On the Complexity of Approximate Query Optimization

[Extended Abstract]

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

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

S. Ganguly
Bell Labs
Lucent Technologies
sganguly@research.bell-labs.com

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

ABSTRACT

In this work, we study the complexity of the problem of approximate query optimization. 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. The complexity gap remains if the number of edges in the query graph is constrained to be a given function $e(n)$ of the number of vertices n of the query graph, where $n(n-1)/2 - \Theta(n^{\tau}) \geq e(n) \geq n + \Theta(n^{\tau})$ and τ is any constant between 0 and 1. These results show that, unless P=NP, the query optimization problem cannot be approximately solved by an algorithm that runs in polynomial time and has a competitive ratio that is within some polylogarithmic factor of the optimal cost.

1. INTRODUCTION

The problem of finding an optimal join order sequence for Select Project Join (SPJ) Queries is a classical problem in Query Optimization and has merited considerable research attention.

It is well-known that this problem is NP-complete in general [1], and continues to be so in many variant formulations [2, 3]. Despite the set of results on NP-completeness, there may be a justifiable optimism about the existence 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.

Principles of Database Systems 2002, Madison, Wisconsin USA
Copyright 2002 ACM 0-89791-88-6/97/05 ...$5.00.

We consider several variants of the query optimization problem. In one variant of the problem, we restrict all join operations to computed using the nested-loops join method only, by following a cost model virtually identical to that of [1]. We then consider a variant where join operations are restricted to be computed using the hash-joins method only. 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 instances where the number of edges in the query graph matches a given function $e(n)$ of the number of vertices n of the query graph, where $n(n-1)/2 - \Theta(n^{\tau}) \geq e(n) \geq n + \Theta(n^{\tau})$, for any constant $0 < \tau < 1$. Under this restriction, we show that the complexity of approximation remains equally difficult.

The rest of the paper is organized as follows. Section 2 presents the problem definitions for two variants of the query optimization problem, namely, QO_N and QO_H, that we study in this paper. Section 3 presents the proof techniques used in proving the hardness of the problems QO_N and QO_H respectively. Sections 4 and 5 presents the complexity results for the problems QO_N and QO_H respectively. In Section 6, we extend the complexity results to the case when instances of the QO_N and QO_H problems are sparse query graphs. We offer concluding remarks in Section 7. Appendix A proves the NP-completeness of another variant of the query optimization problem.

2. PROBLEM DEFINITIONS

In this paper we study two variants of the query optimization problem, which restrict relational join operations to be computed using the nested-loops join method and the hash-joins method respectively. These problems are called QO_N and QO_H respectively, which we define next.

2.1 The QO_N problem

In this section, we define the QO_N problem by specifying its instances, the cost function for an instance and the optimization problem. Our definition of QO_N is similar to that given in [1].

2.1.1 Instance of QO_N

An input instance of the QO_N problem is specified as a five-tuple $(n, Q = (V, E), S, T, W)$ as follows.

The term Q is an undirected query graph $Q = (V, E)$, where V is the vertex set $\{v_1, v_2, \ldots, v_n\}$ and E is a subset of the set of all unordered pairs over V. Each vertex in the vertex set V corresponds to a relation R_i for $1 \leq i \leq n$. An undirected edge $\{v_i, v_j\}$ between two vertices v_i and v_j means that the relations R_i and R_j have a join predicate between them. The absence of an edge between two vertices indicates the absence of a join predicate between the corresponding relations.

The selectivity matrix S is an $n \times n$ symmetric matrix, specifying the selectivity $s_{ij} = s_{ji}$ of the predicate corresponding to relations R_i and R_j, for $1 \leq i, j \leq n$ and $i \neq j$. If relations R_i and R_j are not related by a predicate, then $s_{ij} = s_{ji} = 1$. The relation sizes vector $T = (t_1, t_2, \ldots, t_n)$ specifies the number of the tuples of the relations $R_1, R_2, \ldots, R_n$ respectively. In order to minimize mathematical notation, we assume that the size of each tuple of a base relation and all intermediate relations is one page.

The access path cost matrix is an $n \times n$ matrix, that associates with each unordered pair $\{v_i, v_j\}$ of vertices in V, two integer values, denoted by w_{ij} and w_{ji}. If the vertices do not have an edge between them in the query graph Q, then w_{ij} is defined to be t_i and w_{ji} is defined to be t_j respectively. This cost model corresponds to the fact that for every tuple of relation R_i, all tuples of the relation R_j qualify, since there is no join predicate between R_i and R_j. Otherwise, the vertices v_i and v_j are related by an edge in E, that is, the relations R_i and R_j are related by some predicate P. In this case, the value w_{ij} denotes the least cost of solving the predicate P for some given tuple of R_j (containing join attributes from R_j that are relevant to the predicate P) among all the possible choices of access paths for R_i. We constrain w_{ij} to be at least $t_i s_{ij}$ since this is the expected number of tuples (and therefore pages) output for each tuple of R_j. Similarly, w_{ij} is constrained to be at most t_i, since, in the worst case, all tuples of R_i are accessed once. The value w_{ji} is defined and constrained likewise with the roles of R_j and R_i reversed.

2.1.2 Instance Cost and Decision Problem for QO_N

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

Consider an instance of QO_N as specified above. Let $Z = (v_{z_1}, v_{z_2}, \ldots, v_{z_n})$ be any permutation of vertices in V. The sequence Z is called a join sequence consisting of n join operations denoted by $J_1(Z), J_2(Z), \ldots, J_{n-1}(Z)$. The first join operation $J_1(Z)$ is between relations R_{z_1} and R_{z_2}. The k^{th} join operation is between the result of join operation $J_{k-1}(Z)$ and the relation $R_{z_{k+1}}$.

Let Z be a join sequence and let 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,

$$\begin{array}{lll} N(X) = & 1 & \text{if } X = \phi \\ N(Xv_j) = & N(X) \cdot t_j \prod_{v_i \in X} s_{ij} & \text{otherwise.} \end{array}$$

The intermediate size produced by the join operation $J_i(Z)$ is denoted by $N_i(Z)$ and is defined as $N(Xv_j)$. 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 intermediate size produced by the join operation $J_i(Z)$ is denoted by $N_i(Z)$ and defined as $N(Xv_j)$. The cost of the join operation $J_i(Z)$ according to the nested loops join method, is denoted by $H_i(Z)$, and is defined as

$$H_i(Z) \;=\; N(X) \cdot min_{v_k \in X} w_{jk}$$

The cost of a join sequence Z is defined as the sum of the cost of the $n-1$ join operations, that is,

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

All $n!$ permutations of the elements of V are candidate join sequences. The problem QO_N may now be posed as follows.

