m})$. $w_{0,i}$ represents the least cost of accessing R_0 to match a join predicate with R_i in nested-loops method.
10. A positive integer M.

QUESTION
Does there exist a feasible sequence Z such that $C(Z) \leq M$?

A.4 Problem Specification of SPPCS

In this section, we specify the SPPCS problem and then give a reduction from PARTITION to SPPCS. SPPCS is an abbreviation for Subset Product Plus Complement Sum.

The SPPCS problem is defined as follows.
INSTANCE. A set of m pairs of non-negative integers, $W = \{(p_1, c_1), (p_2, c_2), \ldots, (p_m, c_m)\}$ and a positive integer L.
QUESTION. Does there exist a set $A \subseteq \{1, 2, \ldots, m\}$ such that

$$\prod_{i \in A} p_i + \sum_{j \in \{1,\ldots,m\} - A} c_j \leq L \ ?$$

We present the definition of the version of the PARTITION problem used in this paper.

PARTITION
INSTANCE. A set $U = \{b_1, b_2, \ldots, b_n\}$ of non-negative integers such that $\sum_{i=1}^{n} b_i$ is an even number.
QUESTION. Does there exist a subset V of U such that

$$\sum_{i \in V} b_i = \sum_{j \in \{1,\ldots,n\} - V} b_j \ ?$$

The version of the PARTITION problem is NP-complete as the following argument indicates. The standard definition of the PARTITION problem [4] consists of an instance $U = \{b_1, b_2, \ldots, b_n\}$ of non-negative integers. Let $U' = \{2b_1, 2b_2, \ldots, 2b_n\}$ which polynomially reduces the given instance of PARTITION in the standard definition to the version used in the paper. Thus, the version of PARTITION used is NP-complete.

A.5 NP-completeness of SPPCS problem

In this section, we give the reduction from the partition problem to the $SPPCS$ problem. In order to do so, we present the constructed instance of SPPCS for a given instance of PARTITION. The following definitions are used in the construction.

Notation : For a real number $x \geq 0$, $f_q(x) : \Re \rightarrow Q$ is defined as $f_q(x) = \lfloor 2^q x \rfloor / 2^q$. Given n positive integers $b_1, b_2, \ldots, b_n$, the function $g_q(x)$ is defined as $g_q(x) = 2^q f_q(e^{x/2K})$, where $K = \sum_{i=1}^{n} b_i$. □
In other words, in the binary representation of $f_q(x)$, there are q bits after the binary point. Further, the binary representation of $f_q(x)$ agrees with the binary representation of x from the most significant bit upto the q-th bit after the binary point.

Given an instance $\{b_1, b_2, \ldots, b_n\}$ of PARTITION, we construct an instance of the SPPCS problem as follows. Let $K = \sum_{i=1}^{n} b_i$.

- $p = \lfloor log_2 K \rfloor + 1$, $q = 2p + 7 + n$
- $S = g_{nq}(K/2)$

- $m = 2n$
- for $i = 1, \ldots, n$, $p_i = g_q(b_i)$ and $c_i = 3SK + b_i S$
- for $i = n+1, \ldots, 2n-1$, $p_i = 2^{(i-n)q}$ and $c_i = (i-n)3SK$
- $p_{2n} = 2K$ and $c_{2n} = (2K)\left(\prod_{i=1}^{2n-1} p_i\right) + 1$
- $W = \{(p_1, c_1), (p_2, c_2), \ldots, (p_m, c_m)\}$
- $L = 3KS/2 + n(n-1)3KS/2 + 2K + SK$

It is clear that the above construction can be carried out in polynomial time. The proof that the above mapping is a many to one reduction may be found in [3].

B. PROOF OF NP-COMPLETENESS OF SQO-CP

In this section, we present a polynomial reduction of an instance of the SPPCS problem to an instance of the SQO−CP problem.

The given instance of SPPCS consists of m pairs of non-negative integers (p_1, c_1), (p_2, c_2), $\ldots$, (p_m, c_m) and a positive integer L. Without loss of generality, we may assume that $p_i \geq 2$ and $c_i \geq 1$, for $1 \leq i \leq m$.

The constructed instance of SQO−CP is as follows.

1. The query consists of $m+2$ relations, $R_0, R_1, \ldots, R_{m+1}$, with R_0 as the central relation.
2. $k_s = 4$.
3. Let $J = \left\{4 \cdot k_s \cdot \prod_{1=1}^{m} p_i\right\}^2$ and $U = \sum_{i=1}^{m} c_i + \prod_{i=1}^{m} p_i + 1$.
4. Let d be any even positive value (join attribute size). The pagesize $P = (m+1) \cdot d$.
5. The size of the relations in terms of number of tuples is as follows. $n_0 = 5J^2 \cdot U$, $n_i = (m+1) \cdot n_0 \cdot J^2 \cdot c_i$, for $1 \leq i \leq m$ and $n_{m+1} = (m+1) \cdot n_0 \cdot J^2 \cdot U$.
6. The size of the relations in terms of the number of pages is as follows. $b_i = n_i \cdot d/P = n_0 J^2 c_i$, for $1 \leq i \leq m$, $b_{m+1} = n_0 J^2 U$ and $b_0 = n_0 = 5J^2 U$.
7. The cost of a 2-pass sort of relation R_i is given by $A_i = b_i \cdot k_s$, for $0 \leq i \leq m+1$.
8. The selectivities of the predicate P_i (between R_0 and R_i), for $1 \leq i \leq m$ is $s_i = p_i/n_i$. $s_{m+1} = J/n_{m+1}$.
9. The unit cost of nested-loops access for relation R_i is given by $w_i = J \cdot k_s \cdot p_i$ for $1 \leq i \leq m$, $w_{m+1} = J^2 \cdot k_s$.
10. The unit cost of nested-loops access for relation R_0 to match a tuple from R_i is given by $w_{0,j} = n_0$ for $1 \leq i \leq m+1$.
11. $M = n_0 \cdot J^2 \cdot k_s(L+1) - 1$.

It is clear that the constructed instance has size polynomial in the size of the input instance of SPPCS and can be constructed by an algorithm that runs in polynomial time.

Suppose available memory $mem = n_0/2$ pages. The smallest and the largest relations among the base relations and all possible intermediate relations are R_0 and R_{m+1} respectively. Since $mem < b_0 < b_{m+1} < (mem)^2$, it follows that a 2-pass sort is needed for all relations during query processing. The proof that the above mapping forms a many to one reduction from the problem SPPCS to the problem SQO−CP may be found in [3].