Title

A Global Heuristic for Distributed Join Operations

Author

Segev, A

Publication Date

1989-07-01

LBL-26441
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

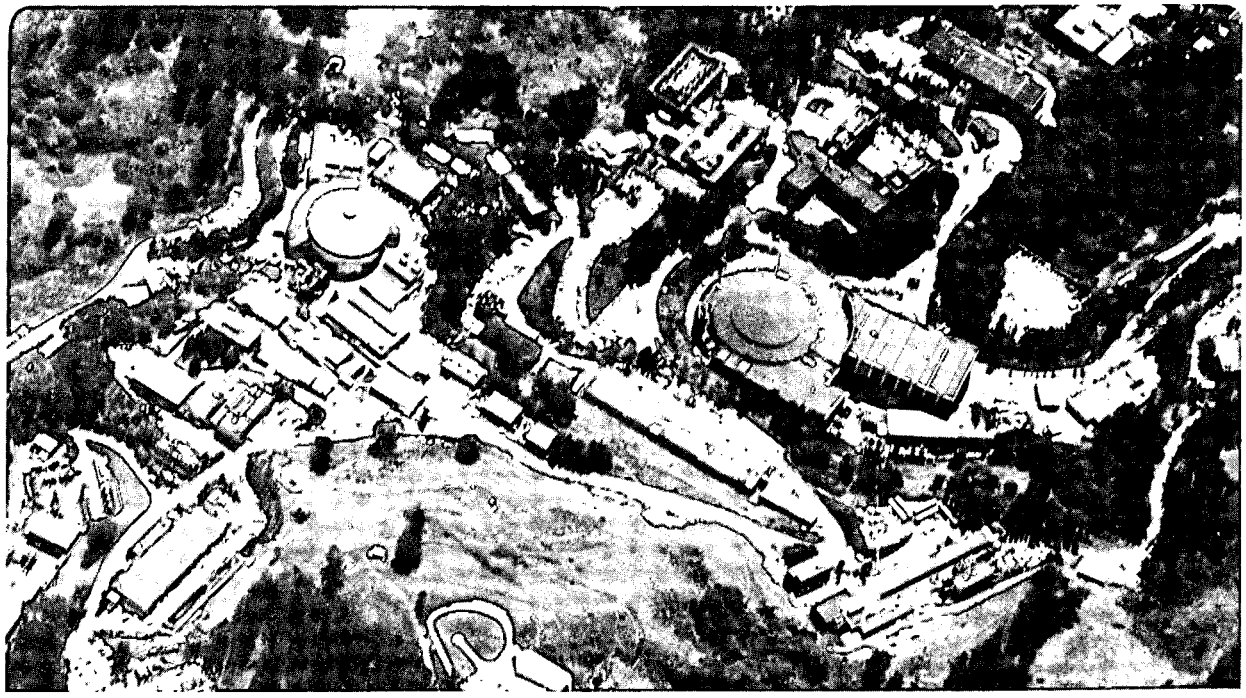
Information and Computing Sciences Division

Submitted to INFOR

A Global Heuristic for Distributed Join Operations

A. Segev

July 1989



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

DISCLAIMER

LBL-26441

A Global Heuristic for Distributed Join Operations

Arie Segev

School of Business Administration
University of California

and

Computing Science Research & Development
Information & Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, California 94720

July 1989

This work was supported by the Director, Office of Energy Research, Applied Mathematical Sciences Research Program, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

A Global Heuristic for Distributed Join Operations

Arie Segev

School of Business Administration
University of California at Berkeley
and
Computer Science Research Department
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, California, 94720

Abstract

This paper addresses the problem of join query optimization in distributed relational database systems. It concentrates on the development of heuristic procedures that attempt to minimize the communication costs incurred by the distributed processing of queries. In particular, the paper deals with a class of heuristics that use a semi-join strategy as the mechanism for communication cost reduction. These heuristics are classified into two types -- local and global heuristics. The global heuristic proposed in this paper is based on an optimal solution to a mathematical model of a relaxed version of the problem. We develop a branch and bound procedure to derive that optimal solution. It is shown that the global heuristic can identify beneficial semi-join operations not included in local heuristic solutions.

(To appear in INFOR)

This work was supported by the Applied Mathematical Sciences Research Program of the Office of Energey Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

1. Introduction

The dramatic decrease in hardware costs and advances in telecommunications have led many organizations to distribute their data processing activities and resources. Distributed processing offers enhanced availability, reliability, parallelism, reduced system costs, and better responsiveness to user needs. To realize those benefits, however, numerous problems have to be solved, including processor selection, file and data allocation, concurrency control, backup and recovery, and query optimizations. For surveys of those problems, see [DOWD82], [HAER83], [JARK84], and [ROTH77].

This paper deals with the problem of query optimization in the context of relational [CODD70] distributed data base management systems (DDBMS). The problem is how to satisfy data retrieval requests in a way that optimizes a given performance measure. The measures to be optimized can be communication cost, processing cost, and response time. Models developed to solve various distributed query optimization problems include [APER83], [BERN81], [CHAN82], [EPST79], [GAVI86], [SEGE86], [SELI80], [WONG77], [YU83], and others (for surveys, see see [JARK84] and [YU84]). The join operation is the most resource consuming operation in a relational database. Therefore, the performance of a DDBMS is highly dependent on the join optimization procedure. There are two basic strategies for optimizing joins; the first performs the join operations directly, while the second precedes the joins by semi-join operations [BERN81b] whenever beneficial. The effect of semi-joins is to reduce the size of the join operands. In the context of DDBMS, semi-joins where introduced to reduce communication costs. In [SEGE88], it is shown that semi-joins also reduce the processing costs associated with distributed joins in many instances.