QO_N: Given an instance $(n, Q = (V, E), S, T, W)$, and a number K, does there exist a join sequence Z such that $C(Z) \leq K$?

2.2 The QO_H problem

We now specify instances, costs and the decision problem for the query optimization problem QO_H in which join operations are computed using the hash-joins method.

An instance of the QO_H problem is specified by a five-tuple $(n, Q = (V, E), S, T, M)$, where the number of relations n, query graph Q, selectivity matrix S and sizes of relations vector T have the same meanings as they were defined to have for the problem QO_N. The term M denotes the total memory that is available for each set of relations scheduled to be joined as a hash-join pipeline. We first describe the execution model, followed by the cost model for estimating the cost of a given pipelined sequence of hash-joins.

2.2.1 Execution Model for Pipelined Hash Joins

The query corresponding to the query graph Q is executed by first ordering the vertices of the vertex set V into a join sequence Z. Let Z be the sequence $v_{z_1}, v_{z_2}, \ldots, v_{z_n}$.

The join sequence Z is executed by decomposing it into a collection of one or more *pipelines*, where each pipeline is specified by two indices i and k, $1 \leq i \leq k \leq n-1$, and is denoted by $P(Z, i, k)$. The pipeline $P(Z, i, k)$ denotes that the join operations $J_i(Z)$ through $J_k(Z)$ (inclusive) are executed concurrently in a pipeline. A pipeline decomposition of a join sequence is a partition of the sequence into contiguous fragments, where (a) each fragment is executed as a pipeline, and (b) the output of each fragment is materialized to disk and used as the outermost relation for the execution of the following fragment (if any). The first relation R_{z_1} from the join sequence Z is used as the outermost relation for the first pipeline. During the execution of a pipelined fragment $P(Z, i, k)$, the join operations within the pipeline are executed concurrently by partitioning the available main memory M among the join operations in that pipeline. The hash tables corresponding to the relations $R_{z_{i+1}}$ through R_{z_k} are built before the phase of pipelined sequence of hash table probes commences. However, if the allocated memory to a join operation in the pipeline is not sufficient to stage its inner relation into memory, then the relation is first partitioned as in a hybrid hash-join algorithm. Such partitioning may have a stalling effect the later join operations in the pipeline.

2.2.2 Instance Cost and Decision Problem for QO_H

In this section, we present formulae for giving costs to a single hash-join operation and use it to define cost formulae for a pipeline decomposition of a join sequence.

The I/O cost of a hash-join operation between outer relation R, with b_R pages, and inner relation S with b_S pages using m pages of memory is denoted by $h(m, b_R, b_S)$ and is abstracted as follows. We assume that R is streaming into memory whereas the inner relation S is resident on disk.

$$h(m, b_R, b_S) = (b_R + b_S)\Theta(g(m, b_S)) + b_S \text{ if } m \geq hj_{min}(b_S)$$

The function $hj_{min}(b_S)$ is assumed to be a given lower bound on the memory that must be made available in order for the hash-join between relations R and S to be feasible. We assume that $hj_{min}(b_S) = \Theta(b_S^{\psi})$, for some constant $0 < \psi < 1$. The function $g(m, b_S)$ satisfies the following constraints.

1. $g(m, b_S)$ is a linear decreasing function of m in the range $hj_{min}(b_S) \leq m < b_S$.
2. $g(m, b_S) = 0$, for $m \geq b_S$,
3. g is a continuous function.
4. $g(hj_{min}(b_S), b_S) = \Theta(1)$.

It follows that $h(hj_{min}(b_S), b_R, b_S) = \Theta(b_R + b_S)$.

The execution of a pipeline $P(Z, i, k)$ is specified by a memory allocation vector that specifies the memory allocated to each of the the $k - i + 1$ hash-join operations in the pipeline. The cost of the execution of a pipeline $P(Z, i, k)$ is the sum of the following components:

1. The cost of reading from disk the intermediate relation which is materialized at the end of the previous pipeline, that is, $N_{i-1}(Z)$.
2. The sum of the cost of performing each of the hash-join operations $J_i(Z)$ through $J_k(Z)$ according to the given memory allocation vector.
3. The cost of writing to disk the output of the pipeline, that is, $N_k(Z)$.

The cost of executing a pipeline decomposition of a join sequence Z according to a given memory allocation vector is defined as the sum of the costs of executing each of the pipelined fragments in the pipeline decomposition. The problem QO_H may now be posed as follows.

QO_H: Given an instance $(n, Q = (V, E), S, T, M)$, and a number L, does there exist a join sequence Z, a decomposition of Z into a sequence of pipelines and a memory allocation vector such that the cost of executing the pipeline decomposition according to the given memory allocation vector is at most L?

3. PRELIMINARIES

In this section, we review the relevant results from the literature and define notation that is used subsequently in the paper. We first review the problem definition for 3SAT(13).

3SAT(13): An instance of this problem consists of (a) a boolean formula in 3CNF such that each variable occurs in at most 13 clauses, and (b) a fraction x where $0 < x \leq 1$. The question is, does there exist an assignment of truth values to the variables such that a fraction x of the clauses are satisfied by that assignment?

Our proofs use the following two variants of the standard clique problem, which we denote by CLIQUE and $\frac{2}{3}$CLIQUE, and define next.

CLIQUE: An instance of this problem is (a) an undirected graph $G = (V, E)$ in which the degree of each vertex is at least $|V| - 14$, and (b) a number k. The question is, does there exist a clique of size k in G?

$\frac{2}{3}$CLIQUE: Instances of this problem are undirected graphs $G = (V, E)$ where the degree of each vertex is at least $|V| - 14$. The question is, does there exist a clique of size $(2/3)|V|$?

We first state two fundamental theorems from [4] that form the starting point of our reductions.

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

PROOF. See proof of Theorem 6.2 of [4]. □

THEOREM 2. *There exist positive constants γ and μ for which there is a polynomial time reduction from 3SAT to VERTEX COVER with the following properties:*

1. *Satisfiable formulae map to graphs G with n vertices whose vertex cover has size at most γn.*
2. *Unsatisfiable formula map to graphs G with n vertices whose vertex cover has size at least $\gamma(1 + \nu)n$.*
3. *$\gamma > \frac{1}{2}$, $\gamma(1 + \nu) > \frac{1}{2}$.*

PROOF. See Theorem 6.22 in [4]. □

LEMMA 3. *There exist constants c and d for which there is a polynomial time reduction that maps instances of 3SAT to graphs from CLIQUE with the following properties:*

1. *Satisfiable formulae are mapped to graphs G with n vertices such that G has a clique of size cn.*
2. *Unsatisfiable formula are mapped to graphs G with n vertices such that the largest clique in G is of size at most $(c - d)n$.*
3. *$c > \frac{2}{3}, c - d > \frac{2}{3}$.*

PROOF SKETCH. The proof is a direct adaptation of Theorem 2. Consider the reduction given in [5] from instances of 3SAT to instances of VERTEX COVER. From Theorem 1, it follows that it is sufficient to restrict ourselves to instances of 3SAT(13) in which either the instance is a satisfiable formula or at most a fraction $1 - \theta$ of the clauses can be satisfied.

