Optimization of Join Operations in Horizontally Partitioned Database Systems

ARIE SEGEV
The University of California

This paper analyzes the problem of joining two horizontally partitioned relations in a distributed database system. Two types of semijoin strategies are introduced, local and remote. Local semijoins are performed at the site of the restricted relation (or fragment), and remote semijoins can be performed at an arbitrary site. A mathematical model of a semijoin strategy for the case of remote semijoins is developed, and lower bounding and heuristic procedures are proposed. The results of computational experiments are reported. The experiments include an analysis of the heuristics' performance relative to the lower bounds, sensitivity analysis, and error analysis. These results reveal a good performance of the heuristic procedures, and demonstrate the benefit of using semijoin operations to reduce the size of fragments prior to their transmission. The algorithms for the case of remote semijoins were found to be superior to the algorithms for the case of local semijoins. In addition, we found that the estimation accuracy of the selectivity factors has a significant effect on the incurred communication cost.

Categories and Subject Descriptors: C.2.4 **[Computer-Communication Networks]**: Distributed Systems—*distributed databases*; G.2.1 **[Discrete Mathematics]**: Combinatorics—*combinatorial algorithms*; G.2.2 **[Discrete Mathematics]**: Graph Theory—*trees*; H.2.3 **[Database Management]**: Languages—*query languages*; H.2.4 **[Database Management]**: Systems—*distributed systems*; *query processing*

General Terms: Algorithms, Languages, Theory

Additional Key Words and Phrases: Database, Lagrangian relaxation, lower bounds, processing, query, semijoin, subgradient, systems

1. INTRODUCTION

One of the important problems in the design and implementation of distributed database management systems (DDBMS) is the efficient processing of queries. Models that attempt to characterize and solve this problem have been developed by Apers et al. [2], Hevner and Yao [22], Paik and Delobel [25], Epstein et al. [11], Wong [36], Yu and Chang [37], and others. The results derived by these models differ substantially and depend on the assumptions made, such as the objective function and parameters used by the model. Most of the models deal with static optimization of single queries (i.e., no competition for resources by multiple queries is addressed), and assume equal unit transmission cost (or time)

Author's address: University of California, Berkeley, Schools of Business Administration, 350 Barrows Hall, Berkeley, CA 94720.

for every pair of nodes. We also deal with static optimization of single queries, but allow for variable transmission costs.

The subject of horizontal partitioning (fragmentation) has been discussed in several papers. In this type of partitioning, a relation is fragmented into several sets of tuples (usually disjoint sets), each stored at a different site. Wong [36] asserts that SDD-1 supports horizontal partitioning, but it is external to his model (i.e., the input to the model is a specific realization of the relations, nonpartitioned and with no duplicates). In a later version of SDD-1 [19], fragmentation is allowed in the context of increased parallelism; that is, fragmentation is used as part of the processing strategy to minimize response time. This paper assumes that fragmentation is predetermined by some data allocation strategy, and the objective function is to minimize the incurred communication cost. Distributed INGRES [33] provides for fragmentation, but the query optimization algorithm limits fragmentation to a single relation; and, as in SDD-1, it is also in the context of increased parallelism. System R* [34] supports fragmentation, but each fragment is viewed as a separate relation. A rigorous treatment of fragmentation is given by Pelagatti and Schreiber [27], but the resulting mathematical formulation applies only to the case where fragments of one relation have to be assembled at a single site. Yu and Chang [37] propose an algorithm for a fragmented database that is based on the premise that there is no benefit in reducing a fragment by another fragment. This paper shows that such a reduction may be beneficial if remote semijoins (remote semijoins are explained in Section 2) are used. Gavish and Segev [15] propose an optimization model for queries involving set operations in fragmented database systems, and the problem of joining two fragmented relations is analyzed in [30], but only for the case of local semijoins, while this paper tackles the more general case of remote semijoins.

Horizontal partitioning of databases is an important case, since it is a *natural* partitioning in many real-world environments. Examples include insurance, banking, credit cards, air-line, hotel, and car rental industries, libraries and patient medical systems, to name only a few. The aforementioned applications have two main attributes that make horizonal partitioning beneficial. The first is *locality of reference* [26]: most of the transactions originating at a particular site need only the data associated with that site. For example, most of the queries or transactions relating to an individual's bank account, insurance policy, or medical records originate from his local branch, agent, or service. The second attribute is the *large size* of those relations (e.g., millions of books in libraries, millions of insurance policies, and millions of credit cards). In these cases, horizontal partitioning and a good allocation of fragments to sites provide shorter response time and lower processing costs for local queries (queries that refer to local data). An additional advantage of fragmentation is a consistency, in many cases, with the organizational structure.

This paper analyzes the problem of optimizing 2-way joins in horizontally partitioned DDBMS. Two types of semijoin strategies are analyzed in Section 2. In Section 3 the choice of an objective function and basic assumptions are discussed, and the 2-way join problem is defined. A mathematical model is developed in Section 4, lower bounding procedures are discussed in Section 5, and heuristic procedures are proposed in Section 6. An analysis of various

R1	PART#	COLOR	WEIGHT
	100	BLUE	20
	200	BLACK	15
	300	BLUE	18
	400	RED	12

R2	PART#	DEPT#	QTY
	100	10	1000
	100	15	1500
	300	10	500
	300	12	800

R1 SEMIJOINED BY R2

⇓

R1	PART#	COLOR	WEIGHT
	100	BLUE	20
	300	BLUE	18

Fig. 1. A semijoin operation.

optimization procedures (including sensitivity and error analysis) is presented in Section 7. Section 8 concludes the paper with a summary and future research directions.

2. LOCAL AND REMOTE SEMIJOINS

It is assumed in this paper that the reader is familiar with the basic relational terminology and operators (see [6, 10] for example).

The semijoin operation has been introduced as a useful mechanism to reduce the amount of transmitted data when processing queries in distributed databases [1, 2, 3, 36]. Theoretical work on the semijoin operation can be found in [4]. A semijoin operation between relation R1 and relation R2 restricts R1 by values that appear in R2's join attribute. Figure 1 shows two sample relations R1 and R2, and the effect of semijoining R1 by R2 over the join attributes R1.PART# and R2.PART#. Note that the semijoin is an asymmetric operator; semijoining R2 by R1 has no effect on the size of R2 in the example of Figure 1. If relation R1 is semijoined by relation R2, we refer to R1 as the *restricted relation* and to R2 as the *restricting relation.*

Given that relation R1 is to be semijoined by relation R2 over join attributes R1.a and R2.b, we distinguish between two modes of executing a semijoin operation. In the first mode, referred to as a *local semijoin* (e.g., [2, 3, 36]), R2.b is transmitted to R1's site and joined with R1. The second mode of executing a semijoin operation, referred to as a *remote semijoin*, allows R1.a to be restricted by R2.b at a remote site and then transmitted back to R1's site for a restriction of R1. The two execution modes and the possible advantage of a remote semijoin are illustrated by the following example.

Example. Given that

- Relation R1 is to be semijoined by relation R2.
- R1 and R2 are stored at sites 1 and 2, respectively.
- The join attributes are R1.a and R2.b.
- Size(R1.a) = 5000 bits (duplicates are eliminated).
- Size(R2.b) = 20,000 bits (duplicates are eliminated).
- Communication cost rate = 10^{-4}\$/bit.
- Selectivity factor = 0.1 (i.e., the size of R1 after the semijoin is size(R1) × 0.1, and the size of R1.a after a restriction by R2.b is size(R1.a) × 0.1).

(a) *Local semijoin case*
The following steps are taken:
1. Project R2 on the join attribute R2.b (eliminating duplicates).
2. Transmit R2.b from site 2 to site 1.
3. Join R1 and R2.b.

Communication cost = size(R2.b) × cost rate = $20{,}000 \times 10^{-4} = \$2$.

(b) *Remote semijoin case*
Selecting site 2 to be the remote site, a *remote semijoin* of R1 implies the following steps:
1. Project R1 and R2 on their join attributes (eliminating duplicates).
2. Transmit R1.a from site 1 to site 2.
3. Restrict R1.a by R2.b and denote the result as $\overline{R1.a}$.
4. Transmit $\overline{R1.a}$ from site 2 to site 1.
5. Restrict R1 by $\overline{R1.a}$.

Communication cost = size(R1.a) × cost rate + size($\overline{R1.a}$) × cost rate
$= 5000 \times 10^{-4} + 500 \times 10^{-4} = \$0.55$.

Notes

(1) Though the remote semijoin operation may incur higher processing cost, it can significantly reduce the communication cost.
(2) If the remote site was chosen to be an arbitrary third site, R1.a and R2.b would have to be transmitted to that site. A choice of a third site may be beneficial when the communication cost rates differ across sites, or a third relation must also be semijoined by one of the two relations. The mathematical model developed in Section 4 allows for an arbitrary remote site.
(3) The local semijoin is a special case of the remote semijoin; that is, the site of R1 (assuming that R1 is the relation to be restricted) is chosen to be the remote site.

3. PROBLEM DEFINITION AND BASIC ASSUMPTIONS

3.1 The Choice of an Objective Function

Common objective functions adopted by query optimization researchers are minimization of total cost, total amount of transmitted data, and response time. The total cost of processing a query consists of two main components. The first component is processing cost and the second is communication cost. Note that if the cost of transmitting a data unit between two sites is a constant for every pair of sites, then minimizing total communication costs is equivalent to minimizing the total amount of transmitted data (for models that use this performance measure see [3, 11, 36]).

The performance measure adopted in this paper is the minimization of the communication costs incurred by processing a query. Minimization of communication costs is a valid objective for systems that use the services of a value-added network (e.g., Tymnet and Telenet). The usage of value-added networks has increased significantly after the divestiture of AT&T [24], their users are paying "real" money for the transmitted data, and the tariffs need not be the same for all pairs of sites. Minimizing communication costs is also a valid objective for systems with a highly congested or slow communication network.