This paper concentrates on the development of heuristic procedures that attempt to minimize the communication costs (or the amount of data transmitted) incurred by the distributed processing of join queries. In particular, the paper deals with a class of heuristics that use a semi-join strategy as the mechanism for communication cost reduction (examples for such heuristics are [APER83], [BERN81], and [HEVN79]). The heuristics that have been

proposed so far can be classified as ***local*** heuristics in the sense that their steps are based on local measures. Most of the local heuristics are "greedy" heuristics, where at each step the most "locally" beneficial semi-join is added to the set of selected semi-joins. The contribution of this paper is in the application of a ***global*** heuristic to the distributed join optimization problem. It is global in the sense of basing the semi-join selection on an optimal solution to a simplified version of the original problem. The simplified problem retains some global characteristics of the original problem (hence the term "global heuristic" and the optimal solution to the first is feasible to the latter. We show that the global heuristic can generate beneficial semi-join operations not included in the solutions generated by local heuristics.

This paper is organized as follows. Section 2 presents the problem of optimizing distributed joins and distinguishes local heuristics from global heuristics. The mathematical notation used in the paper is explained in Section 3 followed by a mathematical analysis of the global heuristic in Section 4. The results of computational experiments are reported in Section 5, and the paper is summarized in Section 6.

2. Local vs. Global Heuristics

The problem addressed in this paper is illustrated in Figure 1. Two relations R1 and R2 are stored at sites 1 and 2 respectively. R1 contains employee data (employee number, name, and the department number in which the employee works), and R2 contains data about departments and projects (department number and project number combinations). If a query at site 3 requires the join of R1 and R2 on D#, the shown RESULT relation has to be displayed at site 3. The join operation concatenates rows (or tuples) of R1 and R2 whenever the D# values are equal. An obvious way of executing the query is to send R1 and R2 to site 3 where they are joined. Another option is to send R2 to site 1 (or R1 to site 2), join them there and send the result to site 3. The semi-join strategy is used to eliminate non-qualifying tuples from the operand relations. The result of semi-joining R1 by R2 is shown as $\overline{R1}$ in Figure 1. It is achieved by projecting the D# column from R2 (eliminating duplicates)

and sending it to site 1 where it is joined with R1. Note that the semi-join operation is asymmetric; a semi-join of R2 by R1 (shown as $\overline{R2}$ in the figure) does not reduce the size of R2.

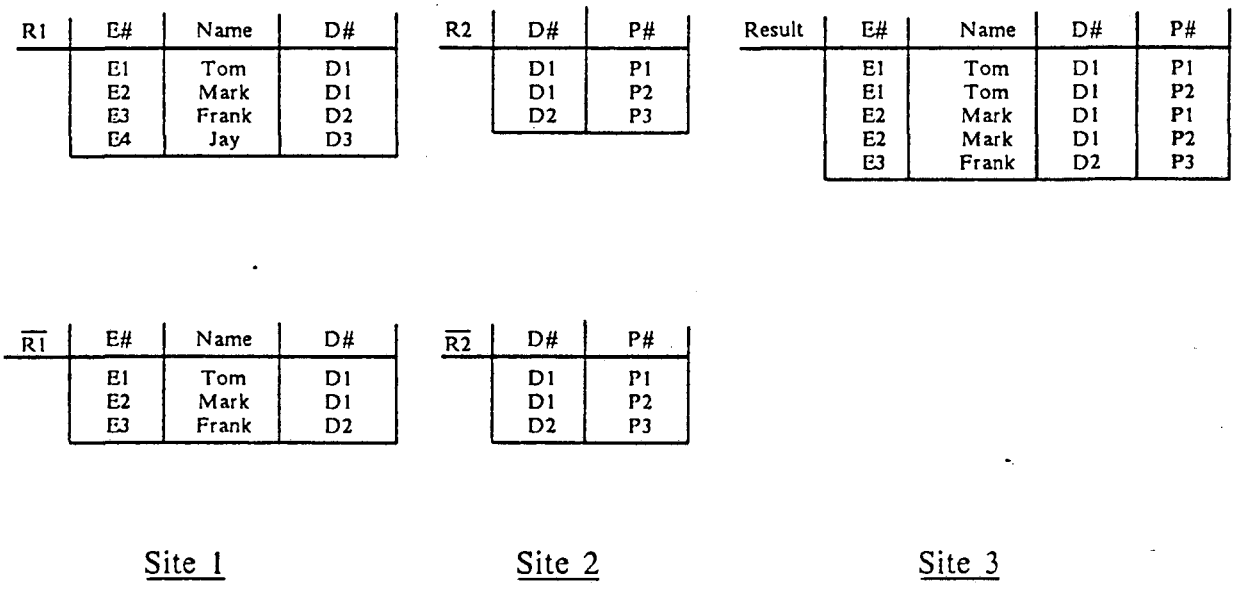


Fig. 1: An Example of Join and Semi-Join Operations

The problem of distributed join optimization is highly complex and various versions of the problem were proved to be NP-Complete (see [SEGE84]). For example, if n relations are referred to by the query and there are m join attributes, then the number of possible semi-joins is (assuming that every relation contains the m join attributes) $S = n(n-1)m$, the number of feasible sets of semi-joins is $2^{n(n-1)m}$, and the number of feasible semi-join strategies is $ST = \sum_{i=0}^{S} \binom{S}{i} i!$. If $n = m = 4$, then $S = 48$ and $ST \cong 10^{62}$.