Consider the graph $G = (V, E)$ obtained by applying the reduction to a formula F that is an instance of 3SAT(13), and which consists of v variables and m clauses. It now follows that if F is satisfiable then G has a vertex cover of size $v + 2m$, and if at most a fraction $1 - \theta$ of the clauses of F are satisfied, then, G has a vertex cover of size at least $v + 2m + \theta m$. The claims about c and $c - d$ may be verified by forming the complement graph G^c of G and then extending G^c with a complete graph over $4v + 3m$ vertices and connecting each of the new vertices to all the vertices in V. □

LEMMA 4. *There exists a positive constant $\epsilon < 1$ and a polynomial time reduction that maps instances of 3SAT to graphs from $\frac{2}{3}$CLIQUE with the following properties:*

1. *Satisfiable formulae are mapped to graphs that have a clique of size at least $2n/3$, where n is the number of vertices in the graph.*
2. *Unsatisfiable formulae are mapped to graphs whose largest clique has size at most $(2-\epsilon)n/3$.*

PROOF SKETCH. The reduction is similar to the argument in Lemma 3. Let $G = (V, E)$ be the graph produced by the reduction and the standard reduction of instances of 3SAT to instances of VERTEX COVER. Let G^c be the complement graph of G. We obtain $G' = (V', E')$ by adding $n_1 = (3\gamma - 1) \cdot |V|$, where γ is the constant mentioned in Theorem 2. The new vertices are all connected to each other and to every vertex in V. Thus $n = |V'| = |V| + n_1 = 3\gamma|V|$. The claims of the lemma can now be directly verified. □

4. COMPLEXITY OF QO_N

In this section, we present a polynomial time reduction function f_N from instances of CLIQUE to QO_N.

Let $G = (V, E)$ be an instance of CLIQUE where $n = |V|$. We construct an instance $(n, Q = G, S, T, W)$ with the following properties. The query graph Q is set to be identical to the input graph G. The selectivity s_{jk} corresponding to the edge $\{v_j, v_k\} \in E$ is set to 1α, where $\alpha = \Omega(1)$. The sizes of the relations corresponding to the vertices in V are set equal to $t(n) = \alpha^{(c-\frac{d}{2})n}$, where c and d are constants obtained from Lemma 3. For each edge $\{j, k\} \in E$, w_{jk} is set to $\omega(n) = t(n)/\alpha(n)$. For vertex pairs $\{v_j, v_k\} \notin E$ we set $w_{jk} = t(n)$.

Since the degree of each vertex in G is at least $n - 14$, we assume that there exists a constant n_0 ($n_0 > 28$) such that any instance G with at least $n \geq n_0$ vertices is a single connected component. Further we assume that $n_0 > 30/d$. For brevity, we denote $\alpha(n), t(n)$ and $\omega(n)$ by α, t and ω respectively. For a graph G, we let $w(G)$ denote the largest size of a clique in G.

We now establish some simple properties of the reduction and define notation that is used later. Let Z be a join sequence for the constructed instance of QO_N. The term $D_i(Z)$ denotes the number of edges in the subgraph of the query graph G formed by the vertices that occupy the first i positions of Z. We let the term $K_{c,d}(\alpha, n)$ to be $\omega \cdot \alpha^{\frac{[(c-\frac{d}{2})n][(c-\frac{d}{2})n+1]}{2}+1}$.

Given a join sequence Z, an edge $\{v_j, v_k\} \in E$ is called a back-edge corresponding to the vertex v_j if v_k appears earlier than v_j in Z. For $1 \leq i \leq n$, 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.

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, it follows that

$$H_i(Z) = \begin{cases} \omega N(X) & \text{if } B_{i+1}(Z) > 0 \\ T_j N(X) & \text{if } B_{i+1}(Z) = 0 \end{cases}$$

Consider the join sequence Z and assume that none of the $n-1$ join operations in Z are cartesian products (i.e., $B_i(Z) > 0$, for $1 \leq i \leq n-1$). Then, for $1 \leq i \leq n-1$, $H_i(Z) = \omega\alpha^{(c-\frac{d}{2})\cdot n \cdot i - D_i(Z)}$.

LEMMA 5. *Let Z be a join sequence for an instance I of QO_N consisting of $n \geq n_0$ vertices in the query graph. If Z has no cartesian products, then, for $i \geq cn, H_{i+1}(Z) \leq (\frac{1}{\alpha}) \cdot H_i(Z)$.*

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

$$\begin{aligned} B_{i+1}(Z) &\geq i - 14 \\ &\geq cn - 14 \\ &= (c - \tfrac{d}{2})n + 1 + (\tfrac{d}{2})n - 15 \\ &> (c - \tfrac{d}{2})n + 1 \quad \text{since } n \geq n_0. \end{aligned}$$

It follows that $H_{i+1}(Z) < H_i(Z) \cdot (\frac{1}{\alpha})$. □

LEMMA 6. *Let G be an instance of CLIQUE consisting of $n \geq n_0$ vertices such that $w(G) \geq cn$. Suppose that the reduction function f_N maps the graph G to the instance I of QO_N. Then, there is a join sequence Z for I such that $C(Z) \leq K_{c,d}(\alpha, n)$.*

PROOF. We consider a join sequence Z with no cartesian products, such that the first cn nodes of Z are the nodes of a clique of size $w(G) \geq cn$. As noted above, G is connected and hence such a sequence Z can be constructed. Let $1 \leq i \leq cn - 1$. Then, $B_i(Z) = i - 1$ and $D_i(Z) = i(i-1)/2$. It follows that

$$\begin{aligned} H_i(Z) &= \omega\alpha^{(c-\frac{d}{2})\cdot n \cdot i - D_i(Z)} \\ &= \omega\alpha^{(c-\frac{d}{2})\cdot n \cdot i - \frac{i(i-1)}{2}} \end{aligned}$$

The function of i in the exponent attains its discrete maximum when $i = (c - d/2)n$ or $i = (c - d/2)n + 1$. Thus, we have the following relations.

$$\begin{aligned} &H_1(Z) < H_2(Z) < \ldots < H_{(c-\frac{d}{2})n}(Z) \text{ and} \\ &H_{(c-\frac{d}{2})n}(Z) = H_{(c-\frac{d}{2})n+1}(Z) \text{ and} \\ &H_{(c-\frac{d}{2})n+1}(Z) > H_{(c-\frac{d}{2})n+1}(Z) > \ldots > H_{cn}(Z) \end{aligned}$$

From Lemma 5, we can further extend the above set of inequalities to obtain the following

$$H_{cn}(Z) > H_{cn+1}(Z) \ldots > H_{n-1}(Z)$$

Further, since Z has no cartesian products, $H_i(Z) < H_j(Z)$ implies that $H_i(Z) \leq (1/\alpha)H_j(Z)$, since, each of $H_i(Z)$ and $H_j(Z)$ are ω multiplied by some (different) power of $1/\alpha$. It therefore follows that

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

□

The following lemma states a simple property about graphs by giving an upper bound on the number of edges that a graph G with a maximum clique size of $w(G)$ can have.