Site 1

R1	PART#	COLOR	WEIGHT
	100	BLUE	20
	200	BLACK	15
	300	BLUE	18
	400	RED	12

Site 2

$R2_1$	PART#	DEPT#	QTY
	100	10	1000
	100	15	1500

Site 3

$R2_2$	PART#	DEPT#	QTY
	300	10	500
	300	12	800

Fig. 2. A fragmented database.

In this case, one may choose either a constant unit transmission cost rate, or vary it across different pairs of sites, thus reflecting different load levels on the communication links. The choice of an objective function may also depend on the time of day or the type of query.

The query optimization algorithms introduced in this paper (and in general any algorithm to minimize communication costs) consist of determining a transmission plan and of selecting sites for executing the relational operations. The resulting local processing still has to be optimized by any of the centralized query optimization algorithms (e.g., [35]). The result of local optimization does not necessarily imply a global optimization of processing costs, but rather the optimum subject to a given transmission plan and assignment of operations to sites.

3.2 The 2-Way Join Problem

The 2-way join optimization problem is defined as the problem of joining two fragmented relations such that the resulting communication costs will be minimized. The procedures to perform a 2-way join on nonfragmented relations are described in [8].

The general problem of joining fragmented relations is discussed in [27], and a condition on the availability of the join operands is introduced. The condition states that at least one relation must be *completely duplicated* with respect to the other relation; that is, at least one relation must be assembled at each of the join sites (there might be multiple join sites if one or more of the relations is fragmented). The reason for requiring complete duplication when joining two relations follows directly from the join definition that implies that no tuple of a relation can be discarded before it is compared with *every* tuple of the second relation. This condition, which is adopted by query optimization algorithms (e.g., [11]) might require excessive data transmissions and is unnecessary, as will be shown later in this section.

To illustrate the concept of complete duplication, consider Figure 2. The figure contains the same relations as Figure 1, but relation R2 has been partitioned into two fragments, $R2_1$ and $R2_2$, which are stored at site 2 and site 3, respectively. Suppose that as part of a strategy to join R1 and R2 over PART#, R1 is to be semijoined by R2. By the definition of the semijoin operation (the arguments are similar for a join operation), a semijoin between R1 and $R2_1$ is useless unless accompanied by a semijoin between R1 and $R2_2$.

The complete duplication requirement of [27] implies that R1, $R2_1$.PART#, and $R2_2$.PART# must be assembled at the same site prior to the semijoin operation. It is argued here that this condition is a restrictive one and that *logical duplication* is sufficient. Logical duplication implies that in order to perform a semijoin operation, each tuple of the restricted fragment (or relation) must be compared with every tuple of the restricting relation and the results of those comparisons unioned. In the example of Figure 2, if R1 is to be semijoined by R2, it can be done by performing two semijoins, one between R1 and $R2_1$ and the other between R1 and $R2_2$. Note, however, that the resulting tuples of the two semijoin operations have to be unioned. Moreover, *every* fragment of the restricting relation must participate in a semijoin of the restricted relation (or fragment); otherwise no tuple of the restricted relation (or fragment) can be eliminated.

The concept of logical duplication has been introduced in order to enable remote semijoin operations between fragments. A remote semijoin on PART# to restrict relation R1 of Figure 2 may involve the following steps:

(1) Project R1 on PART#, resulting in R1.PART# = {100, 200, 300, 400}.
(2) Send copies of R1.PART# to sites 2 and 3.
(3) Restrict R1.PART# at site 2, resulting in $\overline{R1}$.PART# = {100}.
(4) Restrict R1.PART# at site 3, resulting in $\overline{R1}$.PART# = {300}.
(5) Send $\overline{R1}$.PART# and $\overline{R1}$.PART# back to site 1 and union the two sets, resulting in $\hat{R}1$.PART# = {100, 300}.
(6) Restrict R1 by $\hat{R}1$.PART#, resulting in

R1	PART#	COLOR	WEIGHT
	100	BLUE	20
	300	BLUE	18

Note that R1 in this example might have been a fragment of some relation (other than R2), and the example would have looked exactly the same.

3.3 Basic Assumptions

This paper analyzes the problem of joining two fragmented relations using remote semijoins, such that the resulting communication cost will be minimized. A mathematical analysis of the case of local semijoins and a proof that the problem is NP-complete can be found in [30]. The following assumptions underlie the models developed in this paper. These assumptions are made in order to simplify the model, and the implications of their removal are discussed in Section 8.

Assumption 1. There is no duplication of relations and/or fragments.

Assumption 2. The join operation that follows the execution of the selected semijoins is performed at the query site (this assumption is not restrictive for the case of equal transmission cost rates for every pair of sites).

Assumption 3. The join attribute of a fragment can be projected only at the original site of the fragment (the other option is to send the fragment to another node before restriction, but the benefits of such a strategy are doubtful, and it would complicate the model significantly).

Assumption 4. The sets of tuples resulting from distinct semijoins on fragment i are disjoint. This assumption is not essential to the model, but it simplifies its exposition. We discuss in Section 8 how this assumption can be removed.

4. A MODEL OF REMOTE SEMIJOINS

4.1 Notation

Let the two relations of the 2-way join problem be R1 and R2. The following notation will be used henceforth:

S1 =: The set of site numbers among which relation R1 is fragmented.
S2 =: The set of site numbers among which relation R2 is fragmented.
T =: S1 ∪ S2

$$\Gamma_i = \begin{cases} S2, & \text{if } i \in S1 \\ S1, & \text{if } i \in S2 \\ \text{undefined}, & \text{otherwise} \end{cases}$$

(We use this notation to conveniently refer to the following: "if i is a fragment of one relation, then Γ_i is the set of fragments of the other relation.")

q =: The query site number.
C_{ij} =: The transmission cost rate between site i and site j.
R_i =: The fragment of relation R, stored at site i.
F_i =: Size of the fragment stored at site i.
D_i =: Size of the projection of fragment i on the join attribute.
$R1_i \leftarrow R2_j$ =: A semijoin between fragments $R1_i$ and $R2_j$, where $R1_i$ is the fragment to be restricted.
α_{ij} =: The selectivity factor for $R1_i \leftarrow R2_j$ (i.e., the size of the result of $R1_i \leftarrow R2_j$ is $F_i\alpha_{ij}$).

The plus and minus signs in a notation of the type "$k \in S1 \pm i$" is used to represent an element k of the union of set $S1$ and the singleton i, and the removal of element i from the set $S1$, respectively. If S is a set, $|S|$ is its cardinality.

Note. For notational convenience, we use the same number for a site, the fragment stored at that site, and the join attribute of that fragment. For example, a fragment that is stored at site 5 will be assigned the number 5; so will its join attribute. (Additional notation and definitions will be introduced as necessary.)

4.2 Transmissions Associated with Local and Remote Semijoins

In the model developed in this paper, we consider the transmission of two forms of join attributes. Unrestricted join attributes are transmitted to other nodes, either to restrict another join attribute or to be restricted themselves. Restricted join attributes are sent back to their originating site. Figure 3 represents an example of semijoin transmissions. Only the transmissions of unrestricted join attributes are shown, since the transmissions of restricted join attributes are implied.

We distinguish between three possible transfer types of unrestricted join attributes; they are illustrated in Figure 3. It is assumed that fragments 1 and 2 belong to R1; fragments 3, 4, and 5 belong to R2; and that both R1 and R2 have to be restricted by semijoin operations. The assumption that R1 and R2 have to be restricted, and the transmissions shown in Figure 3, are not necessarily optimal, and are only made to illustrate the various transfer types.

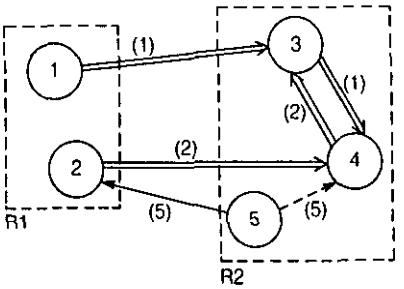

Fig. 3. Transfers of join attributes.

The transfer types can be classified as follows:

(1) Join attribute k is transmitted from site i to site j for a local semijoin ($i \xrightarrow{(k)} j$ in Figure 3), that is, join attribute k is used to restrict fragment j. Note that k is not necessarily equal to i(i is equal to k in Figure 3); this would be the case if join attribute k were transmitted from site k for a semijoin at site i and then further transmitted from site i to site j.

(2) Join attribute k is transmitted from site i to site j for one or more remote semijoins ($i \overset{(k)}{\dashrightarrow} j$ in the figure). In this case join attribute k is restricted at site j by one or more join attributes (possibly different than j). For each remote semijoin, the restricted join attribute k is transmitted back to site k.

(3) Join attribute k is transmitted from site i to site j for a local semijoin and one or more remote semijoins ($i \overset{(k)}{\Longrightarrow} j$) in the figure). This case is the combination of the first two cases.

Figure 3 is a graphical representation of the following semijoin operations:

Semijoin	Site	Type
$R1_1 \leftarrow R2_3$	3	remote
$R2_3 \leftarrow R1_1$	3	local
$R2_3 \leftarrow R1_2$	3	local
$R1_2 \leftarrow R2_3$	3	remote
$R2_4 \leftarrow R1_1$	4	local
$R1_1 \leftarrow R2_4$	4	remote
$R2_4 \leftarrow R1_2$	4	local
$R1_2 \leftarrow R2_4$	4	remote
$R2_5 \leftarrow R1_1$	4	remote
$R1_1 \leftarrow R2_5$	4	remote
$R2_5 \leftarrow R1_4$	4	remote
$R1_2 \leftarrow R2_5$	2	local

Several points should be noted:

(1) Only transfers of unrestricted join attributes are shown in Figure 3. Other types of data flow include the transmission of restricted join attributes back to

their original sites and transfers of restricted fragments to the query site. These data flows are taken into account by the objective function.