The complexity of the problem renders the use of complete enumeration (or a restricted enumeration like a Branch and Bound method) impractical for most queries that involve joins. Consequently, it is expected that heuristic rather than optimal solutions will be used for on-line real world applications. In those instances, mathematical modeling and the derivation of lower bounds for the problem are a useful mechanism to evaluate the performance of the heuristic procedures [SEGE86] because conducting a significant empirical analysis using complete enumeration for sizable problems is likely to be too costly for most researchers.

In this paper, we broadly classify heuristic procedures into two types: local heuristics and global heuristics. Local heuristics, which include query optimization heuristics proposed so far, are "greedy" procedures [HORO78]; they search for the optimal solution by choosing the "best" next solution, where "best" is defined in terms of a criterion relative to the current solution. Those algorithms terminate when no local improvement can be achieved. Many of the algorithms for distributed query optimization can be classified as ADD procedures, e.g. [BERN81], [CHAN82]. Those algorithms start with an empty set of semi-join operations and at each step add the most beneficial semi-join to the set. ADD procedures differ from each other in the set to be selected and the criteria for adding an element to the set. For some problems the selected set constitutes a solution, while for others additional decision variables have to be determined. As an example, consider a distributed join optimization problem. After a set of semi-join operations is selected, one might still have to allocate those operations to sites and to optimize the required transmissions (it would not be necessary if these decisions were made by the set selection procedure). The value of an ADD procedure is that for a given selected set it is often easy to find an optimal or good solution for the rest of the decision variables.

Global heuristics are derived by solving a mathematical model of a relaxed version of the problem, such that the resulting values of the decision variables are feasible for the original problem. To determine a semi-join strategy, two decisions have to be made. First, a subset of feasible semi-joins has to be selected, and second, the sequence of those semi-joins has to be determined.[1] Many of the query optimization algorithms determine the set of selected

semi-joins and their sequence simultaneously. The global heuristic proposed in this paper first relaxes the sequencing aspect of the problem and determines an optimal set of semi-joins for the relaxed problem. The sequence of those semi-joins is then determined by using an heuristic procedure. It is expected that such a global heuristic will generate beneficial semi-join operations not included in a local heuristic solution; in section 5, we present computational results that support this conjecture.

3. Notation

In what follows, we will assume that the problem involves a single join attribute; this assumption is for expository reasons only, and it is shown in Section 4.3 that the resulting mathematical model is also valid for the case of multiple join attributes. The following notation will be used henceforth:

$T =$: The index set of sites and relations.

$q =$: The index of the query site.

$C_{ij} =$: The transmission cost rate between site i and site j.

$R_i =$: Relation i.

$S_i =$: Size of relation i.

$D_i =$: Size of the projection of relation i on its join attribute.

$R_i \leftarrow R_j =$: A semi-join between R_i and R_j, where R_i is the relation to be restricted.

$\alpha_{ij} =$: The selectivity factor for $R_i \leftarrow R_j$, i.e., the size of the result of $R_i \leftarrow R_j$ is $S_i\alpha_{ij}$. If $i = j$, then $\alpha_{ij} = 1$.

If S is a set, $|S|$ is its cardinality.

Note: There is a one-to-one correspondence between a site number and relation number. For example, relation 4 is stored at site 4. That is why the same index set T is used for sites and relations. Frequently, we will use the notation "join attribute i" to mean "the projection of relation i on its join attribute"; that meaning will be obvious from the context.

Additional notation and definitions will be introduced as necessary.

4. The Global Heuristic

Consider the following query optimization strategy: for each relation i execute a set (possibly empty) of semi-joins $\{R_i \leftarrow R_j | j \in J \subseteq T,\ j \neq i\}$ and then send the restricted relation i to the query site where a final join is executed. The benefit associated with that set of semi-joins is given by

$$S_i C_{iq}\left(1 - \prod_{j \in J} \alpha_{ij}\right) .$$

$S_i C_{iq}$ is the cost of sending the unrestricted relation i to the query site. $S_i \prod_{j \in J} \alpha_{ij}$ is the size of relation i after all semi-joints have been executed. Note that under the assumption that the values of the join attributes and independently and uniformly distributed, the total reduction factor is the product of the selectivity factors (see [BERN81] for further details).

The cost associated with that set of semi-joins is

$$\sum_{j \in J} D'_j C_{ji},$$

where D'_j is the size of join attribute j used for the execution of $R_i \leftarrow R_j$. D'_j is not necessarily equal to D_j because relation j might have been reduced by the semi-join $R_j \leftarrow R_k$ prior to the execution of $R_i \leftarrow R_j$. In that case (assuming that R_j was not involved in restricting R_k), $D'_j = D_j \alpha_{jk}$.

It is evident from the above expressions that the benefit of a set of semi-joins is ***independent*** of the sequence of those semi-joins, while the cost of the set is ***dependent*** on the sequence (since D'_j is dependent on the extent to which R_j was restricted prior to $R_i \leftarrow R_j$). The idea behind the global heuristic is to remove the sequence dependency property from the mathematical model. This is done by assuming that the cost of a semi-join $R_i \leftarrow R_j$ is $D_j C_{ji}$, that is, the original attribute of relation j is used in the above semi-join. The effect of this relaxation is an overestimate of a semi-join cost. Section 4.1 presents a mathematical modeling and the solution of the sequence-independent version of the problem (D_j replaces D'_j in the cost arguments).

The solution generated in Section 4.1 consists of a feasible set of semi-joins which are then sequenced in Section 4.2.

