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  
salkiran.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 $2^{\delta (\log^{1-\delta}(K))}$ 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

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, 7]. 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 be 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 restricted 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 $2^{\delta (\log^{1-\delta}(K))}$ 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
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 have 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 optimal (i.e., least) cost execution tree for each input instance. 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 when (a) the query graph is a star graph 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 QON, QOH. QON and QOH 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 QON

3.1.1 Cost model of QON

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

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

\[ N(X) = 1 \text{ if } X = \phi \]
\[ N(X \cdot v_j) = N(X) \cdot n(v_j) \cdot \prod_{v_i \text{ appears in } X} s_{ij} \]

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 \( Z = X v_j Y \) 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 based on corresponding to \( v_j \) is denoted by \( H_s(Z) \) and defined as

\[ H_s(Z) = N(X) \cdot \min\{w_{jk} \mid v_k \text{ appears in } X\} \]

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,

\[ C(Z) = \sum_{i=1}^{n-1} H_s(Z) \]

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 \( v_j \). 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 \( R_k \). The value \( w_{kj} \) is defined likewise with the roles of \( R_k \) and \( R_k \) reversed. If \( \{v_j, v_k\} \notin E \), then \( w_{jk} \) is defined to be \( T_j \).

6. A number \( K \).

An instance of QON can be summarized as a tuple \( (Q = (V, E), S, T, W) \), where \( S \) is the vector of edge selectivities \( (s_1, s_2, \ldots, s_p) \) \((p = |E|)\), \( T \) 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 specification for QOH

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 greater than \( \sqrt{b_S} \).

The cost of performing the hash join \( R \bowtie S \), can be of the general form

\[
\begin{align*}
&\begin{cases}
\beta \cdot b_S \cdot g(m, b_S) + h(m, b_S) \\
&\quad \text{for } m_{min} \geq m < b_S
\end{cases}
\end{align*}
\]

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. \( g(b_S, b_S) = 0 \),
the relation memory and joining it with the relation cost of partitioning the relation

The term concurrently with the available main memory distributed sequence

a sequence called 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 $j_i, j_k$ are performed in a single pipeline. A sequence of pipelines, $S'$ is de- scribed 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 $j_{i_{p+k-1}}$. 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 $C_{\text{pipe}}(j_i, j_k)$. The cost of executing the query is denoted by $C_{\text{seq}}(S', m', M)$, where $m' = (m_1, m_2, \ldots, m_n)$. It is simply the sum of the costs of executing each of the pipelines in the sequence of pipelines $S'$, as shown in Figure 1.

\[ \text{Figure 1: Pipelined execution of a query} \]

1. $g(m, b_S) \geq b_S^{\alpha}$, for all values of $m < b_S$, where $\varepsilon$ is a small constant $\ll 1$ mentioned later in Lemma 4,
2. $h(b_S, b_S) = b_S$,
3. $h(m_{\min}, b_S) = \gamma \cdot b_S$, where $\gamma$ is a constant,
4. $\beta$ is a constant $\geq 2$ 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 se- quence 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 $j_i, j_k$ 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 $j_{i_{p+k-1}}$. 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 $C_{\text{pipe}}(j_i, j_k)$. The cost of executing the query is denoted by $C_{\text{seq}}(S', m', M)$, where $m' = (m_1, m_2, \ldots, m_n)$. It is simply the sum of the costs of executing each of the pipelines in the sequence of pipelines $S'$, as shown in Figure 1.

\[ \text{Figure 1: Pipelined execution of a query} \]

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 $e_i$ 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 $C_{\text{seq}}(S', m', M) \leq K$?

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

\[ \begin{align*}
OPT(\Pi) \geq \epsilon \Rightarrow OPT(\Pi') \geq \epsilon' \\
OPT(\Pi) \leq \frac{\rho}{\rho} \Rightarrow OPT(\Pi') \leq \frac{\epsilon'}{\epsilon'}
\end{align*} \]

Given a boolean formula $\phi$ in 3-CNF form and an assignment $A$ of truth values to the variables, the expression $val(A/\phi)$.
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 $\epsilon$ such that the minimum vertex cover of $G$ has size $\geq c'(1 + \epsilon)n$. Thus every clique of $G^c$ has size $\leq n - c'(1 + \epsilon)n = n(c - c')$.