(2) A remote semijoin may take place at site k even though it does not involve fragment k at that site. An example of that is the restriction of join attribute 5 by join attributes 1 and 2 at site 4 (see Figure 3).

(3) The transfers of an unrestricted join attribute constitute a directed spanning tree rooted at the original site of the join attribute (i.e., all arcs labeled by the same join attribute number constitute a spanning tree).

4.3 A Mathematical Formulation

A mathematical programming formulation of the model is given as problem P_J (Problem Join) below. The decision variables used in the formulation are defined as follows:

$$X_i = \begin{cases} 1, & \text{if fragment } i \text{ is restricted by a semijoin} \\ 0, & \text{otherwise;} \end{cases}$$

$$Y_{ijk} = \begin{cases} 1, & \text{if a semijoin } R1_i \leftarrow R2_j, \quad i \in S1 \text{ and } j \in S2, \\ & \quad \text{takes place at site } k \\ 0, & \text{otherwise;} \end{cases}$$

$$W_{ijk} = \begin{cases} 1, & \text{if join attribute } i \text{ is transmitted from site } j \text{ to site } k \\ 0, & \text{otherwise;} \end{cases}$$

g_{ijk} = amount of flow on the arc represented by $W_{ijk} = 1$.

The g_{ijk} variables are used to prevent cycles (by using flow constraints) in the directed graph representing the transmissions of an unrestricted join attribute. Intuitively, the variable Y_{ijk} represents a node k where fragment i is to be semijoined by fragment j, while the variable W_{ijk} represents an arc into node k from node j for the join attribute originating at node i.

Problem P_J.

$$\text{Min}\left\{\sum_{i\in T} F_i C_{iq}\left(1 - \left(1 - \sum_{j\in\Gamma_i} \alpha_{ij}\right)X_i\right) + \sum_{i\in T}\sum_{j\in T}\sum_{k\in T} D_i C_{jk} W_{ijk} + \sum_{i\in T}\sum_{j\in\Gamma_i}\sum_{k\in T} D_i \alpha_{ij} C_{ki} Y_{ijk}\right\}. \tag{1}$$

Subject to

$$\sum_{k\in T} Y_{ijk} = X_i, \qquad i \in T, \quad j \in \Gamma_i. \tag{2}$$

$$Y_{ijk} \le \sum_{\substack{t\in T \\ t\neq k}} W_{itk}, \qquad i \in T, \quad j \in \Gamma_i, \quad k \in T, \quad k \neq i. \tag{3}$$

$$Y_{ijk} \le \sum_{\substack{t\in T \\ t\neq k}} W_{jtk}, \qquad i \in T, \quad j \in \Gamma_i, \quad k \in T, \quad k \neq j. \tag{4}$$

$$W_{itk} \le \sum_{\substack{j\in T \\ j\neq t}} W_{ijt}, \qquad i, k, t \in T, \quad k \neq t, \quad k \neq i, \quad i \neq t. \tag{5}$$

$$\sum_{k \in T} g_{ikt} - \sum_{\substack{k \in T \\ k \neq i}} g_{itk} = \sum_{j \in \Gamma_i} Y_{ijt} + \sum_{j \in \Gamma_i} Y_{jit}, \qquad i, t \in T, \quad t \neq i. \tag{6}$$

$$g_{ikt} \leq |T| \, W_{ikt}, \qquad i, k, t \in T. \tag{7}$$

$$X_i, \; Y_{ikt}, \; W_{ikt} \in \{0, 1\}, \quad g_{ikt} \geq 0, \qquad i, k, t \in T. \tag{8}$$

The first summation in (1) is the cost of transmitting fragments to the query site. The cost of transmitting fragment i to the query site is $F_i C_{iq}$ if it is not restricted by semijoin operations, and $F_i C_{iq} \sum_{j \in \Gamma_i} \alpha_{ij}$ if it is restricted. The selectivity factors are added because the restricting fragments belong to the same relation and the assumption that the fragments are disjoint. The first triple summation accounts for the cost of transmitting unrestricted join attributes to semijoin sites, and the second accounts for the cost of transmitting restricted join attributes back to their original sites.

The constraints in (2) state that if a fragment is to be semijoined, it has to be restricted by every fragment of the other relation. Constraints (3) and (4) say that in order for a semijoin to take place at a site, both of the join attributes must be available at that site. The constraints in (5) ensure that a join attribute can be transmitted from site t to site k only if it is available at site t. Constraints (6) and (7) are the flow constraints that guarantee the tree structure of the transfers of each unrestricted join attribute (see [14] for an analysis of cycle prevention constraints).

5. A LOWER BOUNDING PROCEDURE

An important part of the analytical analysis of the 2-way join problem is the provision of tight lower bounds on the value of an optimal solution to the model. Due to the inherent difficulty of the problem (a special case was proven to be NP-complete in [30]), it is expected that heuristic rather than optimal solutions will be used for a real-world application. The lower bounds, when compared to the value of heuristic solutions, provide information about the quality of the solutions generated by the heuristic procedures. For example, if in most cases the heuristic's value is only 1 percent greater than the optimal value, then there is no sense in devoting years of research to devise better algorithms. In the absence of lower bounds for the 2-way join problem, the only way to compare the heuristics' solutions to the optimal solution is by complete enumeration of the solution space. Conducting a significant empirical analysis using complete enumeration for sizable problems is likely to be too costly for most researchers.

Lower bounds are also needed if one wants to compute the optimal solution using a branch-and-bound algorithm [23]. Optimal solutions were not computed in the empirical analysis reported in this paper, due to the prohibitive amount of computation time required. Hence, if the heuristic's solution and the lower bound are 5 percent apart, we know that the heuristic's value is *at most* 5 percent away from the optimum.

The lower bounds derived in this paper are based on a Lagrangian relaxation coupled with a subgradient procedure (a description of the procedure is given below). Lagrangian relaxation [16] has been applied successfully in many combinatorial optimization problems such as location problems [7, 12, 18],

distribution systems design [17], the traveling salesman and related problems [13, 20], and the design of computer networks [14].

A summarized description of the Lagrangian relaxation follows. Interested readers can consult [16] for more details.

The general integer linear programming problem can be written as

$$(P) \underset{X \geq 0}{Min}\ \mathbf{cX}.$$

Subject to

$$\mathbf{AX} \geq \mathbf{b}$$
$$\mathbf{BX} \geq \mathbf{d}$$
$$X_i \text{ integer}, \quad i \in I.$$

Where **X**, **b**, **c**, and **d** are vectors, **A** and **B** are matrices of conformable dimensions, and the index set I denotes the variables required to be integer. It is assumed that the constraints $\mathbf{BX} \geq \mathbf{d}$ have a special structure that enables an efficient solution once the constraints $\mathbf{AX} \geq \mathbf{b}$ are eliminated. We define the Lagrangian relaxation of problem (P) relative to $\mathbf{AX} \geq \mathbf{b}$ and a conformable nonnegative vector λ to be

$$(PR_\lambda) \underset{X \geq 0}{Min}\ \mathbf{cX} + \lambda(\mathbf{b} - \mathbf{AX}).$$

Subject to

$$\mathbf{BX} \geq \mathbf{d}$$
$$X_i \text{ integer}, \quad i \in I.$$

Denoting the value of an optimal solution to problem (P) by $Z(P)$, we have the following relation:

$$Z(PR_\lambda) \leq Z(P).$$

Consequently, the Lagrangian relaxation of (P) constitutes a lower bound on the optimal value of (P). For the relaxation to be useful, problem (PR_λ) has to be significantly easier to solve than problem (P). For a given Lagrangian relaxation, the tightest lower bound which can be achieved is $\max_\lambda\{Z(PR_\lambda)\}$. The vector λ which achieves that maximization is the best set of Lagrangian multipliers.

The subgradient procedure is used to approximate the best Lagrangian multipliers. It is an iterative procedure that updates the vector λ at each iteration based on the value $Z(PR_\lambda)$. The main steps of the subgradient procedure are described next. For a detailed description of the procedure and its theory, see [20, 29]. The procedure starts with an initial vector λ^0 and the optimal solution $\mathbf{X}^0$ to problem (PR_{λ^0}). For each iteration t, the subgradient direction vector is calculated as

$$\Phi^t = \mathbf{b} - \mathbf{AX}^t.$$

Using a step size S^t, the new vector λ^{t+1} is given by

$$\lambda^{t+1} = \lambda^t + S^t\Phi^t.$$

A commonly used step size is

$$S^t = C^t \frac{\bar{Z}(P) - Z(PR_{\lambda^t})}{\| \Phi^t \|^2},$$

where $\bar{Z}(P)$ is an upper bound (feasible solution) on the value of (P), $\| \cdot \|$ denotes the norm function, and C^t is a scalar that is initially equal to 2 and is updated every k iterations.

The lower bound on the value of an optimal solution to the model of remote semijoins is derived through a Lagrangian relaxation of problem P_J presented in the previous section. The main idea in that relaxation is to reduce problem P_J to a set of uncapacitated plant location problems by relaxing constraints (4), (5), and (6). The plant location problem is concerned with a set of candidate plant sites and a set of customers each with a demand for a commodity that can be supplied from any site at which there is an open plant. The objective function is to satisfy the customers' demand and minimize the total cost, which consists of fixed costs of opening plants and variable costs of supplying the customers. Though NP-complete [7], the plant location problem is considered to be "easy." An optimization algorithm developed by [12] has exhibited excellent performance in many applications of the plant location problem. This procedure has been used in the solution of the Lagrangian relaxation of problem P_J. The mathematical details of the relaxation of this problem can be found in [31]. The lower bounds are used in Section 7 to evaluate the heuristic procedures presented next.