4.1 A Mathematical Model of the Sequence-Independent Problem

The following decision variables are used in the mathematical formulation of the problem.

$$Y_{ij} = \begin{cases} 1, & \text{if relation } R_i \text{ is semi-joined by relation } R_j \\ 0, & \textit{otherwise} \end{cases}$$

Since the simplified problem is sequence-independent, the decisions regarding the restriction of the various relations are independent of each other. For each relation R_i, $i = 1, \ldots, |T|$, we have the following cost minimization non-linear integer problem.

Problem (NLP_i):

$$\text{Min}\{S_i C_{iq} \prod_{j \in T} \left(\max\{(1 - Y_{ij}), \alpha_{ij}\}\right) + \sum_{j \in T} D_j C_{ji} Y_{ij}\}$$

Subject to: $Y_{ij} \in \{0, 1\}$, $j \in T$

The first part of the objective function accounts for the cost of sending the restricted relation to the query site. The total restriction factor is represented by the product part, where $\max\{(1 - Y_{ij}), \alpha_{ij}\}$ assures that the selectivity factor $o \leq \alpha_{ij} \leq 1$ will be part of the product only if $Y_{ij} = 1$. The second part of the objective function accounts for the cost of selected semi-joins.

Problems (NLP_i) were solved by a branch and bound procedure and by a heuristic algorithm (denoted as Algorithm 1). Algorithm 1 which is described below was used as the upper bound in the branch and bound procedure. Algorithm 1 is not proven to be optimal for problems (NLP_i), but in all problem instances that we examined (200 cases) it generated the optimal solution.[2]

A Branch and Bound Procedure

The root level of the branch and bound tree is numbered zero. The leaf level is numbered $|T| - 1$. There are $|T| - 1$ decision variables since Y_{ii} can be arbitrarily set to either 1 or 0 (recall that $\alpha_{ii} = 1$). The tree nodes at level k represent the binary decisions for variable $Y_{i(k)}$, where (k) denotes an original relation number. We need to use (k) and not k since the order of the tree levels does not necessarily correspond to the original order of the relations. For a node at level k, let $\text{PRED}_d(k) = \{j | Y_{ij}$ correspond to nodes on the path from the root to node $d\}$; the variable Y_{ij} corresponding to node d is not included in $\text{PRED}_d(k)$. We divide $\text{PRED}_d(k)$ into two sets (note that T_0^k, T_1^k, $F(T_1^k)$, and B_j^k below are a function of node d; to simplify the notation we don't add an explicit reference to d):

$$T_0^k = \left\{ j \in \text{PRED}_d(k) | Y_{ij} = 0 \right\}$$

$$T_1^k = \left\{ j \in \text{PRED}_d(k) | Y_{ij} = 1 \right\}$$

T_0^k and T_1^k represent the sets of variables that have already been fixed at 0 and 1 respectively. We also define the current restriction factor associated with the set $\text{PRED}_d(k)$ as:

$$F(T_1^k) = S_i C_{iq} \prod_{j \in T_1^k} \alpha_{ij} .$$

If node d at level k represents the decision $Y_{ij} = 1$, then the net benefit of that node is defined as:

$$B_j^k = F(T_1^k)(1 - \alpha_{ij}) - C_{ji} D_j .$$

Using the above definitions, we are now in the position to describe the ordering and fathoming rules associated with the branch and bound tree.

A Dominance Rule: Y_{ij_1} dominates Y_{ij_2} if in an optimal solution to problem (NLP$_i$), $Y_{ij_1} = 0$ implies $Y_{ij_2} = 0$ and $Y_{ij_2} = 1$ implies $Y_{ij_1} = 1$.

Lemma 1

If $\alpha_{ij_1} < \alpha_{ij_2}$ and $D_{j_1}C_{j_1i} < D_{j_2}C_{j_2i}$, then Y_{ij_1} dominates Y_{ij_2}.

Proof

(By contradiction). Assume that in an optimal solution to problem (NLP$_i$) $Y_{ij_1} = 0$ and $Y_{ij_2} = 1$. Let the optimal value of the objective function be Z^*. It is easy to see that by setting $Y_{ij_1} = 1$ and $Y_{ij_2} = 0$ the resulting value of the objective function, denoted $\overline{Z}$, is such that $\overline{Z} < Z^*$. □

We use the above result to construct the branch and bound tree such that the top levels correspond to variables in dominance order (if it exists); more specifically, let $Y_{ij_1}, \ldots, Y_{ij_m}$ be such that $\alpha_{ij_p} < \alpha_{ij_{P+1}}$ and $D_{j_P}C_{j_Pi} < D_{j_{P+1}}D_{j_{P+1}i}$ for $p = 1, \ldots, m-1$. The variables $Y_{ij_1}, \ldots, Y_{ij_m}$ correspond to levels 1 through m of the tree. During the traversal of the tree, if a noded corresponding to $Y_{ij_P} = 0$ is visited (and not fathomed), only a single path is followed in the sub-tree rooted at note d between levels $P + 1$ and m; that path corresponds to $Y_{ij} = 0$ for $j = j_{P+1}, \ldots, j_m$. The rest of the Y_{ij} variables correspond to levels $m + 1$ through $|T| - 1$; they are ordered based on B_j^k whose value at the root node is: $B_j^0 = S_i C_{iq}(1 - \alpha_{ij}) - C_{ji}D_j$. Variables

with lower B_j^0 are at a lower tree level. The logic underlying this ordering is that fathoming (based on the rule described below) is more likely to occur at the top of the tree when such an ordering is used.