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

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

where, $f(I)$ is a graph $G = (V, E)$ with $|V| = n$ and $|E| > \frac{8.18n^2 - 18n}{8n^2 - 18n}$ and $\epsilon$ 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,

$$|E| \geq \frac{|V^c|(|V^c| - 14)}{2} + n' \cdot |V^c| > \frac{8.18n^2 - 18n}{8n^2 - 18n} \cdot |V^c|$$

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).

$$w(G^c) \geq \frac{|V^c| - c \cdot |V^c|}{2} \cdot |V^c|$$

$$\Rightarrow w(G) = \frac{2}{3} \cdot c \cdot |V^c| = \frac{2}{3} \cdot |V|$$

Hence

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

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

$$w(G^c) \leq \frac{|V^c| - (c - d) \cdot |V^c|}{2} \cdot |V^c|$$

$$\Rightarrow w(G) = \frac{2}{3} \cdot c \cdot |V^c| = \frac{2}{3} \cdot |V|$$

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

**5. HARDNESS OF APPROXIMATING QON AND QOH**

**5.1 Hardness of Approximating QON**

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

The reduction is from an instance of the MAX-Clique problem to the QON problem, defined as follows. Let $G = (V, E)$ be an instance of MAX-Clique. We construct an instance $Q = (Y, S, T, W)$ 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 be a constant, that is, $s_i = \frac{1}{\alpha}$, where $\alpha \geq 4$ is some constant. For every vertex $v \in \bar{V}$, the size of the relation corresponding to that vertex which is denoted by $T_i$ is set equal to $\alpha^{(c - \frac{1}{2})n}$, 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 \( QON \) 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 \( \omega(n) \cdot \alpha^{\frac{(c-\frac{d}{2})n}{2}} \).

Let \( Z \) be the sequence \( X v_j Y \), where \( |X| = 1 \). Recall that \( H_i(Z) = N(X) \cdot \min \{ \omega(v)|v \text{ appears in } X \} \). 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 \( 1 \leq i \leq n-1 \), \( H_{i+1}(Z) = N(X v_j) \cdot \omega(n) = N(X) \cdot \omega(n) = \alpha^{\frac{(c-\frac{d}{2})n}{2}} \cdot B_{i+1}(Z) \). It therefore follows that \( H_i(Z) = N(X) \cdot \omega(n) = \alpha^{\frac{(c-\frac{d}{2})n}{2}} \cdot B_i(Z) \). Now, let \( H_1(Z) < H_i(Z) \). Then \( H_i(Z) \leq \alpha^\frac{(c-\frac{d}{2})}{2} \cdot H_1(Z) \), \( 1 \leq i \leq n \); since \( H_i(Z) = \omega(n) \cdot \alpha^z_i \) for some integer \( z_i \).

**Lemma 5.** For \( i \geq cn \), \( H_{i+1}(Z) \leq \alpha^\frac{(c-\frac{d}{2})}{2} \cdot H_i(Z) \).

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

**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 \( QON \). 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.

\[
\begin{align*}
H_{c \cdot n} & = \cdots H_{(c-\frac{d}{2})n+1} < \cdots < H_{c \cdot n-1} \quad (1) \\
H_{(c-\frac{d}{2})n+1} & = \cdots H_{(c-\frac{d}{2})n} < \cdots < H_1 \quad (2)
\end{align*}
\]