6. HEURISTIC PROCEDURES

Three heuristic procedures for solving problem P_J are developed in this section. The first heuristic is an iterative set selection heuristic, where the selected set consists of fragments to be restricted by semijoin operations. The second heuristic is a simpler (and faster) version of the first heuristic, and the third heuristic is a set selection procedure combined with an algorithm to optimize the transmissions associated with the selected semijoin operations.

We propose three procedures since they have different computational complexities, and therefore the trade-off between the computing time and the quality of the solutions can be explored. The behavior of these procedures is analyzed in Section 7.

6.1 An Add Procedure

In this section an "add" procedure is developed to derive a solution to problem P_J. The add procedure is a greedy set selection heuristic [23], which has been applied successfully to many applications. Many of the algorithms for distributed query optimization can be classified as add procedures (e.g., [3, 5]). These algorithms start with an empty set of semijoin operations and at each step add the most beneficial semijoin to the set. Add procedures differ from each other in the set to be selected and in 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. Consider the distributed query optimization problem. After a set of semijoin operations is selected, one might still

have to allocate the 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 advantage 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. The add procedure used in this paper selects the set of fragments to be restricted by semijoin operations. Initially this set is the empty set, and at each iteration it is augmented by the fragment that contributes most to the reduction in cost. If no such fragment exists, the procedure terminates. The procedure is stated as Algorithm 1, below. (A detailed formulation of Algorithm 1 is given in Appendix 1.) The following sets are used by Algorithms 1 and 2:

M is the set of fragments to be restricted by semijoin operations.
N is the set of fragments yet to be considered for restriction.
L_i is the set of sites at which join attribute i is available.

Algorithm 1 works as follows. When fragment i is considered for a restriction by fragment k, the following alternatives are considered:

(1) A local semijoin at site i is chosen and the join attribute of fragment k is transmitted to the site of fragment i. Join attribute k is available at the set of sites L_k, and the site that minimizes the communication cost of transmitting join attribute k to site i is chosen to be the sending site. The set L_k is augmented by $\{i\}$.
(2) A remote semijoin is chosen and join attribute i is transmitted to site j belonging to the set L_k. The site j is chosen such that the cost of transmitting join attribute i to site j and back (the restricted attribute) to site i is minimized. The set L_i is augmented by $\{j\}$.

For both alternatives, the semijoin cost is the cost of transmission as described above, and the semijoin benefit is the cost savings that results from sending only the restricted fragment to the query site. If neither alternative is beneficial and the algorithm terminated, the unrestricted fragment is transmitted to the query site.

Algorithm 1

1. (*Initialization*). Initialize M to the empty set, N to the set of all fragments, and total communication cost to the cost of transmitting the unrestricted fragments to the query site. $L_i = \{i\}$, $\forall i \in T$.
2. For every fragment in N calculate:
   Net cost = cost of semijoin − benefit of semijoin.
3. Let $\tilde{i}$ be the fragment with minimum net cost in step 2. If the net cost of $\tilde{i}$ is nonnegative stop. Otherwise continue.
4. Add the net cost of fragment $\tilde{i}$ to the total communication cost.
5. Update the sets L_i.
6. Remove $\tilde{i}$ from N and add it to M.
7. If the set N is empty stop; Otherwise goto step 2.

6.2 A Single Path Heuristic

In this section, a fast "single path" (i.e., each fragment is considered only once) algorithm is proposed. The underlying idea is to examine each fragment only once, and, if beneficial, to restrict it by semijoin operations. The procedure is

stated as Algorithm 2 below. (A detailed formulation of Algorithm 2 is given in Appendix 1.)

Algorithm 2

1. (*Initialization*). Initialize M to the empty set, N to the set of all fragments, and total communication cost to the cost of transmitting the unrestricted fragments to the query site. $L_i = \{i\}$, $\forall i \in T$.
2. For every fragment in N perform steps 3 and 4.
3. Net cost = cost of semijoin − benefit of semijoin.
4. If net cost is negative then do:
    1. Add net cost to total communication cost.
    2. Update the sets L_i.
    3. Remove the fragment from N and add it to M.

The logic of Algorithm 2 is basically the same as that of Algorithm 1, except that each fragment is considered for a restriction only once. If that restriction is beneficial, the fragment is added to the set M (step 4.3 of the algorithm); if the restriction is not beneficial, the fragment remains in the set N without any further consideration (i.e., it is transmitted unrestricted to the query site). Calculating net cost and updating the sets L_i in step 4 are done as explained for Algorithm 1.

6.3 A Set Query Based Heuristic

The 2-way join problem involves two types of decisions. Decisions of the first type determine the restriction of fragments by semijoin operations, and decisions of the second type determine the transmission plan of join attributes. In Algorithms 1 and 2 the transmission plan is determined during the process of selecting the set of fragments to be restricted. The algorithm proposed in this section consists of two stages. In the first stage the set of fragments to be restricted by semijoin operations is selected. The second stage optimizes the transmissions associated with semijoins selected in the first stage.

A mathematical model of set queries developed in [15] can be applied to the stage of optimizing the transmissions associated with remote semijoin operations. Set queries involve set operations (such as set difference and set intersection) between a single set of tuples (referred to as the *condition set*) and a group of other geographically dispersed sets of tuples (each one of these sets is referred to as a *target set*). The idea is to use the optimization procedure developed for set queries by using the intersection operator for semijoin operations as the set operation of the set queries model.

Optimizing the transmissions associated with the restriction of a *single* fragment can be represented as a set query problem in the following way. Assume that the single fragment to be restricted is some $i \in T$. Considering only the join attribute of the tuples, the condition set consists of the values of the join attribute i, and each target set j consists of the values of join attributes $j \in \Gamma_i$ (i.e., the join attributes of fragments of the other relation). The set operation in this case is set intersection (associated with the semijoins), and it has to be performed between join attribute i and each join attribute $j \in \Gamma_i$. The result of each set operation is a restricted version of join attribute i, and it has to be transmitted back to site i.

A generalized two-stage heuristic procedure, based on the set query model, to solve the 2-way join problem is stated as Algorithm 3 below. The details of the algorithm are dependent on the actual set selection procedure and the heuristic used to solve the set query problem, and is described below.

Algorithm 3

1. Apply a set selection procedure whose output is the set $M = \{i \mid i \in T$, and fragment i is to be restricted by a semijoin$\}$.
2. For every $i \in M$ solve the set query problem with the set of nodes $\{\Gamma_i + i\}$, where i is both the query site (of the set query problem) and the condition set site, and Γ_i are the set sites.
3. Eliminate duplicate transmissions.

Algorithm 3 consists of two conceptual parts. The first part determines the set of required semijoins (step 1), and the second part optimizes the data transmissions required to perform these semijoins (steps 2 and 3). Step 3 is required because the set query algorithm is applied to each selected semijoin operation at a time, and each join attribute might be involved in several semijoin operations.

The version of Algorithm 3 that was tested in this study used Algorithm 1 (without step 4) as step 1, and Algorithm AH from [15] as step 2. A statement of Algorithm AH in the context of the 2-way join problem is given as Algorithm 3.2 below. (A detailed statement of the algorithm is given in Appendix 1.) For further details about the set query model, see [15].

The following definitions are used in Algorithm 3.2:

$M =$: the set of sites at which join attribute i (the condition set) is available.
$N =$: the set of sites yet to be considered for a transmission of join attribute i.
$C_M =$: total transmission cost of the semijoin execution, given that join attribute i is transmitted to the set of sites M.
$C_j =$: total transmission cost of the semijoin execution, given that join attribute is transmitted to the set of sites $M + \{j\}$, where $j \in N$.

Algorithm 3.2 (used as step 2 of Algorithm 3)

1. Initialize M to the site of fragment i, N to Γ_i, and C_M to the cost of transmitting join attributes $j \in \Gamma_i$ to site i.
2. For every $j \in N$ calculate C_j as the sum of the cost of transmitting join attribute i to sites $M + \{j\}$, the cost of sending join attributes $k \in N$, $k \neq j$, to sites $M + \{j\}$, and the cost of transmitting the remote semijoin results to site i.
3. Find a $\bar{j}$ for which C_j (as calculated in step 2) is minimal. If $C_{\bar{j}} > C_M$ stop. Otherwise continue.
4. Replace the value of C_M by the value of $C_{\bar{j}}$; remove $\bar{j}$ from the set N and add it to the set M.
5. If the set N is empty stop. Otherwise go to step 2.

6.4 A Numerical Example

In this section a numerical example of Algorithm 1 is presented. Examples of Algorithms 2 and 3 are not presented here since Algorithm 2 is a simplified version of Algorithm 1, and detailed examples of the set query algorithms can be found in [15]. The example is based on the data given in Table I. Relation R1 consists of two fragments numbered 1 and 2, which are stored at sites 1 and 2 respectively. Relation R2 consists of two fragments numbered 3 and 4, which are

Table I. Data for an Example of Algorithm 1

i	F_i	D_i
1	17	5
2	12	3
3	18	2
4	14	1

C_{ij}

i \ j	q	1	2	3	4
q	0	1	2	2	3
1	1	0	3	4	2
2	2	3	0	1	3
3	2	4	1	0	4
4	3	2	3	4	0

α_{ij}

i \ j	1	2	3	4	$\sum_{j \in \Gamma_i} \alpha_{ij}$
1	1	1	0.2	0.1	0.3
2	1	1	0.1	0.3	0.4
3	0.4	0.1	1	1	0.5
4	0.2	0.1	1	1	0.3

stored at sites 3 and 4 respectively. The query site is represented by q in Table I (q is different from the fragment sites in this example, but can be one of those sites in general). $|S1| = |S2| = 2$. $\Gamma_i = \{3, 4\}$ if $i \in \{1, 2\}$, and $\Gamma_i = \{1, 2\}$ if $i \in \{3, 4\}$. The execution of Algorithm 1 is illustrated step by step.