Lemma 2

When node d at level k is visited, and it represents the decision $Y_{ij} = 1$, then if $B_j^k \leq 0$ there exists an optimal solution (given that the variables for levels 1 through $k-1$ are fixed as given by T_1^k and T_0^k), such that $Y_{ij} = 0$.

Proof

Substituting the decision variable values according to T_1^k and T_0^k in problem (NLP_i) produces the following new problem (corresponding to level k of the tree):

Problem (NLP_i^k):

$$\text{Min} \left\{ F(T_1^k) \prod_{j \in (T - T_1^k - T_0^k)} \left(\max \{(1 - Y_{ij}), \alpha_{ij}\} \right) + \sum_{j \in T_1^k} D_j C_{ji} + \sum_{j \in (T - T_1^k - T_0^k)} D_j C_{ji} Y_{ij} \right\}$$

Subject to: $Y_{ij} \in (0, 1),\ j \in T - T_1^k - T_0^k$

Assume that in an optimal solution to problem (NLP_i^k) with an objective value of Z^*, there is a $\tilde{j}$ such that $Y_{i\tilde{j}} = 1$ and $B_{\tilde{j}}^k \leq 0$. Also, let $\hat{T}_1^k = \left\{ j \neq \tilde{j} \mid Y_{ij} \text{ is set to 1 in an} \right.$

optimal solution to problem $(\mathrm{NLP}_i^k)\}$. Substituting $Y_{i\tilde{j}} = 1$ by $Y_{i\tilde{j}} = 0$, the new objective value is Z' and

$$Z' - Z^* = F(T_1^k) \prod_{j \in \hat{T}_1^k} (\alpha_{ij})(1 - \alpha_{i\tilde{j}}) - D_{\tilde{j}} C_{\tilde{j}i}.$$

Now, $B_j^k \leq 0$ implies that $F(T_1^k)(1 - \alpha_{i\tilde{j}}) - C_{\tilde{j}i} D_{\tilde{j}} \leq 0$, and since $0 \leq \prod_{j \in \hat{T}_1^k} (\alpha_{ij}) \leq 1$ it follows that $Z' - Z^* \leq 0$. □

Following Lemma 2, a node d at level k that represents $Y_{ij} = 1$ is fathomed if $B_j^k \leq 0$.

Additional fathoming is done when a value of a lower bound at a node is greater than or equal to the upper bound value generated by Algorithm 1 (described below). We derive the lower bound value as follows. Consider a node d at level k. Let Z_k^* denote the optimal solution to problem (NLP_i) given that the variables corresponding to levels 1 through k are fixed as determined by T_0^k, T_1^k and node d. Also, let j_d identify the variable Y_{ij} corresponding to node d, and I_d be 1 if $Y_{ij_d} = 1$ and 0 otherwise. We define $F(T_1^k)^+ = F(T_1^k) \times \max\{(1 - I_d), \alpha_{ij_d}\}$. The following Lemma defines a lower bound Z_k^L associated with node d at level k.

Lemma 3

Let

$$L = \{j \in \{T - T_1^k - T_0^k - j_d\} | (D_j C_{ji} - F(T_1^k)^+ (1 - \alpha_{ij})) \leq 0\},$$

and

$$Z_k^L = F(T_1^k)^+ + \sum_{j \in T_1^k} D_j C_{ji} + D_{j_d} C_{j_d i} I_d) + \sum_{j \in L} (D_j C_{ji} - F(T_1^k)^+ (1 - \alpha_{ij})).$$

Then, $Z_k^L \le Z_k^*$.

Proof

When node d at level k is fixed to either 0 or 1, the remaining problem is

$$\text{Min} \left\{ F(T_1^k)^+ \prod_{j \in (T - T_1^k - T_0^k - j_d)} \left(\max \{(1 - Y_{ij}),\ \alpha_{ij}\}\right) + \sum_{j \in T_1^k} D_j C_{ji} + D_{j_d} C_{j_d i} I_d \right.$$

$$\left. + \sum_{j \in (T - T_1^k - T_0^k - j_d)} D_j C_{ji} Y_{ij} \right\}$$

Subject to: $\quad Y_{ij} \in (0,\ 1),\ \ j \in T - T_1^k - T_0^k - j_d$

The basis of the lower bound is the following inequality (recall that $0 \le \alpha_{ij} \le 1$):

$$\prod_j \alpha_{ij} \ge 1 - \sum_j (1 - \alpha_{ij}) .$$

Consequently, for any combination of Y_{ij} values in the product term of the above objective function, the following inequality holds:

$$\prod_{j \in (T - T_1^k - T_0^k - j_d)} \left(\max \{(1 - Y_{ij}),\ \alpha_{ij}\}\right) \ge 1 - \sum_{j \in (T - T_1^k - T_0^k - j_d)} (1 - \alpha_{ij}) Y_{ij} .$$

Replacing the left-hand side by the right-hand side of this inequality in the objective function and solving the revised problem optimally yields a lower bound. In the revised problem $Y_{ij} = 1$ if $j \in L$ and 0 otherwise. Consequently, the resulting value of Z_k^L is as given in the Lemma statement. □

An Upper Bound Procedure

The upper bound procedure is an ADD procedure (stated as Algorithm 1 below) which uses the logic introduced as part of the branch and bound procedure. Since we need only denote current values in the description of the procedure, the notation T_0^k, T_1^k, and $F(T_1^k)$ is replaced by T_0^i, T_1^i, and $F(T_1^i)$. $T_0^i = \{j \in T \mid Y_{ij} = 0\}$ denotes the current set of variables set to 0 at any stage of Algorithm 1 when applied to problem (NPL$_i$). Similarly, $T_1^i = \{j \in T \mid Y_{ij} = 1\}$, and the current restriction factor is defined as $F(T_1^i) = S_i C_{iq} \prod_{j \in T_1^i} \alpha_{ij}$. At each iteration of Algorithm 1, the next semi-join to be chosen (i.e., added to T_1^i) is the one which maximizes the net benefit relative to the current restriction factor. The net benefit for $R_i \leftarrow R_j$ is given by $B_j = F(T_1^i)(1 - \alpha_{ij}) - C_{ji}D_j$.