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

\[
H_{n-1} < \cdots < H_{c \cdot n} < \cdots < H_{(c-\frac{d}{2})n+1} \quad (3)
\]

\[
C(Z) = \sum_{i=1}^{(c-\frac{d}{2})n} H_i + \sum_{i=(c-\frac{d}{2})n+1}^{n-1} H_i \leq H_{(c-\frac{d}{2})n} \left[ \frac{c}{2} + (\frac{d}{2})^2 + \ldots \right] + H_{(c-\frac{d}{2})n} \left[ 1 + (\frac{d}{2}) + (\frac{d}{2})^2 + \ldots \right] \leq 2^{\frac{c}{2}} \cdot H_{(c-\frac{d}{2})n}, \text{ since } H_{(c-\frac{d}{2})n} = H_{(c-\frac{d}{2})n+1} \leq \alpha \cdot H_{(c-\frac{d}{2})n} \text{ since } \alpha \geq 4 \leq \omega(n) \cdot \alpha^{\frac{(c-\frac{d}{2})n}{2}+1} = K_{c, d}(n).
\]

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

**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 \( v_k \) \( V_1 \), there exists an edge \( \{ v_i, v_k \} \notin E \) for some \( v_i \in V_1 \); otherwise \( V_1 \cup \{ v_k \} \) will form a clique. Thus, \( |E| \leq \frac{n(n-1)}{2} - |V-V_1| = \frac{n(n-1)}{2} - n + w(G) \).

**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 \( QON \) obtained by applying the reduction \( f \) to \( G \). Then \( OPT(Q) \leq K_{c, d}(n) \cdot \alpha^\frac{(c-\frac{d}{2})n}{n} \).

**Proof:** Consider an optimal sequence \( Z = R_1, R_2, \ldots, R_{n+1} \) 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

\[
D_{(c-\frac{d}{2})n}(Z) \leq (c-\frac{d}{2})n \left[ (c-\frac{d}{2})n-1 \right] - (\frac{d}{2})n \leq \frac{n(n-1)}{2} - (\frac{d}{2})n \leq K_{c, d}(n) \cdot \alpha^\frac{(c-\frac{d}{2})n}{n}.
\]

**Theorem 9.** There is a positive constant \( \epsilon \) such that there is a gap preserving reduction from \( MAX-3SAT(13) \) to \( QON \) such that

(a) Satisfiable formulas map to instances of \( QON \) where optimal cost \( \leq K_{c, d}(n) \)

(b) Unsatisfiable formulas map to instances of \( QON \) where optimal cost \( \geq K_{c, d}(n) \cdot \alpha^\frac{(c-\frac{d}{2})n}{n} \), where \( n \) is the size of the query graph produced by the reduction.

Note that \( \alpha \) can be any number \( \geq 4 \) whose size in (number of bits) is some polylogarithmic function of the size of the input instance. By choosing \( \alpha \) to be 4, we obtain \( K_{c, d}(n) \) to be approximately \( 4^\frac{n}{\text{poly}(n)} \). Thus, the gap produced is \( 4^\frac{n}{\text{poly}(n)} \) which is \( 2^\frac{n}{\text{poly}(n)} \cdot K \), that is, solving the \( QON \) problem approximately with a better competitive ratio than \( 2^\frac{n}{\text{poly}(n)} \cdot K \) using a polynomial time algorithm would mean that \( P = \text{NP} \).

By increasing \( \alpha \), the gap expression remains of the form \( 2^\frac{n}{\text{poly}(n)} \cdot K \), with \( h \) tending to 1, or equivalently, the gap expression can be made to equal \( 2^\frac{n}{\text{poly}(n)} \cdot \delta \) 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 $\frac{2}{9}$CLIQUE problem to the $QO_H$ problem.

We now give the reduction from $\frac{2}{9}$CLIQUE to $QO_H$. Let $G = (V, E)$ be an instance, $I$ of $\frac{2}{9}$CLIQUE, where $V = \{v_0, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$. 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 edges 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 $s = B^2 \cdot \frac{m}{2} - i$ 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 $M = (m - 1) \cdot B$ pages.

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

Some Comments and Observations

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

Select $R_{0,0,a} 
\text{from } R_{0,0}, R_{1,0}, R_{2,0}, \ldots, R_{m,0} 
\text{where} 
\text{predicates between } R_{0,0} \text{ and each of } R_{1,0} \text{ to } R_{m,0} 
\text{and predicates among } R_{1,0} \text{ to } R_{m,0}$

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 $\geq \sqrt{n_0} = \sqrt{m^2 \cdot B^2} = m \cdot B \geq \frac{(m - 1)}{2} \cdot B = M$. 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 $\frac{2}{9}$CLIQUE problem.