Step 1. (*Initialization*)

$M = \varnothing$; $N = \{1, 2, 3, 4\}$; total communication cost $= \sum_{i=1,\ldots,4} F_i C_{iq} = 119$.

$L_1 = \{1\}$; $L_2 = \{2\}$; $L_3 = \{3\}$; $L_4 = \{4\}$.

Iteration 1

Step 2. Costs, benefits, and net costs of semijoining fragments in N:

Fragment	Cost	Benefit	Net cost
1	10.0	11.9	−1.9
2	5.0	14.4	−9.4
3	13.4	18.0	−4.6
4	5.7	29.4	−23.7

The net cost of fragment 1 is calculated as follows (the calculations for the other fragments are similar): the benefit of semijoining fragment 1 is $(1 - \alpha_{13} - \alpha_{14})F_1C_{1q} = 11.9$. The cost of the semijoin by fragment 3 is the minimum between the cost of a local semijoin ($D_3C_{31} = 8$) and the cost of a remote semijoin ($D_1C_{13} + D_1\alpha_{13}C_{31} = 24$). Similarly, the cost of the semijoin by fragment 4 is the minimum between $D_4C_{41} = 2$ and $D_1C_{14} + D_1\alpha_{14}C_{41} = 12$. It follows that the total cost of the semijoins is 10, and the net cost is $10 - 11.9 = -1.9$.

Step 3. Fragment 4 has minimum net cost, which is negative. The cost is incurred by two semijoins, one by fragment 1 and the other by fragment 2 (since all fragments of R1 must participate in the restriction of fragment 4).

Step 4. Total communication cost $= 119 - 23.7 = 95.3$.

Step 5. The semijoin of fragment 4 by fragment 1 is remote. Consequently, join attribute 4 is sent to site 1, and
$L_4 = L_4 + \{1\} = \{1, 4\}$.
The semijoin of fragment 4 by fragment 2 is remote. Hence
$L_4 = L_4 + \{2\} = \{1, 2, 4\}$.
Step 6. $M = M + \{4\} = \{4\}$; $N = N - \{4\} = \{1, 2, 3\}$.
Step 7. N is not empty implies continuation with step 2.

Iteration 2

Step 2. Costs, benefits, and net costs of semijoining fragments in N:

Fragment	Cost	Benefit	Net cost
1	8.0	11.9	−3.9
2	2.0	14.4	−12.4
3	13.4	18.0	−4.6

Step 3. Fragment 2 has minimum net cost, which is negative. The cost is incurred due to two semijoins, one by fragment 3 and the other by fragment 4.
Step 4. Total communication cost = 95.3 − 12.4 = 82.9.
Step 5. The semijoin of fragment 2 by fragment 3 is local, hence
$L_3 = L_3 + \{2\} = \{2, 3\}$.
The semijoin of fragment 2 by fragment 4 is local. Site 2 is already included in L_4, however.
Step 6. $M = M + \{2\} = \{2, 4\}$; $N = N - \{2\} = \{1, 3\}$.
Step 7. N is not empty.

Iteration 3

Step 2. Costs, benefits, and net costs of semijoining fragments in N:

Fragment	Cost	Benefit	Net cost
1	6.0	11.9	−5.9
3	13.4	18.0	−4.6

Step 3. Fragment 1 has minimum net cost, which is negative. The cost is incurred due to two semijoins, one by fragment 3 and the other by fragment 4.
Step 4. Total communication cost = 82.9 − 5.9 = 77.0.
Step 5. The semijoin of fragment 1 by fragment 3 is local, hence
$L_3 = L_3 + \{1\} = \{1, 2, 3\}$.
The semijoin of fragment 1 by fragment 4 is local. Site 1 is already included in L_4, however.
Step 6. $M = M + \{1\} = \{1, 2, 4\}$; $N = N - \{1\} = \{3\}$.
Step 7. N is not empty.

Iteration 4

Step 2. Costs, benefits, and net costs of semijoining fragments in N:

Fragment	Cost	Benefit	Net cost
3	13.4	18.0	−4.6

Step 3. Fragment 1 has negative net cost, and is to be semijoined by fragments 1 and 2.
Step 4. Total communication cost = 77.0 − 4.6 = 72.4.

Step 5. Both semijoins of fragment 3 are local. Sites 1 and 2 are included in L_3, however.
Step 6. $M = M + \{3\} = \{1, 2, 3, 4\}$; $N = N - \{3\} = \emptyset$.
Step 7. N is empty, so stop.

The plan generated by Algorithm 1 is to restrict all the fragments by semijoin operations. The total cost of this plan is 72.4, compared to 119, which is the cost of transmitting the fragments, unrestricted, to the query site. It is interesting to note that for this example the algorithm that allows only local semijoins (described in [30]) generates a plan of restricting fragments 1, 2, and 4 by semijoins, and a total cost of 97.3. The difference in the costs is due to the fact that the solution, generated by Algorithm 1 includes a restriction of fragment 3 by semijoin operations. This restriction is beneficial only when remote semijoin operations are included. The example illustrates that significant savings in cost (35 percent in the case of the example) can be realized by incorporating remote semijoin operations.

7. ANALYSIS OF 2-WAY JOIN ALGORITHMS

In this section we report the results of extensive computational experiments designed to evaluate the performance of 2-way join algorithms. Section 7.1 compares the heuristic procedures proposed in this paper to the lower bound of Section 5. Section 7.2 presents the results of a sensitivity analysis, which provides information both about the performance of each algorithm as a function of the system's parameters and about its performance relative to the other algorithms. Section 7.3 describes the set of experiments and their results regarding the effect of errors in estimating the size of intermediate results on the actual cost of processing the query.

The algorithms that have been developed in this paper and in [30] were intended to solve the same problem, namely the 2-way join problem. The major difference between the algorithms is the set of permissible transmissions of join attributes. Consequently, it is desirable to compare the relative performance of all those procedures in order to provide information about the applicability of each algorithm to a particular environment. This global analysis of the 2-way join algorithms enables a system designer to choose the most appropriate algorithm for the system being designed. In Sections 7.2 and 7.3, Algorithms 1, 2, and 3 of this paper are compared with two local semijoin algorithms from [30], referred to as Algorithms A and B. Algorithm A is a set selection procedure for local semijoins, and Algorithm B generates optimal solutions for the special case of direct transfers (i.e., a semijoin $R1_i \leftarrow R2_j$ takes place either at site i or site j, and join attribute j is either transferred from site j to site i or fragment i is transferred from site i to site j).[1]

Query optimization algorithms can be classified into two broad categories; those for fragmented databases and those for nonfragmented databases. A comparison of the algorithms proposed in this paper with algorithms for the nonfragmented environment is inappropriate because in such a case we have to

[1] In [30], Algorithm A is Algorithm 3 and Algorithm B is Algorithm 4. Note that Algorithm B is optimal for the special case of direct transfers, but is a heuristic in the general case.

Table II. Performance of Heuristics Relative to the Plant Location Lower Bound

		Relative cost[a] Algorithm 1				Relative cost Algorithm 2				Relative cost Algorithm 3			
$\|T\|$	$\|S1\|, \|S2\|$	AVG.	MIN.	MAX.	CPU[b]	AVG.	MIN.	MAX.	CPU	AVG.	MIN.	MAX.	CPU
5	2, 3	1.06	1.02	1.10	0.01	1.08	1.02	1.12	0.00	1.04	1.00	1.08	0.01
10	2, 8	1.03	1.01	1.06	0.02	1.07	1.03	1.10	0.01	1.03	1.00	1.07	0.07
	5, 5	1.07	1.02	1.10	0.03	1.09	1.04	1.13	0.02	1.04	1.00	1.09	0.08
15	2, 13	1.02	1.00	1.05	0.06	1.05	1.03	1.08	0.02	1.02	1.00	1.04	0.12
	7, 8	1.06	1.01	1.09	0.11	1.07	1.03	1.12	0.04	1.03	1.01	1.05	0.30
20	5, 15	1.03	1.00	1.09	0.21	1.05	1.01	1.08	0.07	1.03	1.01	1.06	0.62
	10, 10	1.08	1.03	1.11	0.24	1.09	1.02	1.14	0.08	1.05	1.01	1.08	0.75
30	5, 25	1.02	1.00	1.04	0.62	1.02	1.00	1.06	0.14	1.01	1.00	1.03	2.41
	15, 15	1.05	1.02	1.08	0.97	1.05	1.03	1.09	0.18	1.03	1.01	1.05	2.82
40	5, 35	1.06	1.03	1.11	1.11	1.02	1.00	1.04	0.24	1.02	1.00	1.04	3.72
	20, 20	1.08	1.03	1.12	1.36	1.06	1.04	1.09	0.26	1.04	1.01	1.09	4.93

[a] Relative cost of an algorithm = value of heuristic solution/lower bound.
[b] Average CPU time, measured in IBM4341 seconds.
Note. An early version of this paper had erroneous numbers in this table.

assume the assembly of the fragments of the relations before the algorithms can be applied, which contradicts the premise of our model. Consequently, any comparison is limited to the fragmented environment.

7.1 Heuristic Performance Relative to Lower Bound

Lower bounding and several heuristic procedures have been developed in this paper. These procedures were programmed in Fortran, and comparative computational experiments have been carried out. The objectives of the experiments reported in this section were to compare the heuristics to the lower bound and to evaluate the required computing time as a function of the problem size. The results of those experiments are summarized in Table II. The experiments were based on the following data: the fragment size was drawn from a uniform distribution with a range of [10, 20], the size of the join attribute ranged from 2 percent to 25 percent of the fragment size, the unit transmission costs were drawn from a uniform distribution with a range [0, 5], and the selectivity factors were drawn from a uniform distribution with a range of [0, 0.1]. Note that no specific units were assigned to the size and cost data, because the solution to the problem is effected by the relative and not absolute magnitudes. Hence the range [10, 20] can be interpreted as 10K to 20K, 10M to 20M, and so on.