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 W be a set of $w(G)$ vertices of G which form a clique. For each vertex $v \in V - W$, there exists a vertex $w \in W$, such that the pair $\{v, w\}$ is not an edge in E. Otherwise $W \cup \{v\}$ will be a clique of size $w(G)+1$. It follows that $|E| \leq \frac{n\cdot(n-1)}{2} - |V - W| = \frac{n\cdot(n-1)}{2} - n + w(G)$. $\square$

Lemma 8. *Let $G = (V, E)$ be a graph with $n > 30/d$ vertices such that $w(G) \leq (c - d)n$. Let I be the instance of QO_N obtained by applying the reduction f_N to G. Then, for every join sequence Z, $C(Z) \geq K_{c,d}(\alpha, n) \cdot \alpha^{(\frac{d}{2})n-1}$.*

Proof. Consider any join sequence Z. Consider the subgraph $G_1 = (V_1, E_1)$ of G induced by the nodes of G which occur in the first $(c - \frac{d}{2})n$ positions of Z. Let w_1 be the size of the largest clique in G_1. By our hypothesis about $w(G)$, it follows that $w_1 \leq (c-d)n$. From Lemma 7, we obtain that

$$\begin{aligned} D_{(c-\frac{d}{2})n}(Z) &= |E_1| \\ &\leq [(c-\tfrac{d}{2})n]\,[(c-\tfrac{d}{2})n-1]/2] \\ &\quad -(c-\tfrac{d}{2})n + (c-d)n \\ &= [(c-\tfrac{d}{2})n]\,[(c-\tfrac{d}{2})n-1]/2] + \tfrac{d}{2}n \end{aligned}$$

It therefore follows that

$$\begin{aligned} C(Z) &\geq H_{(c-\frac{d}{2})n} \\ &\geq \omega\alpha^{[(c-\frac{d}{2})n][(c-\frac{d}{2})n]-D_{(c-\frac{d}{2})n}(Z)} \end{aligned}$$

Note that the above inequality holds irrespective of whether cartesian products occur in the first $(c - \frac{d}{2})n$ join operations in Z or not. If there are cartesian products, then the cost is higher than the RHS above (since $\omega < t_j/\alpha$, for $1 \leq j \leq n$), otherwise, the RHS is a correct estimate. Thus,

$$\begin{aligned} C(Z) &\geq \omega\alpha^{\frac{[(c-\frac{d}{2})n][(c-\frac{d}{2})n+1]}{2} + (\frac{d}{2})n} \\ &\geq K_{c,d}(\alpha, n) \cdot \alpha^{(\frac{d}{2})n-1}. \end{aligned}$$

$\square$

The following result formally states the complexity of the approximating QO_H by composing the reduction f_N with the reduction function in Lemma 3.

Theorem 9. *There is a polynomial time reduction from 3SAT to QO_N with the following properties.*

1. *Satisfiable formulas map to instances of QO_N such that there is a join sequence Z with cost $C(Z) \leq K_{c,d}$ (α,n).*

2. *Unsatisfiable formulas map to instances of QO_N such that every join sequence Z has cost $C(Z) \geq K_{c,d}(\alpha, n)\cdot \alpha^{\Theta(n)}$.*

3. *$\log\ K_{c,d}(\alpha, n) = \Theta(n^2 \log\ \alpha)$.*

where n is the number of vertices in the query graph produced by the reduction function.

By setting $\alpha(n) = 4^{n^{1/\delta}}$, for any constant $\delta > 0$, the gap function $\alpha^{\Theta(n)} = 2^{\Theta(n)n^{1/\delta}}$, which in terms of $K_{c,d}(\alpha(n)$ gives a a complexity gap of $2^{Theta(log^{1-\delta} K)}$. Thus, solving the QO_N problem approximately using a polynomial time algorithm with a competitive ratio better than $2^{\Theta(log^{1-\delta} K)}$, for any $\delta > 0$ would imply that P=NP.

It is apparent from the construction of the reduction function that the existence of cartesian products in the join sequence serves only to increase the cost. Hence, even if we had restricted the join sequences in the problem definition of QO_N to have no cartesian products, the same complexity gap would be obtained.

5. COMPLEXITY OF QO_H

In this section, we present a reduction f_H from $\frac{2}{3}$CLIQUE to QO_H.

Let $G = (V, E)$ be an instance of $\frac{2}{3}$CLIQUE, where $V = \{v_1, v_2, \ldots, v_n\}$. The instance $f_H(G)$ of QO_H is constructed as follows.

1. The query graph is $G' = (V', E')$, where $V' = V \cup \{v_0\}$. The vertex v_0 corresponds to a new relation R_0 and is joined by an edge in E' to every vertex $v_i \in V$, $1 \leq i \leq n$.

2. Let $\alpha = \Omega(4^n)$ and $t = \alpha^{(n-1)/2}$. The number of tuples in the relations corresponding to the vertices in V are set to t. The number of tuples for the relation R_0, that is, t_0, is set to $\Theta((nt)^{1/\phi})$.

3. The selectivities s_{jk} and s_{kj} corresponding to a pair of vertices $\{v_j, v_k\} \in E$ are set to $1/\alpha$. The selectivity of the join predicate corresponding to the edges between v_0 and v_i, for $1 \leq i \leq n$, is set to $1/2$. The selectivity corresponding to pairs of vertices not connected by an edge in E is set to 1.

4. The available main memory M is set to $(n/3 - 1)t + 2 \cdot hj_{min}(t)$ pages.

We let $L(\alpha, n)$ denote the expression $t_0\alpha^{(n^2/9)}$ and let $G(\alpha, n)$ denote the expression $t_0\alpha^{(n^2/9+n\epsilon/3-1)}$, where ϵ is a constant < 1 mentioned in the statement of Lemma 4

Note that the construction of t_0 is designed so that $M < hj_{min}(t_0)$, that is, the available memory M is less than the minimum memory required to build a hash table on the relation R_0. Thus the only feasible join sequences are the ones where v_0 appears as the first element of the sequence. To simplify further discussions, we do not introduce the notion of a page size in the reduction. We assume that tuples of the base relation R_0 and the result of joining R_0 with any subset of the remaining relations results in tuples that span exactly one page. The following result is an elementary lemma concerning the optimal memory allocation to join operations in a pipeline and the resulting cost of the pipeline under the given memory allocation.