Algorithm 1 resorts B_j in descending order to take advantage of the fact that once B_j becomes negative Y_{ij} can be set to zero. Algorithm 1 terminates when either $|T - T_0^i - T_1^i| = 0$ (Step 6c) or all the elements in $T - T_0^i - T_1^i$ have negative net benefit (Step 4b). When the algorithm terminates, the set T_1^i represents the set of semi-joins selected to restrict relation R_i.

After the branch and bound procedure and/or Algorithm 1 are run $|T|$ times (once for each relation R_i), the global set of selected semi-joins is given by the union of the sets T_1^i, $i = 1, \ldots, |T|$. Those semi-joins are sequenced in Section 4.2.

Algorithm 1

Step 1: (Initialization) $T_0^i = T_1^i = \emptyset, \quad F(T_1^i) = S_i C_{iq}$

Step 2: Calculate $B_j = F(T_1^i)(1 - \alpha_{ij}) - C_{ji}D_j$ for all $j \in T - T_0^i - T_1^i$.

Let $n = \left| T - T_0^i - T_1^i \right|$.

Step 3: Sort B_j in descending order. Let (j) denote the original index of element j in the sorted sequence.

Step 4: If $B_1 > 0$ go to Step 5; else do:

a. $Y_{(j)} = 0, j = 1, \ldots, n.$

b. Go to Step 7.

Step 5: a. $Y_{(1)} = 1$

b. $F(T_1^i) = F(T_1^i)\alpha_{i(1)}$

c. $T_1^i = T_1^i + \{(1)\}$

Step 6: Let $\tilde{j}$ be the first index $j = 2, \ldots, n$ such that $B_j \leq 0$.

Do a. $Y_{(j)} = 0, \quad j = \tilde{j}, \ldots, n$

b. $T_0^i = T_0^i + \{(\tilde{j}), (\tilde{j} + 1), \ldots, (n)\}$

c. If $\left| T - T_0^i - T_1^i \right| > 0$, go to Step 2.

Step 7: Stop.

4.2 A Sequencing Algorithm

Algorithm 1 generates a set of selected semi-joins. The sequencing of those semi-joins has a significant impact on the resulting communication costs. In this section a sequencing algorithm is proposed whose output is the order of execution of the selected semi-joins. That order of execution can be represented as a directed acyclic flowgraph [BERN81]. Algorithm 2 below orders the execution of the semi-joins based on their net benefit. The net benefit is defined to be the benefit of the semi-join minus its cost. The benefit of a semi-join in the context of sequencing is not the reduction of the restricted relation (because that reduction is sequence-independent for a given set of semi-joins), but rather a lower cost of subsequent semi-joins due to size reduction of the join attribute of the restricted relation. For example, if semi-joins $R_i \leftarrow R_j$ and $R_k \leftarrow R_i$ are selected (i.e., $Y_{ij} = Y_{ki} = 1$), then the benefit of $R_i \leftarrow R_j$ (assuming that it is the first to be executed) is in reducing the cost of $R_k \leftarrow R_i$; the net benefit equals to $D_i C_{ik}(1 - \alpha_{ij}) - D_j C_{ji}$. At every iteration Algorithm 2 adds the semi-join with maximum current net benefit to the sequence, modifies the size of the restricted relation's join attribute and selectivity factors, updates the net benefits, and proceeds with the next iteration. SCOST is the cost of the semi-joins and is updated in Step 6 after a new semi-join is added to the sequence SQ (which is initially empty) in Step 5.

Algorithm 2

Step 1: (Initialization) SCOST = 0, $SQ = \emptyset$, $D'_i = D_i$, $\alpha'_{ij} = \alpha_{ij}$, for all $_{ij} \in T$.

Step 2: For every $i \in T$ and $j \in T_1^i$ calculate:

$$\text{Net Benefit} = \sum_{(k \mid Y_{ki}=1)} (D'_i C_{ik}(1 - \alpha'_{ij})) - D'_j C_{ji}$$

Step 3: Let $\tilde{i}$ and $\tilde{j}$ be the subscripts for which net benefit is maximized.

Step 4: Update D_i' and α_{ij}' as described in [BERN81].

Step 5: Remove $\tilde{j}$ from $T_{\tilde{i}}^{i}$ and add $(\tilde{i}, \tilde{j})$ to SQ.

Step 6: SCOST = SCOST + $D_{\tilde{j}} C_{\tilde{j}\,\tilde{i}}$

Step 7: If $\underset{i}{U}\{T_{\tilde{i}}^{i}\}$ is not empty, go to Step 2, else stop.