The results presented in Table II are the average values of nine problems, and the values of the lower bounds were derived by a subgradient procedure applied to the Lagrangian relaxation of Section 5.

Based on the data presented in Table II, it seems that Algorithm 3 generates solutions with minimum communication costs among the three heuristic procedures, though at the expense of increased computation times. However, it is expected that the running time of Algorithm 3 can be reduced significantly by reprogramming the set query algorithms as an integral part of the algorithm. The reduction in running time is expected to be due to the elimination of duplicate transmissions during the solution of a set query problem, instead of a

separate step after the applications of the set query algorithms. In general all three heuristic procedures exhibited a good performance, and could be used in an actual application.

7.2 Sensitivity Analysis

The sensitivity analysis experiments were designed to analyze the sensitivity of solutions, generated by the heuristic procedures, to various parameters of the models (except the number of fragments, which was varied in Section 7.1). The model's parameters chosen to be varied and the rationale for their selection are as follows:

(1) *The selectivity factors.* The values of the selectivity factors are likely to have a significant effect on the solutions generated by the five algorithms considered here. Large selectivity factors may imply that semijoin operations will not significantly reduce the size of fragments, leading to solutions with similar costs generated by all five algorithms. For small values of the selectivity factors and join attributes that are not very large (a very large join attribute may render a semijoin useless, regardless of the size of the selectivity factor), it is expected that costs of solutions generated by the algorithms will significantly differ from each other.

(2) *The ratio of the join attribute's size to its fragment's size.* The communication cost of semijoin operations is incurred by transferring join attributes. Hence it is likely that join attributes whose size is small relative to their fragment will cause the algorithms to generate solutions that include more semijoin operations than solutions for the case of large join attributes. One would expect larger differences among the algorithms when the number of semijoin operations is large.

(3) *The variance of the size of fragments and join attributes.* The basic difference between the group of algorithms for the case of local semijoins and the group of algorithms for the case of remote semijoins is that the latter type of algorithms permit the restriction of a fragment at a remote site as well as at the local site. Consequently, it is likely that if an algorithm for the case of local semijoins chooses to restrict a fragment, the same decision will be made by an algorithm for the case of remote semijoins. However, a decision may be made by a local semijoin algorithm not to restrict a fragment i, $i \in T$, due to the large size of some join attribute j, $j \in \Gamma_i$ (which would have to be transmitted to site i), though the size of join attribute i is very small. This decision could be reversed by an algorithm that incorporates remote semijoins, since fragment i can be restricted by transmitting join attribute i instead. This leads to an intuitive prediction that the costs of solutions generated by the two types of algorithms will differ from each other when the variance of the size of join attributes is large.

(4) *The variance of the transmission cost rates.* The variance of the unit transmission costs was found to have a significant impact on one of the algorithms for set query optimization [15], whose performance deteriorated when the variance was large. This led us to investigate the effect of this parameter in the case of the 2-way join problem. However, we found that the variance of the unit transmission costs did not have any noticeable impact on the relative performance

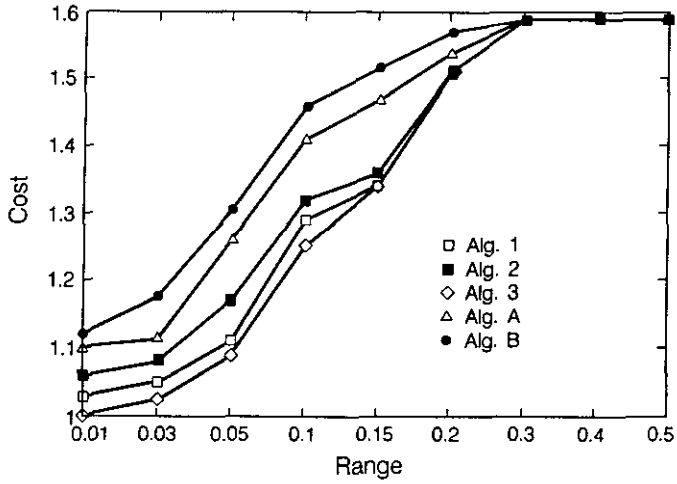

Fig. 4. Effect of selectivity factors.

of the five heuristic procedures. Therefore, the analysis will deal only with the effect of changes in the values of the other parameters.

The purpose of the sensitivity analysis is to validate the above conjectures and to determine what is "small" and what is "large." The input data that was fixed for all the sensitivity analysis experiments consisted of the number of fragments of the first relation ($|S1| = 15$), the number of fragments of the second relation ($|S2| = 10$), and the unit transmission costs ($C_{ij} \sim U(0, 5)$).[2]

The effect of changes in the values of the selectivity factors α_{ij} is shown in Figure 4. Every point of the graphs in Figure 4 represents the average cost of nine sample problems. All costs were normalized by dividing them by the value of the minimum average cost. The selectivity factors were derived by first generating $\alpha'_{ij} \sim U(0, b)$, where b is the range's upper limit, as specified in Figure 4, and then setting $\alpha_{ij} = \alpha'_{ij}/|\Gamma_i|$. Dividing α'_{ij} by $|\Gamma_i|$ ensures that $\hat{\alpha}_i = \sum_{j \in \Gamma_i} \alpha_{ij} \leq b$. $\hat{\alpha}_i$ may be viewed as the total selectivity factor of fragment i. The fragment size F_i was drawn from a uniform distribution with a range of [10, 20], and the size of the join attribute D_i was derived by generating $d_i \sim U(0.02, 0.20)$ and multiplying it by F_i.

The results of the experiments validated the intuitive conjecture: for large values of the selectivity factors ($\alpha'_{ij} \sim U(0, b)$, $b \geq 0.3$), the values of all solutions generated by the five heuristic procedures converged to the value of no semijoin solution (i.e., performing a direct join at the assembly site). In general, the smaller the selectivity factors, the larger the gap between the values of solutions generated by algorithms for the case of only local semijoins and those generated by algorithms that incorporate both local and remote semijoins. Figure 4 demonstrates the effectiveness of semijoin operations when the selectivity factors are small (up to 50 percent reduction in cost occurred in the experiments).

[2] Whenever convenient, we use the notation $p \sim U(a, b)$ to mean that the values of a parameter p were drawn randomly from a uniform distribution with a range $[a, b]$.

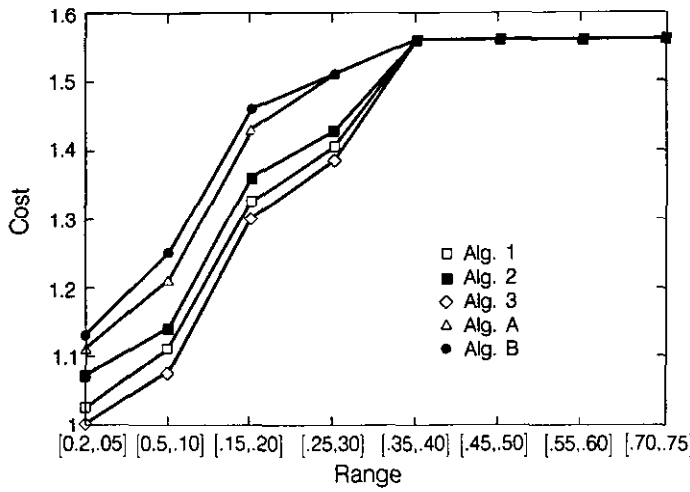

Fig. 5. Effect of relative size.

The effect of the size of the join attribute relative to its fragment's size is shown in Figure 5. The input data to the experiments were the same as for the case of Figure 4, except that the selectivity factors were drawn from a fixed range ($\alpha'_{ij} \sim U(0, 0.05)$). The size of join attributes (as a fraction of the fragments' size) was drawn from a uniform distribution with a range of $[a, b]$. The costs of the generated solutions as a function of the range $[a, b]$ are shown in Figure 5. The behavior of the algorithms, as exhibited by the graphs in Figure 5, is very similar to the case where the selectivity factors were varied. When the join attributes are large, semijoin operations are not beneficial, even for the small selectivity factors selected, and the values of solutions generated by all five heuristic procedures converge to the value of no semijoin solution.

Figure 6 illustrates the effect of variance of the size of fragments and their join attributes. The fragment size F_i was drawn from a uniform distribution with a range of $[5, b]$, where b is as specified in the figure, and the size of the join attribute D_i was derived by generating $d_i \sim U(0.02, 0.20)$ and multiplying it by F_i. The variance of the size of fragments, and consequently the variance of the absolute size of the join attributes, was controlled by changing the value of b. All the cost values for a particular range were divided by the value of the solution, generated by Algorithm 3 for that range, and the relevant information in Figure 6 is the change in the relative difference among the algorithms as a function of the range $[5, b]$. Figure 6 demonstrates a general increase in the gap between the performance of algorithms that utilize only local semijoins and algorithms that incorporate remote semijoins, as a function of the variance of the size of fragments (the smaller the variance, the smaller the gap).

Though Figures 4 through 6 have shown that the difference in performance between local and remote semijoin algorithms may vary as a function of the parameters' values, the difference in performance among the three algorithms that incorporate remote semijoin operations was found to be less sensitive to changes in the values of the model's parameters. The experiments have

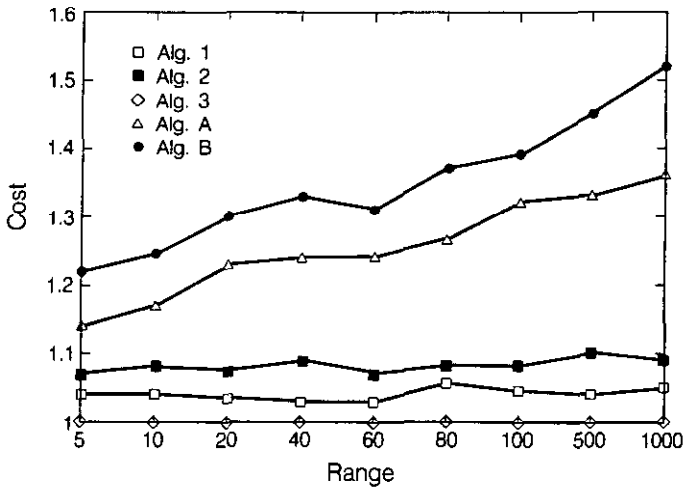

Fig. 6. Effect of variance in fragment size.

demonstrated again that incorporating semijoin operations in general and remote semijoin operations in particular can significantly reduce the communication costs incurred by processing a query.