Lemma 10. *Let $G = (V, E)$ be an instance of $\frac{2}{3}$CLIQUE and consider the instance I of QO_H obtained by applying the reduction function f_H to G. Let Z be a join sequence for I beginning with v_0. Consider a pipeline $P(Z, i, k)$, and let the join operations in P with the lowest and the second lowest sizes of the outer relations be $J_j(Z)$ and $J_l(Z)$ respectively.*

1. *if $(k-i+1) \leq (n/3-1)$, the optimal memory allocation allocates t pages to all the join operations. The cost of the pipeline is $\Theta(N_{i-1}(Z) + N_k(Z))$.*
2. *if $(k-i+1) = n/3$, the optimal memory allocation allocates $hj_{min}(t)$ pages to the join operations $J_j(Z)$ and t pages to all the other join operations. The cost of the pipeline is $\Theta(N_{i-1}(Z) + N_k(Z) + N_{j-1}(Z))$.*
3. *If $(k-i+1) = n/3+1$, then the optimal memory allocates $hj_{min}(t)$ pages to join operations $J_j(Z)$ and $J_l(Z)$, and t pages to all other join operations. The cost of the pipeline is $\Theta(N_{i-1}(Z) + N_k(Z) + N_{j-1}(Z) + N_{l-1}(Z))$.*

Proof Sketch. Item 1 is self-evident. The claims in items 2 and 3 about the optimal memory allocation follows from the linearity of the cost function $g(m, b_s)$ as a function of m. By the definition of the cost function h in Section 2.2, it follows that the cost of a pipeline is proportional to the sum of the sizes of the outermost relation, the result size of the pipeline and the sum of the inner relations corresponding to the join operations that were allocated minimum memory. $\square$

The following lemma is an auxiliary lemma that places upper bounds on the sizes of the intermediate relations.

Lemma 11. *Let $G = (V, E)$ be an instance of $\frac{2}{3}$CLIQUE such that there exists a clique $C \subset V$ of size $2n/3$, where $n = |V|$. Let Z be a join sequence for the constructed instance $f_H(G)$ of QO_H which begins with v_0, followed by the vertices in C in some order, followed by the vertices in $V - C$. Then, each of $N_1(Z)$, $N_{n/3}(Z)$, $N_{2n/3}(Z)$, $N_{n-1}(Z)$ and $N_n(Z)$ is $O(L(\alpha, n))$.*

Proof. Let $1 \leq j \leq 2n/3$. Consider the set of vertices U_j that occupy positions 2 through $j+1$ positions in Z. The vertex v_0 has j edges to the vertices in U_j and each of these edges has a selectivity of 1/2. The vertices of U_j form a clique of size j, and hence there are $j(j-1)/2$ edges, each with selectivity α. The product of the selectivities is $\alpha^{-j(j-1)/2}2^{-j}$. The product of the sizes of the relations is $t_0 t^j$. Thus

$$N_j(Z) = t_0 t^j \alpha^{-j(j-1)/2} 2^{-j}$$

Substituting $t = \alpha^{(n-1)/2}$ and noting that $2^{-j} < 1/\alpha$, we get

$$\begin{aligned} N_j(Z) &= t_0 \alpha^{(j(n-1)/2 - j(j-1)/2)} 2^{-j} \\ &< t_0 \alpha^{(j(n-1)/2 - j(j-1)/2)}/\alpha \\ &= t_0 \alpha^{(j(n-j)/2-1} \end{aligned}$$

Thus

$$\begin{aligned} N_{n/3}(Z) &= O(t_0 \alpha^{(n/3)(2n/3)(1/2)} the) = O(t_0 \alpha^{n^2/9}) \\ &= O(L(\alpha, n)) \end{aligned}$$

Replacing j by 1 and $2n/3$ yields the bounds for N_1 and $N_{2n/3}$ as claimed.

We now estimate upper bounds for $N_{n-1}(Z)$ and $N_n(Z)$. By definition, $N_n(Z) = t_0 t^n \alpha^{-|E|} 2^{-n}$. Since the last vertex in Z may be connected to at most $(n-1)$ other vertices in V, it follows that $N_{n-1}(Z) \leq t_0 t^{n-1} \alpha^{-|E|+(n-1)} 2^{-(n-1)} = N_n(Z)\alpha^{(n-1)/2}$. Since $|E| = n^2/2 - \Theta(n)$ (degree of each vertex is at least $n-14$), the claim of the lemma follows. $\square$

Lemma 12. *Suppose the instance $G = (V, E)$ of $\frac{2}{3}$CLIQUE has a clique C of size $\geq \frac{2n}{3}$. Then the constructed instance I of of QO_H has a join sequence Z and a pipeline decomposition P whose cost is $O(L(\alpha, n))$.*

Proof. Let Z be any sequence that begins with v_0 followed by the $2n/3$ vertices of C in any order, followed by the vertices in $V - C$ in some order.

Consider a pipeline decomposition of Z with five pipelines, namely, $P_1(1,1)$, $P_2(2, n/3)$, $P_3(n/3+1, 2n/3)$, $P_4(2n/3+1, n-1)$ and $P_5(n, n)$. The sizes of the intermediates generated by this decomposition of pipelines is $N_1(Z)$, $N_{n/3}(Z)$, $N_{2n/3}(Z)$, $N_{n-1}(Z)$ and the final result $N_n(Z)$. By Lemma 11, each of these sizes is bounded by $O(L(\alpha, n))$.

The pipelines P_1, P_2, P_4 and P_5 have no more than $(n/3-1)$ join operations and the available memory $(n/3-1)t + hj_{min}(t)$ is sufficient for each of these join operations. Therefore, in these four pipelines, there is no additional cost incurred due to insufficient memory allocation.

We now consider the pipeline P_3, which has $n/3$ join operations. By Lemma 10, the join that uses the the smallest size relation as its outer relation is given the least memory. Therefore, the join operation $J_{n/3+1}(Z)$ is given the minimum memory $hj_{min}(t)$ and its cost becomes $\Theta(N_{n/3}(Z))$. Thus the cost of pipeline P_3 is $\Theta(N_{n/3}(Z) + N_{2n/3}(Z))$, which, by Lemma 11 is $O(L(\alpha, n))$. It follows that the total cost of the pipeline decomposition P_1 through P_5 for the join sequence Z is $O(L(\alpha, n))$. $\square$

We now consider the family of instances $G = (V, E)$, of $\frac{2}{3}$CLIQUE whose largest clique size is no more than $(2-\epsilon)n/3$, where $n = |V|$.