4.3 Extensions of the Model

The procedures described in Sections 4.1 and 4.2 are for a single join attribute. The case of multiple join attributes is handled in the following way. α_{ij} is changed to α_{ijk} and D_i is changed to D_{ik}, where k denotes the attribute number. Additional index set, denoted by A, is used to subscript the join attributes, and problem (NLP_i) is changed to:

$$\text{Min}\left\{ S_i C_{iq} \prod_{j \in T} \prod_{k \in A} \left(\max\{(1 - Y_{ijk}),\ \alpha_{ijk}\}\right) + \sum_{j \in T} \sum_{k \in A} D_{jk} C_{ji} Y_{ijk} \right\}$$

Subject to: $Y_{ijk} \in \{0,\ 1\},\quad j \in T,\quad k \in A$

where:

$$Y_{ijk} = \begin{cases} 1, & \text{if } R_i \text{ is semi-joined by } R_j \text{ on join attribute } k \\ 0, & \text{otherwise.} \end{cases}$$

This problem is exactly the one described in Section 4.1, though with a greater number of variables (renumbering the subscripts is required to get the original form of problem (NLP_i)). The absence of attribute $k \in A$ from relation i is taken care of by setting $D_{ik} = 0$ and $\alpha_{ijk} = 1$, for all $j \in T$.

The global heuristic can be improved by following the sequencing algorithm with any query optimization algorithm that will consider only the sets T_0^i, $i = 1, \ldots, |T|$. This may add more beneficial semi-joins to the sequence.

5. Computational Experiments

A set of computational experiments was carried out to evaluate the performance of both the branch and bound procedure and Algorithm 1. Problems (NLP_i) were solved for 200 cases in all of which Algorithm 1 generated the optimal solution. In order to evaluate the semi-join strategy generated by the global heuristic, it was compared with an enhanced version of the SDD-1 query optimization algorithm (Algorithm OPT from [BERN81]). The enhancement to Algorithm OPT is based on [YU85], and we will refer to it as Algorithm EOPT. Since algorithm EOPT assumes a uniform transmission cost rate, $C_{ij} = C = 5$, for all i, $j \in T$ was used in the experiments. The results of the experiments are shown in Table 1. The figures in the table represent the average (of eight data sets) ratio of algorithm EOPT's cost to the global heuristic's cost. The number of relations and join attributes was 5 and 4 for two cases, 4 and 3 for two cases, 4 and 2 for two cases, and 3 and 2 for two cases. The extended version of the global heuristic (see Section 4.3) was used in the experiments.

The most important parameters in the distributed join optimization problem are α_{ij}, D_i, and S_i. The smaller the selectivity factors, the more beneficial the semi-joins are likely to be. Smaller D_i makes the semi-join cost lower. Since the benefit of a semi-join $R_i \leftarrow R_j$ (for a given α_{ij} and D_j) is dependent on the size of R_i, that is, the larger S_i the more beneficial the semi-join, we use the parameter β to express the relative weight of the join attributes and relation sizes. When α and β assume large values, the global heuristic achieves roughly the same solution as EOPT. This is expected since not many semi-joins are beneficial, and the global heuristic does not have the "opportunity" to discover beneficial semi-joins. For very small values of α and β (e.g., α = [.01, .02] and β = [.01, .05]) we would expect the performance of the algorithms to be similar because most semi-joins are clearly beneficial. We would expect the global heuristic to exhibit the best relative performance when β values are relatively small (in that case, the sequence-independent problem is a very good model of the actual problem) and α values are not too small and not too large (in that case, identifying the complete set of beneficial semi-joins is difficult). Of course, determining what is "too small" or "too large" is the subject of a simulation study in the context of a particular case.

In general, the results presented in Table 1 validate the above conjectures, and it is evident that the global heuristic exhibits a promising performance. It should be noted that the join optimization problem is more difficult when the inter-site communication cost rates are not equal. In that case it is expected that the global heuristic will perform better (relative to local heuristics) than in the case of equal inter-site communication cost rates. In order to compare the global heuristic with an established local heuristic we used equal communication cost rates in the experiments. Finally, in all the cases that were run, the optimal solution was achieved in less than one second of CPU time on an IBM3090.

β Range \ α Range	[.01, .02]	[.01, .04]	[.01, .08]	[.01, .16]	[.01, .32]
[.01, .05]	1.05	1.43	1.29	1.48	0.99
[.01, .10]	1.10	1.33	1.20	1.15	1.05
[.01, .20]	1.16	1.25	1.17	1.02	1.02
[.01, .40]	1.26	1.16	0.93	1.13	1.00
[.01, .60]	1.10	1.10	0.98	1.00	1.00

Notes:
1. α Range designates the range of the uniform distribution from which the selectivity factors α_{ij} were drawn.
2. β Range designates the range of the uniform distribution from which $\beta_i = \frac{D_i}{S_i}$ were drawn.

Table 1: Cost Ratios (Algorithm EOPT to Global Heuristic)

6. Conclusions

This paper has dealt with a class of heuristic procedures that attempts to minimize the communication costs incurred by the distributed processing of join queries in a relational DBMS. Those heuristics use the semi-join strategy as the mechanism for communication cost reduction. The paper has classified heuristic procedures into two types: local heuristics and global heuristics.

A global heuristic has been proposed in this paper; it consists of three stages: 1) selecting a set of semi-joins by solving a mathematical model of a simplified version of the problem;

2) sequencing the execution of those semi-joins; and 3) looking for beneficial semi-joins not selected in stage 1. The simplified model used in stage 1 is achieved by ignoring the effect of the semi-join sequence on the cost of the semi-joins. That model, however, considers the global effect of the selected set of semi-joins on the size reduction of the relations.