Lemma 10. If the instance $I$ of $\frac{2}{9}$CLIQUE has a clique $C$ of size $\geq \frac{m}{2}$ 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 $\frac{m}{2}$ relations in the first pipeline, $\frac{m}{2} + 1$ relations in the second pipeline and $\frac{m}{2} - 1$ relations in the last pipeline - $P(1, \frac{m}{2}), P(\frac{m}{2} + 1, 2m + 1)$ and $P(2m + 2, m)$ is the sequence of pipelines. The parameter $m'$ is the minimum memory that should be allocated to the join.

The last join in the pipeline $P(\frac{m}{2} + 1, 2m + 1)$ is with a relation $R_0$ which is not present in $C$. Consider the number of edges between the vertex $v_0$ and the vertices corresponding to the relations involved in the joins preceding the join with $R_0$. The number of edges among vertices $v_i 
\in C \leq \frac{(2m)(2^m - 1)}{2}$.

The total number of edges is $> \frac{8m^2 - 18m}{13}$.

So, there are $> \frac{2m^2 - 12m}{13}$ edges remaining. Let the vertices corresponding to the last $\frac{m}{2}$ relations i.e. the vertex $v_0$ and the $\frac{m}{2} - 1$ 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 $Q = \frac{2m^2 - 9m}{13}$. Now the number of edges remaining is $> \frac{3m^2 - 9m}{13}$, which are the edges between vertices in $C$ and vertices in $Q$. Since the number of vertices in $Q$ is $\frac{m}{2}$, on an average, each vertex in $Q$ should be connected to $> \frac{3m^2 - 9m}{13} \cdot \frac{m}{2}$ vertices in $C$. Or in other words there should exist at least one vertex which is connected to $> \frac{m^2}{2} + 1 = \frac{m}{2} - 1$ vertices in $C$. The relation $R_0$ is chosen to correspond to such a vertex $v_0$. Now it can be verified that the number of pages materialized at the end of pipeline is

$\leq \frac{m^2}{2} \cdot B^2 \cdot B^{(2m)} \cdot s \frac{(2^m - 1)}{2} \cdot \frac{(2m - 1)}{2}$.

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

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

3. The cost of executing the pipeline $P(2m + 2, m)$ 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 $\leq m^2 \cdot B^2 \cdot B(\frac{2m}{3}) \cdot s^2 m (\frac{2m}{3} + 1) / 2$. The total cost of executing the three pipelines is $\leq K$. Hence the proof.

**Lemma 11.** If the instance $I$ of the $\frac{3}{2}$-CLIQUE problem has no clique of size $\geq \frac{2m}{3}$, then for the instance $f(I)$ of QO$_m$ there no sequence of pipelines $S'$ and memory allocation vector $m'$ such that $C_{seq}(S', m', M) \leq K \cdot 2 \cdot (\frac{1}{2})^2 (\log K - 4)\frac{1}{2} - 4$.

**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

$$\geq 2 \cdot m^2 \cdot B^2 \cdot B(\frac{2m}{3} + 1) \cdot s^2 (\frac{2m}{3} + 1) / 2.$$ 

$$= 2 \cdot m^2 \cdot B^2 \cdot B(\frac{2m}{3} + 1) \cdot s^2 (\frac{2m}{3} + 1) / 2.$$

$$= 2 \cdot m^2 \cdot B^2 \cdot B(\frac{2m}{3} + 1) \cdot s^2 (\frac{2m}{3} + 1) / 2.$$

$$= 2 \cdot m^2 \cdot B^2 \cdot B(\frac{2m}{3} + 1) \cdot s^2 (\frac{2m}{3} + 1) / 2.$$