Lemma 13. *Let $G = (V, E)$ be an instance of $\frac{2}{3}$CLIQUE whose largest clique size is no more than $(2-\epsilon)n/3$, where $n = |V|$. Let Z be any feasible sequence of the constructed instance of QO_H, such that Z begins with v_0. For $1 \leq j \leq m/3$, $N_{m/3+j}(Z) = \Omega(G(\alpha, n))$.*

Proof. Let $1 \leq j \leq (n/3-1)$. Consider the set of vertices $U_{n/3+j}$ that occupy positions 2 through $n/3+j+1$ positions in Z and let $E_{n/3+j}$ denote the set of edges in E among vertices in $U_{n/3+j}$. The vertex v_0 has j edges to the vertices in U_j and each of these edges has a selectivity of 1/2. Let $D_{n/3+j}$ denote the number of edges in $E_{n/3+j}$.

By construction, the product of the relation sizes corresponding to the first $n/3+j$ join operations is $t_0 t^{n/3+j}$. The product of the selectivities of join operations consists of the selectivities contributed by the set $E_{n/3+j}$ of edges. The total contribution of these edges to $N_{n/3+j}(Z)$ is $\alpha^{-D_{n/3+j}(Z)}$. A set of $n/3+j$ edges connecting v_0 to the vertices in $U_{n/3+j}$ contributes a total factor of $2^{-(n/3+j)}$ towards $N_{n/3+j}(Z)$ $N_{n/3+j}(Z) = t_0 t^{n/3+j} \alpha^{-D_{n/3+j}(Z)} 2^{-n/3-j}$. If $1 \leq j \leq n/3-1$, then $D_{n/3+j}(Z) \leq (n/3+j)(n/3+j-1)/2$. Substituting this expression in place of $D_{n/3+j}$ in the expression for $N_{n/3+j}(Z)$ and noting that t is $\alpha^{(n-1)/2}$, we get $N_{n/3+j}(Z) = t_0 \alpha^{(n/3+j)(n-1)/2 - (n/3+j)(n/3+j-1)/2} 2^{-n/3-j}$. Since $2^{-n} > 1/\alpha$, we get

$$\begin{aligned} N_{n/3+j}(Z) &= \Omega(t_0 \alpha^{(n/3+j)(n-1)/2 - (n/3+j)(n/3+j-1)/2) - 1}) \\ &= \Omega(t_0 \alpha^{(n/3+j)(2n/3-j)/2-1}) \end{aligned}$$

Note that the minimum value of $(n/3+j)(2n/3-j)$ in the range $1 \le j \le n/3-1$ is obtained for $j=1$ and $j=n/3-1$. Substituting $j=1$, we get

$$N_{n/3+j}(Z) = \Omega(t_0\alpha^{(n^2/9+n/2-1)}) = \Omega(G(\alpha,n)), 1 \le j < n/3.$$

We now consider $N_{2n/3}$. Since the largest clique of G has size at most $(2-\epsilon)n/3$, it follows from Lemma 7 that for any feasible sequence Z,

$$\begin{aligned} D_{2n/3}(Z) &\le (2n/3)(2n/3-1)/2-(2n/3)+(2-\epsilon)n/3 \\ &= n^2/9-n/3-n\epsilon/3. \end{aligned}$$

Thus,

$$\begin{aligned} N_{2n/3}(Z) &\ge t_0t^{2n/3}\alpha^{-(n^2/9+n\epsilon/3)}2^{-2n/3} \\ &\ge t_0\alpha^{(n^2/9+n\epsilon/3-1)} = G(\alpha,n). \end{aligned}$$

□

We are now in a position to prove that instances of the $\frac{2}{3}$CLIQUE problem which have cliques of size less than $(2-\epsilon)n/3$ are mapped to instances of QO_H whose join sequences have cost $\Omega(G(\alpha,n))$ thereby proving the complexity gap.

LEMMA 14. *Suppose the instance $G=(V,E)$ of $\frac{2}{3}$CLIQUE problem has no clique of size $(2-\epsilon)n/3$, where $n=|V|$ and consider the instance I of QO_H produced by applying the reduction function f_H. Then, every join sequence Z and its decomposition into a sequence of pipelines has cost $\Omega(G(\alpha,n))$.*

PROOF. Let Z be any join sequence for the instance I of QO_H. Consider the join operations $J_{n/3+1}(Z)$, $J_{n/3+2}(Z)$, $\dots$, $J_{2n/3+1}(Z)$. Since, by Lemma 13, the output sizes of the join operations $J_{n/3+j}(Z)$ is $\Omega(G(\alpha,n))$, for $1 \le j \le n/3$, it follows that the above $n/3+1$ join operations must be placed in a single pipeline. We now consider the problem of optimal distribution of the available memory M.

The total available memory is $(n/3-1)t+2hj_{min}(t)$, and thus, two join operations must be allocated minimum memory and the rest must be allocated memory t. The smallest outer relation is the input to $J_{n/3+1}(Z)$ of size $\Theta(L(\alpha,n)$, and so we must allocate $hj_{min}(t)$ to the join $J_{n/3+1}(Z)$. However, all the other $n/3$ intermediates, that is, the outputs of join operations $J_{n/3+1}(Z)$ through $J_{2n/3}(Z)$ have size at least $\Omega(G(\alpha,n))$. It follows from Lemma 10 that by allocating minimum memory to any of the join operations $J_{n/3+j+1}(Z)$, for $j \in \{1,3,\dots,n/3\}$, the cost becomes $N_{n/3+j}(Z)=\Omega(G(\alpha,n))$. We conclude that, for every memory allocation, the cost of the pipeline decomposition is $\Omega(G(\alpha,n))$. □

By composing f_H with the reduction function defined in Lemma 4, we obtain the following complexity result.

THEOREM 15. *There is a polynomial time reduction from 3SAT to QO_H with the following properties.*

1. *Satisfiable formulas map to instances of QO_H such that there is a feasible join sequence Z, a pipeline decomposition of Z and a memory allocation vector such that the cost is $O(L(\alpha,n))$.*

2. *Unsatisfiable formulas map to instances of QO_H such that for every feasible join sequence Z, all pipeline decompositions have a cost $\Omega(G(\alpha,n))$.*

3. *$log\ L(\alpha,n)=\Theta(n^2 log\ \alpha)$ and $G(\alpha,n)=L(\alpha,n)\alpha^{\Theta(n)}$.*

where $n+1$ is the number of vertices in the query graph produced by the reduction function.