For stage 1 of the problem, we have developed a branch and bound procedure. The procedure employs fathoming rules and a lower bound as was stated in Lemmas 1 through 3. Algorithm 1 was used to produce an upperbound for the branch and bound procedure. It turned out, however, that for all the cases that were run, Algorithm 1 produced the optimal solution. Algorithm 1 is not proven or disproven to be optimal, and it will be a challenging exercise to ascertain one of the two possibilities. Given the performance of Algorithm 1, and the fast processing time required for some ad-hoc online queries, it is recommended that for such cases this algorithm rather than the branch and bound procedure is used. For small number of relations, and for pre-compiled queries, it is appropriate to use the branch and bound procedure. Since the overall solution is not necessarily optimal, and Algorithm 1 generates feasible solutions, it can be used regardless of whether or not it is optimal for problem (NLP_i).

The paper has presented the results of computational experiments. Those results exhibit a promising performance of the global heuristic and validate some conjectures about the global heuristic. Current and future research concentrate on further applications of the global heuristic. That is, we find the approach used in this research to be of significance to other distributed query optimization problems.

Acknowledgement

I would like to thank Mr. Steve Cosares of the University of California at Berkeley for programming the algorithms and running the computational experiments.

FOOTNOTES

1. For more complicated strategies not discussed in this paper, a third decision is the location of semi-join execution (see [SEGE86] for a description of those strategies).

2. The consequences of non-optimality are discussed in Section 6.

REFERENCES

APER83 Apers P. M. G., Hevner A. R., Yao S. B., "Optimization Algorithms for Distributed Queries," *IEEE Transactions on Software Engineering* Vol. SE-9, 1 (Jan. 1983), pp. 57-68.

BERN81 Bernstein A. P., et al., "Query Processing in a System for Distributed Databases (SDD-1)," ACM TODS, 6, 4 (Dec. 1981), pp. 602-625.

BERN81b Bernstein A. P., Chiu D. W., "Using Semi-Joins to Solve Relational Queries," *Journal of the ACM*, 28, 1 (Jan. 1981), pp. 25-40.

CHAN82 Chang Jo-Mei, "A Heuristic Approach to Distributed Query Processing," Proceedings of the 8th International Conference on Very Large Data Bases, pp. 54-61, 1982.

CODD70 Codd E. F., "A Relational Model for Large Shared Data Banks," Communications of the ACM, 13, 6 (June 1970), pp. 909-917.

DANI82 Daniels D., "Query Compilation in a Distributed Database System," IBM Research Report, RJ3423 (March 1982).

DANI82b Daniels D., et al., "An Introduction to Distributed Query Compilation in R^*," IBM Research Report, RJ3497 (June 1982).

DOWD82 Dowdy L. W., Foster D. V., "Comparative Models of the File Assignment Problem," *Computing Surveys*, Vol. 14, 2 (June 1982), pp. 287-313.

EPST79 Epstein R. S., Stonebraker M., Wong E., "Distributed Query Processing in a Relational Database System," Proceedings ACM-SIGMOD, May 1979, pp. 169-180.

GAVI86 Gavish B., Segev A., "Set Query Optimization in Distributed Database Systems," *ACM Transactions on Database Systems*, 11, 3 (1986), pp. 265-293.

HAER83 Haerder T., Reuter A., "Principles of Transaction-Oriented Database Recovery," *Computing Surveys*, Vol. 15, 4 (Dec. 1983), pp. 287-317.

HEVN79 Hevner A. R., Yao S. B., "Query Processing in Distributed Database Systems," *IEEE Transactions on Software Engineering*, Vol. SE-5, 3 (May 1979), pp. 177-187.

HORO78 Horowitz E., Sahni S., *Fundamentals of Computer Algorithms*, Computer Science Press, Inc., 1978.

JARK84 Jarke M., Koch J., "Query Optimization in Database Systems," *Computing Surveys*, Vol. 16, 2 (June 1984), pp. 111-152.

PAIK79 Paik I-S., Delobel C., "A Strategy for Optimizing the Distributed Query Processing," Proceedings of the First International Conference on Distributed Computing Systems, Huntsville, Alabama, October 1979.

ROTH77 Rothnie J. B., Goodman N., "A Survey of R&D in Distributed Database Management," Proceedings of the International Conference on Very Large Databases, Tokyo, October 1977, pp. 48-62.

SEGE84 Segev A., "Optimizing Fragmented 2-Way Joins," Proceedings of the 4th International Conference on Distributed Computing Systems, San Francisco, May 1984, pp. 378-388.

SEGE86 Segev A., "Optimization of Join Operations in Horizontally Partitioned Database Systems," *ACM Transactions on Database Systems*, 11, 1 (1986), pp. 48-80.

SEGE88 Segev A., "Strategies for Distributed Query Optimization," *Information Sciences*, forthcoming.

SELI80 Selinger P. G., Adiba M. E., "Access Path Selection in Distributed Data Base Management Systems," Proceedings of the International Conference on Databases, July 1980, pp. 204-215.

WONG77 Wong E., "Retrieving Dispersed Data from SEE-1: A System for Distributed Databases," Proceedings of the Third Berkeley Workshop on Distributed Data Management and Computer Networks, May 1977, pp. 217-235.

YU83 Yu C. T., Chang C. C., "On the Design of a Query Processing Strategy in a Distributed Database Environment," Proceedings ACM SIGMOD, May 1983, pp. 30-39.

YU84 Yu C. T., Chang C. C., "Distributed Query Processing," *ACM Computing Surveys*, Vol. 16, 4 (December 1984), pp. 399-433.

YU85 Yu C. T., "Distributed Database Query Processing," in *Query Processing in Database Systems*, Edited by W. Kim, D. S. Reiner, and D. S. Batory, Springer-Verlag, pp. 48-61.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
1 CYCLOTRON ROAD
BERKELEY, CALIFORNIA 94720