$$= K \cdot 2 \cdot (\frac{1}{2})^2 (\log K - 4)\frac{1}{2} - 4.$$

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 $\geq K \cdot 2 \cdot (\frac{1}{2})^2 (\log K - 4)\frac{1}{2} - 4$. 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 $\geq \frac{2m}{3}$ in the instance $I$. It has been shown that the number of edges in the subgraph comprising of the any subset of vertices $\frac{\sqrt{2}}{3}$ in number is $< \frac{2m}{3} \cdot m - \frac{3}{7}$. Since the number of edges is $\frac{3}{7} m$ less than the number of edges in a clique of size $\frac{2m}{3}$. The materialization cost is

$$\geq \frac{m}{3} \cdot B \cdot B(\frac{2m}{3}) \cdot s^2 (\frac{2m}{3} + 1) / 2.$$

$$= K \cdot 2 \cdot (\frac{1}{2})^2 (\log K - 4)\frac{1}{2} - 4.$$

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

$$\geq \beta \cdot m^2 \cdot B^2 \cdot B(\frac{2m}{3}) \cdot s^2 (\frac{2m}{3} + 1) / 2.$$

$$= K \cdot 2 \cdot (\frac{1}{2})^2 (\log K - 4)^{\frac{1}{2}} - 4.$$

Since $\leq B$ pages are 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 $g(\min, B) \geq B^{-\frac{1}{2}}$. Hence the proof.

**Theorem 12.** Approximating QO$_m$ to within a factor of $2^n (\log K - 4)^{\frac{1}{2}} - 4$ is NP-Hard. Here $K$ is the optimal cost of executing the query and $\alpha$ is a constant.

**Proof:** Follows from Lemmas 10 and 11 that

$$\text{Instance } I \in \frac{3}{2}\text{-CLIQUE } \Rightarrow \text{OPT}(f(I)) \leq K \cdot \frac{1}{2} (\log K - 4)\frac{1}{2} - 4.$$

The constant $\alpha$ mentioned in the statement of Theorem 12 is $\frac{1}{2}$.

### 5.3.1 Hardness of Approximability under further restrictions on QO$_m$ 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 $\epsilon(n)$ is a given function of the number of vertices $n$, where $3m \leq \epsilon(n) \leq n^2/3$, the approximation problem for the QO$_m$ problem remains equally difficult.

**Theorem 13.** Given a function $\epsilon(m)$, $3m \leq \epsilon(m) \leq \frac{2m}{3}$ such that we can find a graph with $\epsilon(m)$ edges in polynomial time. Then there is a gap preserving reduction from instances of MAX-3SAT(13) to instances of QO$_m$ where the query graph has $m$ vertices and $\epsilon(m)$ edges such that

1. Satisfiable formulae map to instances of QO$_m$ where optimal cost $\leq K_{c,d}(n)$, and
2. Unsatisfiable formulae map to instances of QO$_m$ where optimal cost $\geq K_{c,d}(n) \cdot \alpha^{\epsilon(n)}$, 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 $\epsilon(n^2) - |E'| - 1$ edges. Clearly, for any value of $\epsilon(m)$, we can construct a graph with $\epsilon(m)$ 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 in $e(n^2)$.

2. Consider any constant $\beta \geq 4$ such that $\beta$ is bounded by a polynomial in $n$. Let $\alpha = \beta^{\alpha^2}$. 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 $\beta^\alpha$.

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

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

5.4 $\text{QO}_{\text{N}}$ and Sparse Graphs

It can be observed that the constructed instance of $\text{QO}_{\text{H}}$ 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 $V' = V \cup \{v_0\} \cup \{v'_1, v'_2, \ldots, v'_{m^2-2}\}$ and the edge set $E' = E \cup \{e_{p+1}, e_{p+2}, \ldots, e_{p+\beta}\} \cup \{e'_1, e'_2, \ldots, e'_{m^2-2}\}$. The vertex $v_0$ corresponds to the relation $R_0$. Edge $e_{p+1}$ means that there is a join predicate between relations $R_0$ and relation $R_i$. The vertex $v'_j$ corresponds to the relation $R'_j$. Edge $e'_j$ means that there is a join predicate between relation $R_j$ 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 $R_0$, $n_0 = m^2 \cdot B^2$. 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 $(\frac{2^{m^2}}{m^2-1})^j$, 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 $s = B^{(m^2-1)/(m-1)}$, 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 $R_j'$, for $1 \leq j \leq m^2 - m$ is 1.