Note that if $\alpha=4^n$, then $log\ \alpha=\Theta(n)$ and hence $log\ L(\alpha,n)=\Theta(n^3)$. Thus, $G(\alpha,n)=L(\alpha,n)2^{\Theta(log^{2/3}(L(\alpha,n)))}$. The bound of $2^{log^{2/3}(L)}$ may be further improved as in the problem QO_N by letting α to be $4^{n^{1/\delta}}$, for any constant $\delta>0$. This yields a gap factor of $2^{log^{1-\delta}L}$, as claimed in Section 1. Thus, designing a polynomial time algorithm that approximates QO_H with a competitive ratio that is better than a factor $2^{log^{1-\delta}L}$, for any constant $\delta>0$ is equivalent to showing that P=NP.

6. COMPLEXITY OF QO_N AND QO_H FOR SPARSE QUERY GRAPHS

In Sections 4 and 5, we presented polynomial time reductions of problems CLIQUE and $\frac{2}{3}$CLIQUE to QO_N and QO_H respectively using reduction functions f_N and f_H. A common characteristic of these reductions is that the query graph of the resulting instance is a dense graph, namely, the number of edges in the graph are $n^2/2-\Theta(n)$. This observation leaves open the possibility of finding efficient approximation algorithms for problems QO_N and QO_H for sparse query graphs with better competitive ratios than the gaps proved in Sections 4 and 5 respectively.

In this section, we show that, for any (real) constant $\tau>0$, the complexity of finding the approximate optimal solution for problems QO_H and QO_N remain unchanged from the previous sections, when the input is restricted to query graphs, whose number of edges is a given function $e(n)$ of the number of vertices n in the query graph, such that $n(n-1)/2-\Theta(n^\tau) \ge e(n) \ge n+\Theta(n^\tau)$, where τ is any constant $0<\tau<1$.

6.1 Complexity of QO_N for sparse query graphs

We define a reduction $f_{N,e}$ that maps instances of CLIQUE to instances of QO_N whose query graph has m vertices and $e(m)$ edges. Choose $k=\Theta(2/\tau)$. Let $G_1=(V_1,E_1)$ be an instance of CLIQUE where $|V_1|=n$. We construct an auxiliary connected undirected graph $G_2=(V_2,E_2)$ where $|V_2|=n^k-n$ and $|E|=e(n^k)-|E_1|-1$. We then construct the query graph Q as $Q=(V,E)$, where $V=V_1\cup V_2$ and $E=E_1\cup E_2\cup\{v_1,v_2\}$, where v_1 and v_2 are arbitrary vertices from V_1 and V_2 respectively.

From the above construction, it is clear that $|V|=n^k$ and $|E|=e(n^k)$, thus, the query graph Q has the desired form. Further, G_2 can be constructed to be a connected graph, since, we can choose $k=\Theta(2/\tau)$ such that $|E_2|=e(n^k)-|E_1|-1 \ge n^k-n-1$ edges.

Let $\beta=4$, and let $\alpha=\beta^{n^{2k+2}}$. The size of each relation corresponding to nodes in V_1 is set to be the same as in our reduction f_H of Section 4 (that is, $t=\alpha^{(c-d/2)n}$). The size of each relation corresponding to nodes in V_2 is set to $u=\beta^n$. The selectivity factor of each edge is set to $1/\alpha$ for each edge in E_1 and to $1/\beta$ for each edge in E_2. The selectivity factor corresponding to the edge $\{v_1,v_2\}$ between vertex sets V_1 and V_2 is also set to $1/\beta$. We set w_{jk} to be t/α if the edge $\{v_j,v_k\}\in E$ and $v_j\in V_1$. We set w_{jk} to u/β if the edge $\{v_j,v_k\}\in E$ and $v_j\in V_2$. The property of the reduction $f_{N,e}$ is explained in the following theorem.

Theorem 16. *For every $\tau > 0$, There exists a polynomial time reduction $f_{N,e}$ that maps instances $G_1 = (V_1, E_1)$ of CLIQUE with n vertices to instances I of QO_N whose query graph $Q = (V, E)$ has m vertices and $m(m-1)/2 - \Theta(m) \geq e(m) \geq m + \Theta(m^\tau)$ edges, with the following properties.*

1. *$k = \Theta(2/\tau)$, $m = \Theta(n^k)$ and $\alpha = \Omega(4^{n^{2k+2}})$.*
2. *If G_1 has a clique of size at least cn, then there exists a join sequence Z for I such that $C(Z) \leq K_{c,d}(\alpha, n)$.*
3. *If G_1 has cliques of size at most $(c-d)n$, then for all join sequences Z for I, $C(Z) > K_{c,d}(\alpha, n)\alpha^{dn/2-1}$.*

Proof SKETCH. The size of the cartesian product of the relations in G_2 is small. It is bounded above by $\beta^{n \cdot n^k} = \beta^{n^{k+1}} = \alpha^{1/n^{k+1}}$. Similarly, the product of all the selectivities in G_2 is bounded below by $\beta^{-n^k \cdot n^k} = \beta^{-n^{2k}} > 1/\alpha$. Due to these properties, the selectivities corresponding to the edges of the auxiliary graph G_2 and the relations corresponding to nodes of G_2 contribute at most a factor of $\alpha^{\Theta(1)}$ to the costs $H_i(Z)$ where the i^{th} join involves a vertex from V_1. Thus, by extending the graph, we have introduced a factor of at most $\alpha^{\Theta(1)}$ to the cost. □

6.2 Complexity of QO_H for sparse query graphs

We define a reduction $f_{H,e}$ from instances of $\frac{2}{3}$CLIQUE to instances of QO_H with query graphs whose number of edges is a given function $e(n)$ of the number of vertices n. The reduction $f_{H,e}$ bears a correspondence with the reduction function f_H (presented in Section 5), that is very similar to the correspondence between the reduction functions $f_{N,e}$ and f_N.

Choose $k = \Theta(2/\tau)$. Let $G_1 = (V_1, E_1)$ be an instance of $\frac{2}{3}$CLIQUE where $|V_1| = n$. We construct an auxiliary connected undirected graph $G_2 = (V_2, E_2)$ where $|V_2| = n^k - n - 1$ and $|E| = e(n^k) - |E_1| - n - 1$. We then construct the query graph Q as $Q = (V, E)$, where $V = V_1 \cup \{v_0\} \cup V_2$ and $E = E_1 \cup E_2 \cup \{v_1, v_2\} \cup \{(v_0, u) | u \in V_1\}$. The vertex v_0 is a new vertex not appearing in V_1 or V_2 and v_1 and v_2 are two arbitrary vertices from V_1 and V_2 respectively.

By the above construction, $|V| = n^k$ and $|E| = e(n^k)$, thus, the query graph Q has the desired form. Further, G_2 can be constructed to be a connected graph by the same argument as in Section 6.1.