Two parameters of the model were found to have a substantial impact on the relative performance of the heuristic procedures: the selectivity factors and the size of join attributes. Both parameters have had a similar effect on the relative performance. Large values of the selectivity factors and/or the size of join attributes reduced the differences among the values of solutions generated by the five heuristic procedures, due to increased number of fragments being transmitted, unrestricted, to the query site. Though the computational experiments were carried out under the assumptions of the 2-way join problem, it is predicted that the effect of the size of selectivity factors and join attributes will be similar for other problem instances. If that conjecture is validated, it may lead to the identification of those cases where semijoin algorithms will perform unsatisfactorily.

7.3 Error Analysis

The objective of this section is to analyze the effect of errors in estimating the values of the selectivity factors α_{ij}. Three aspects of that effect are investigated. The effect of errors on the actual cost of processing the query, using a particular optimization algorithm, is examined in Section 7.3.1. The effect of errors on the actual cost of processing a query as a function of the model's parameters is discussed in Section 7.3.2, and a comparison of the heuristic procedures, based on the actual cost of processing the query, is made in Section 7.3.3.

The purpose of analyzing the sensitivity of the algorithms to errors in estimating result sizes is to provide information about the benefits of dedicating resources to obtain more accurate estimates. These benefits are estimated as

follows (it is assumed that perfect information is available on all parameters but the selectivity factors): Let

- a denote the actual selectivity factors;
- e denote the estimated selectivity factors;
- P_a denote the transmission plan generated by an algorithm, using the actual selectivity factors (perfect information); and
- P_e denote the transmission plan generated by an algorithm, using the estimated selectivity factors.

We define the actual cost of a transmission plan to be the cost of that plan as calculated from the objective function of the optimization model using the actual (true) values of the selectivity factors. The estimated cost of a plan is similarly calculated, but by using the estimated selectivity factors. This section deals with three types of costs as defined next. Let

- $C_e(P_e)$ denote the estimated cost of executing a query according to plan P_e. The cost is calculated by using the estimated selectivity factors. This cost is the usual criterion used to evaluate the cost of transmission plans generated by query optimization algorithms.
- $C_a(P_e)$ denote the actual cost (calculated by using the actual selectivity factors) of a transmission plan generated by an algorithm that uses the estimated selectivity factors. This cost can be calculated if the estimation errors are known.
- $C_a(P_a)$ denote the actual cost of a transmission plan generated by an algorithm using the actual selectivity factors. This cost can be calculated if the estimated selectivity factors and the estimation errors are given, and is used to evaluate the benefit of getting more accurate estimates. An estimation error of $\pm r$ percent implies that

$$a = e\left(1 \pm \frac{r}{100}\right).$$

The difference $C_a(P_e) - C_a(P_a)$ is the value of replacing the estimates by perfect information. If the ratio $C_a(P_e)/C_a(P_a)$ is close to 1 for a given estimate, there is probably no benefit in improving the accuracy of the estimate.

7.3.1 *Effect of Estimation Errors on the Actual Costs.* The experiments reported in this section were carried out with the same input data that was used for the experiments in Section 7.1 (except for the parameters being varied). Every point on the graphs in Figures 7 through 12 is the average value of nine sample problems.

Figure 7 shows the effect of underestimating the selectivity factors on the ratio $C_a(P_e)/C_a(P_a)$. The cost of not having perfect information is quite significant, though moderate for errors smaller than 30 percent. It seems that the ratio $C_a(P_e)/C_a(P_a)$ is larger for the iterative procedures (Algorithms A, 1, and 3) than for the single path procedures (Algorithms B and 2). Figure 8 illustrates the effect of overestimating the selectivity factors on the ratio of actual costs. The relative difference among the five heuristic procedures is similar to the case of

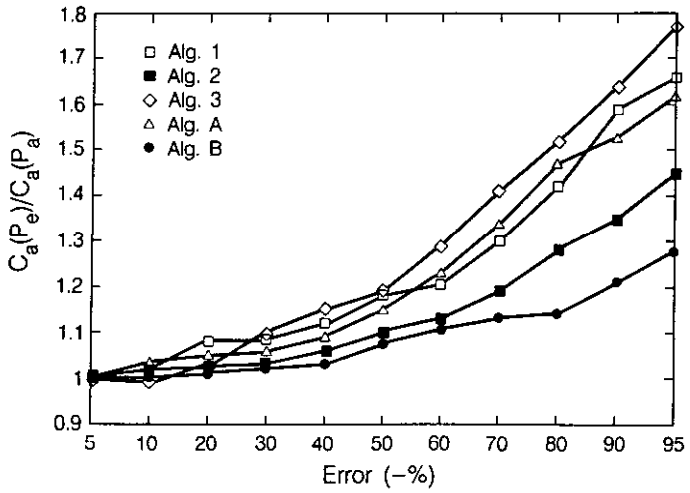

Fig. 7. $C_a(P_e)/C_a(P_a)$ as a function of underestimation.

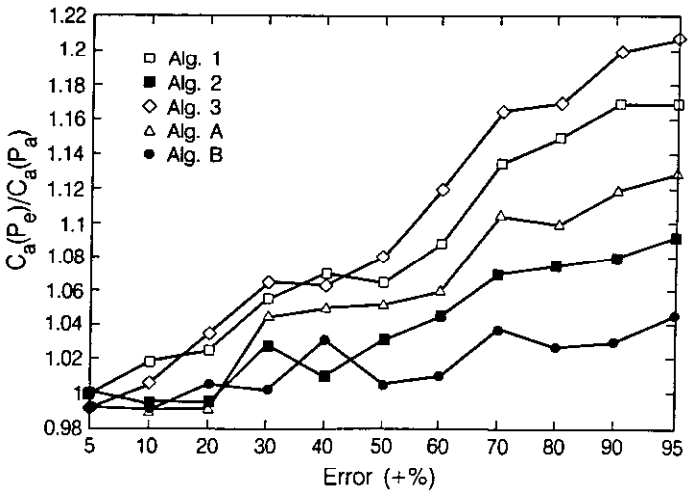

Fig. 8. $C_a(P_e)/C_a(P_a)$ as a function of overestimation.

underestimation, but there is a sharp reduction in the absolute value of the ratio $C_a(P_e)/C_a(P_a)$.

The difference between the case of underestimation and the case of overestimation is likely to be dependent on the values of the actual selectivity factors. If, under perfect information, a "no semijoin" decision is made for many of the fragments, then underestimation of the selectivity factors is likely to have stronger effect on the actual cost than an overestimation. To see that consider the extreme case where the solution generated by a particular algorithm, under perfect information, is to transfer all fragments, unrestricted, to the query site ("no semijoins" policy). Clearly, if the selectivity factors are overestimated, the

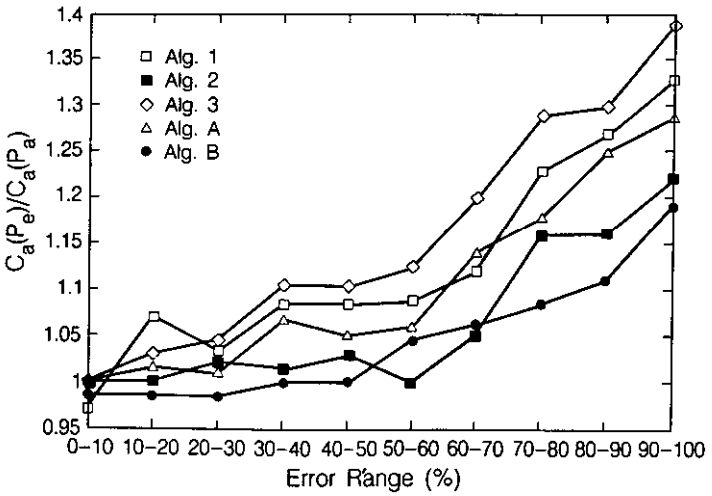

Fig. 9. Effect of random errors on the ratio of actual costs.

algorithm will generate the same "no semijoins" policy. However, it is quite possible that for underestimated selectivity factors, a completely different policy will be generated.

The effect of random errors in estimating the selectivity factors on the ratio $C_a(P_e)/C_a(P_a)$ is shown in Figure 9. The estimation errors were drawn from a uniform distribution with error ranges as specified in the figure. This was done independently for each fragment, coupled with a random generation of the error sign. As can be seen from the graphs in Figure 9, the relative difference among the five heuristic procedures is smaller for the case of random errors than for the cases of consistent underestimates and overestimates. The random generation of the error sign seems to have a moderating effect on the ratio $C_a(P_e)/C_a(P_a)$ compared to the case of underestimates. However, the ratio still remains large for large error values.

7.3.2 *Sensitivity of Errors' Effect to Model's Parameters.* The experiments reported in this section were designed to test the effect of errors in estimating the selectivity factors on the ratio $C_a(P_e)/C_a(P_a)$ as a function of the model's parameters, to help a system designer to choose the most appropriate optimization algorithm for a particular environment. Note that the analysis in this section is different than that in Section 7.2 because only estimated costs were considered in Section 7.2. The results reported in this section provide the methodological basis for investigating an actual system. The input data that was used for the experiments was the same as for the sensitivity analysis experiments (see Section 7.2). The estimation error was fixed at 50 percent underestimate.