6. The available main memory $M = (\frac{m^2}{2} - 1) \cdot B$ pages.

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

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 $R'_j, 1 \leq j \leq m^2 - m$, 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 $\text{QO}_{\text{N}}$ and $\text{QO}_{\text{H}}$ 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 $\text{SQO} - \text{CP}$ 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 $\text{SQO} - \text{CP}(\text{Star Query Optimization without Cartesian Products})$ 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 $2^{\log^{1-\delta}(n)}$ 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 $3n \leq f(n) \leq n^{\gamma / 3}$ 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


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 $R_0$ 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 $S_n$ 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

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

Thus, $A_i = b_k \cdot k_i$. The term $k_i$ 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

\[
\text{select } R_0.\{\text{attribute list}\}
\text{from } R_0, R_1, \ldots, R_m
\text{where predicates}
\]

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.

\[
b(R_0) = b_r
\]

\[
b(X) = n(X) \text{ if } X \text{ contains at least 2 relations.}
\]

Let \(Z = XS, Y\) be a feasible sequence. The cost of the sort-merge join operator \(S_i\) is estimated as

\[
b(X) \cdot (k_i - 1) + A_i
\]

Let \(Z = XN, Y\) be a feasible sequence. The cost of the nested-loops join operator \(N_i\) is given by

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

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.

\[
D(\phi, R_rN_i, Y) = \begin{cases} 
  b_r + w_i \cdot n_0 + D(R_0N_i, Y) & r = 0 \\
  b_r + w_{0,r} \cdot n_r + D(R_rN_0, Y) & r \neq 0
\end{cases}
\]

\[
D(\phi, R_rS_i, Y) = C_{sm}(R_r, R_1) + D(R_rS_i, Y)
\]

\[
D(W, S_i, Y) = b(W) \cdot (k_r - 1) + A_r + D(WS_i, Y)
\]

\[
D(W, N_i, Y) = n(W) \cdot w_i + D(WN_i, Y)
\]

\[
D(W, \phi) = 0
\]

### 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 \(\sum_{i=1}^n b_i = \text{an even number}\).

**QUESTION.** Does there exist a subset \(V\) of \(U\) such that \(\sum_{b_i \in V} b_i = L\)?

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 \(x \geq 0\), \(f_q(x) : R \rightarrow Q\) is defined as \(f_q(x) = \lceil 2^x / 2^q \rceil\). Given \(n\) positive integers \(b_1, b_2, \ldots, b_n\), the function \(g_n(x)\) is defined as \(g_n(x) = 2^y f_q(x^z/2^K)\), 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 up to 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 \(K = \sum_{i=1}^n b_i\),

- \(p = \lceil log_2 K \rceil + 1, q = 2p + 7 + n\)
- \(S = g_n(K/2)\)
\[ m = 2n \]

- \[ p_i = g_0(b_i) \text{ and } c_i = 3SK + b_iS \]
- \[ p_i = 2^{(i-n)/2} \text{ and } c_i = (i - n)3SK \]

\[ p_{2n} = 2K \text{ and } c_{2n} = (2K) \left( \sum_{i=1}^{2n-1} p_i \right) + 1 \]

\[ W = \{(p_1, c_1), (p_2, c_2), \ldots, (p_m, c_m)\} \]

\[ L = 3KS/2 + m(n-1)3KS/2 + 2K + SK \]

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 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 \( J = \left\{ 4 \cdot k_s \cdot \prod_{i=1}^{m} p_i \right\}^2 \) 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).
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 \( w_{0,i} = n_0 \) 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 \( \text{mem} = n_0/2 \) 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 \( \text{mem} < b_0 < b_{m+1} < (\text{mem})^2 \), 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].