We let $\alpha(n) = \Omega(4^{n^{2k+2}})$ be a polynomially computable function of n. The number of tuples in the relations corresponding to V_1 is set to $t = \alpha^{(n-1)/2}$ as in the reduction f_H. The number of tuples in the relation R_0, corresponding to the vertex v_0 is set to $\Theta((nt)^{1\phi})$. The available main memory M is set as before to $(n/3 - 1)t + hj_{min}(t)$ pages. The selectivities of each edge in E_1 is set to $1/\alpha$. The selectivity of edges between v_0 and vertices in V_1 is set to $1/2^n$. The sizes of relations corresponding to vertices in V_2 is set to 2^n. The selectivities of all edges in E_2 and the edge $\{v_1, v_2\}$ connecting V_1 with V_2 is set to $1/2$.

Note that the subgraph of G comprising the vertex set $V_1 \cup \{v_0\}$ is essentially a copy of the graph that would be obtained by applying the reduction function f_H to the instance G_1, except for using a larger value of α. As discussed in Section 6.1, the cartesian product of the relations corresponding to the vertices in V_2 have a size that is insignificant compared to t. Similarly, the product of the selectivities of edges in E_2 is $\Omega(2^{-n^{2k}/2}) = \Omega(\alpha^{0.5})$. Thus, the new vertices and the edges do not play any significant role in the cost of join sequences. We therefore state the following theorem without further ado.

Theorem 17. *For every $\tau > 0$, there exists a reduction function $f_{H,e}$ that maps instances $G_1 = (V_1, E_1)$ of $\frac{2}{3}$CLIQUE with n vertices to instances I of QO_N whose query graph $Q = (V, E)$ has m vertices and $m(m-1)/2 - \Theta(m^\tau) \geq e(m) \geq m + \Theta(m^\tau)$ edges, with the following properties.*

1. *$k = \Theta(2/\tau)$, $m = \Theta(n^k)$ and $\alpha = \Omega(4^{n^{k+1}})$*
2. *If G_1 has a clique of size at least $2n/3$, then there exists a join sequence for I such that its cost is at least $L(\alpha, n)$.*
3. *If G_1 does not have a clique of size less than $(2-\epsilon)n/3$, then all join sequences have cost $\Omega(G(n, \alpha))$.*
4. *log $L(n, \alpha) = \Theta(n^2 log\ \alpha)$ and $G(\alpha, n) = L(\alpha, n)\alpha^{\Theta(n)}$.*

6.3 Comments

By composing the reduction functions $f_{N,e}$ (respectively $f_{H,e}$) with the reduction function of Lemma 3 (respectively, Lemma 4), and letting $\alpha = \Theta(4^{n^{(k+1)/\delta}})$, we conclude that designing a polynomial time approximation algorithm for instances of QO_N (resp. QO_H) whose query graphs have m vertices and $e(m)$ edges, where, $m(m-1)/2 - \Theta(m^\tau) \geq e(m) \geq m + \Theta(m^\tau)$ edges, and whose competitive ratio is at most $2^{log^{1-\delta} L}$, where L is the optimal cost, for any value of $1 > \tau > 0$ and $\delta > 0$ is equivalent to $P = NP$.

The work presented in [1] and [6] present polynomial time algorithms for tree queries for a problem which is very similar to QO_N. It is therefore an interesting question to ask if polynomial time algorithms could be designed for optimizing queries with connected query graphs which have edges other than tree edges. The results of this section place limits on an affirmative answer to the above question. First note, that since $e(m) \geq m + \Theta(m^\tau)$, for any constant τ, the only possible family of queries that could be optimized in polynomial time are the ones which have $m + o(m^\tau)$ edges, for any $\tau > 0$.

7. CONCLUSIONS

In this paper, we study the complexity of two variants of the query optimization problem, namely, QO_N and QO_H.

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. Further, the complexity result remains true, when the number of edges in the query graph of the instances for QO_N and QO_H is constrained to match a given function $e(n)$ of the number of vertices n of the query graph, where $n(n-1)/2 - \Theta(n) \geq e(n) \geq m + \Theta(m^\tau)$, for any constant $0 < \tau < 1$.

These results show that, unless P=NP, the query optimization problem cannot be approximately solved by an algorithm that runs in polynomial time and has a competitive ratio that is within any polylogarithmic factor of the optimal cost in polynomial time.

8. REFERENCES

[1] Toshihide Ibaraki and Tiko Kameda. On the optimal nesting order for computing n-relational joins. *ACM Transactions on Database Systems*, 9(3):482–502, September 1984.

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

[3] Chih-Ping Wang and Ming-Syan Chen. On the complexity of distributed query optimization. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):650–662, August 1996.

[4] Sanjeev Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Algorithms.* Technical Report TR-476-94. Department of Computer Science, Princeton University, 1994.

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

[6] R. Krishnamurthy, H. Boral and C. Zaniolo. Optimization of non-recursive queries. In *Proceedings of the International Conference on Very Large Databases.* Morgan Kaufmann, 1986.

[7] S. Chatterji, S.S.K. Evani, S. Ganguly and M.D. Yemmanuru. On the complexity of approximate query optimization. Lucent Internal Technical Document ITD-02-42549Z, 2002. Lucent Technologies.

APPENDIX

A. COMPLEXITY OF SQO-CP

In this section, we formulate a variant of the query optimization problem for star query graphs in which cartesian products among relations are not allowed and joins are allowed to be performed using both the nested-loops join method and the merge-sort method. Determining the complexity of this problem was posed as a question in [1]. In this appendix, we show that the above problem is NP-complete. 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 $0 \le r \le m$.

Let P_i denote the predicate between R_0 and R_i, for $1 \le i \le m$. 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 restrict the scope of the problem. 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.

We assume that the query is of the following 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.

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

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.

$$\begin{array}{lcl} D(\phi, R_r N_i Y) & = & \begin{cases} b_0 + w_i \cdot n_0 + D(R_0 N_i, Y) & r = 0 \\ b_r + w_{0,r} \cdot n_r + D(R_r N_0, Y) & r \neq 0 \end{cases} \\ D(\phi, R_r S_i Y) & = & C_{sm}(R_r, R_i) + D(R_r S_i, Y) \\ D(W, S_i Y) & = & b(W) \cdot (k_s - 1) + A_i + D(W S_i, Y) \\ D(W, N_i Y) & = & n(W) \cdot w_i + D(W N_i, Y) \\ D(W, \phi) & = & 0 \end{array}$$

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 <u>S</u>ubset <u>P</u>roduct <u>P</u>lus <u>C</u>omplement <u>S</u>um.

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 [5] 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 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 $K = \sum_{i=1}^{n} b_i$.

- $p = \lfloor log_2 K \rfloor + 1$, $q = 2p + 7 + n$
- $S = g_{nq}(K/2)$
- $m = 2n$
- 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 the full version of this paper [7].

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 the full version of this paper [7].