Figure 10 shows the effect of 50 percent underestimation of the selectivity factors as a function of their values. The estimated selectivity factors were drawn from the ranges specified in Figure 10 as described in Section 7.2 (since the lower limits were set to zero, only the upper limits appear in the figure). As can be seen from the figure, the ratio $C_a(P_e)/C_a(P_a)$ is decreasing as the values of the

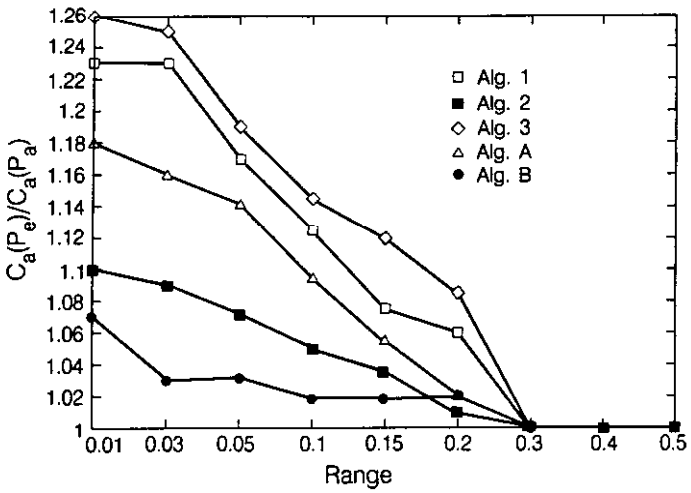

Fig. 10. Effect of estimation errors as a function of the selectivity factors.

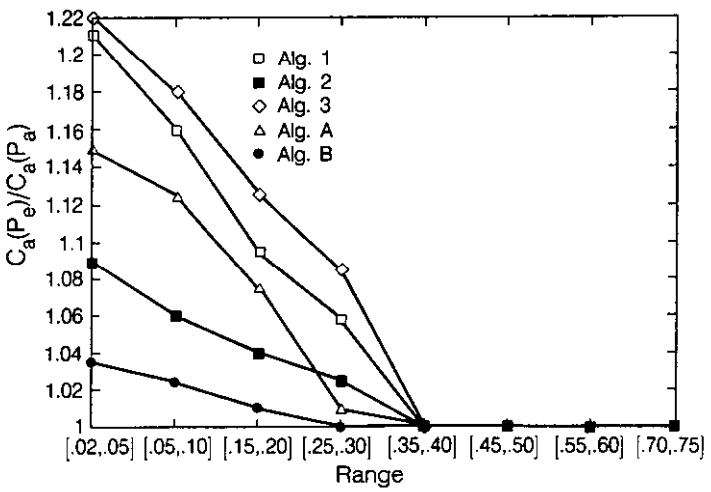

Fig. 11. Effect of estimation errors as a function of join attribute size.

selectivity factors are increasing. This behavior was expected because the larger the values of the selectivity factors, the larger is the number of fragments being transferred, unrestricted, to the query site. Eventually, the solutions generated by the heuristic procedures converge to the no semijoin solution, both for the actual values and for the estimated values of the selectivity factors, and the ratio $C_a(P_e)/C_a(P_a)$ becomes 1.

Figure 11 shows the effect of 50 percent underestimation of the selectivity factors as a function of the size of the join attributes. The size of the join attribute as a fraction of the fragment size was drawn from a uniform distribution with a range of $[a, b]$, where a and b are as specified in the figure. The effect on the

ratio $C_a(P_e)/C_a(P_a)$ is similar to Figure 10: it is decreasing as the size of the join attributes is increasing.

The variance of the size of fragments was found to have no significant impact on the ratio $C_a(P_e)/C_a(P_a)$ for a fixed 50 percent underestimation.

7.3.3 *Comparison of Heuristic Procedures Based on Actual Cost.* It is customary to use $C_e(P_e)$ as a criterion for selecting an optimization algorithm. A better choice would be to compare the algorithms based on $C_a(P_e)$, because in an actual environment the transmission plan is determined with estimated selectivity factors, and the cost that matters is the actual cost of that plan. This can be achieved by examining $C_a(P_e)$ over the expected range of errors, and then averaging the cost differences between the algorithms over that range. In this section we distinguish between two cases of errors in estimating the selectivity factors. In the first case the magnitude of an estimation error is identical for every fragment (constant error), and in the second case the errors are introduced randomly (variable error). We evaluate the first case analytically and the second case empirically.

The case of constant error. Let P^i denote the transmission plan generated by Algorithm i. Then, for a constant error size r, $C_a(P_e^i) - C_e(P_e^i)$ is shown to be linear in r for all the 2-way join algorithms presented in this paper. Let **X**, **Y**, and **W** be the values of the decision variables in problem P_J as determined by an arbitrary Algorithm i. Also define the following sets, where n can assume either the value 1 or the value 0.

$$S_n^x = \{i \mid X_i = n\}$$

$$S_n^y = \{i, j, k \mid Y_{ijk} = n\}$$

$$S_n^w = \{i, j, k \mid W_{ijk} = n\}.$$

It follows from (1) that

$$C_e(P_e^i) = \sum_{i \in S_0^x} F_i C_{iq} + \sum_{i \in S_1^x} \left(F_i C_{iq} \sum_{j \in \Gamma_i} \alpha_{ij} \right) + \sum_{i,j,k \in S_1^y} D_i \alpha_{ij} C_{ki} + \sum_{i,j,k \in S_1^w} D_i C_{jk}. \tag{9}$$

$$C_a(P_e^i) = \sum_{i \in S_0^x} F_i C_{iq} + \sum_{i \in S_1^x} \left(F_i C_{iq} \sum_{j \in \Gamma_i} \left(1 \pm \frac{r}{100} \right) \alpha_{ij} \right) + \sum_{i,j,k \in S_1^y} D_i \left(1 \pm \frac{r}{100} \right) \alpha_{ij} C_{ki} + \sum_{i,j,k \in S_1^w} D_i C_{jk}. \tag{10}$$

Subtracting (9) from (10), we get

$$C_a(P_e^i) - C_e(P_e^i) = \frac{\pm r}{100} \left(\sum_{i \in S_1^x} \left(F_i C_{iq} \sum_{j \in \Gamma_i} \alpha_{ij} \right) + \sum_{i,j,k \in S_1^y} D_i \alpha_{ij} C_{ki} \right). \tag{11}$$

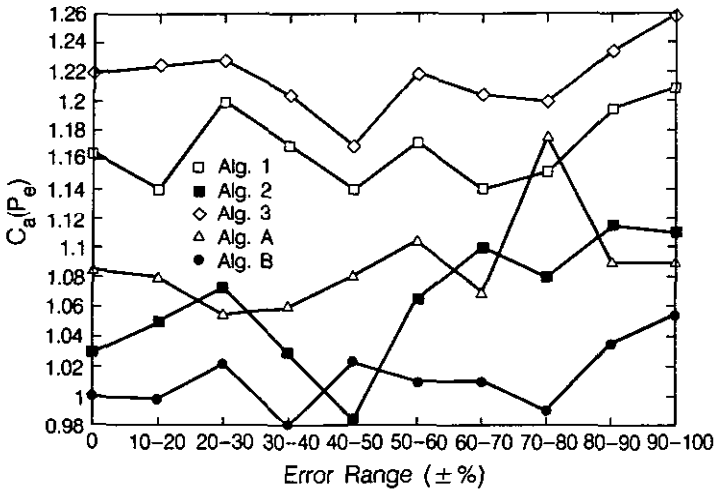

Fig. 12. A comparison of heuristic procedures based on actual costs.

Equation (11) implies that the ratio

$$\frac{C_a(P_e^i) - C_e(P_e^i)}{C_a(P_e^k) - C_e(P_e^k)}$$

is a constant for any two Algorithms i and k, and a constant underestimate or overestimate of r percent. The value of the first summation in (11) is a function of the number of fragments restricted by semijoin operations ($|S_i^*|$), and the value of the second summation is a function of the number of remote semijoin operations ($k = i$ implies a local semijoin operation, and $k \neq i$ implies a remote semijoin operation). Hence a comparison of the five heuristic procedures based on $C_e(P_e)$ is likely to differ from a comparison based on $C_a(P_e)$, if the magnitude and sign of errors vary among different semijoin operations and the sets of semijoin operations differ among the policies generated by the heuristic procedures.

The case of variable error. The case of random estimation errors was analyzed empirically, and the results are presented in Figure 12. The input data (other than the errors) for this experiment were the same as for the case of Figure 9. The point of zero-error in Figure 12 can be viewed as $C_e(P_e)$ because at this point $C_a(P_e) = C_e(P_e)$. As can be seen from the figure, the relative performance of the heuristic procedures differs in the case of $C_e(P_e)$ (zero-error point) from that in the cases of $C_a(P_e)$ (for the other error points). A conclusion of the experiments is that no trend is evident, but there are significant differences between the performance of the algorithms at different error points. Consequently, it is possible that a complex algorithm that outperforms a simple algorithm according to the estimated cost will show equivalent performance according to the actual cost. Therefore, it is worth checking $C_a(P_e)$ on the error range in a real system. It seems, however, that the results of such comparisons are data and algorithm dependent. Another difficulty in applying such an analysis stems from the fact

that estimating the error range might be as imprecise as estimating the selectivity factors themselves.

8. SUMMARY

This paper has introduced two types of semijoin strategies, *local* and *remote*. Using the semijoin operation as a size reduction operation, a mathematical model has been developed for the case of remote semijoins. Lower bounding and heuristic procedures have been proposed and the results of computational experiments presented. These experiments have revealed good performance of the heuristic procedures, and demonstrated the benefit of using semijoin operations to reduce the size of fragments. The computational experiments have revealed a significant difference between algorithms developed for the case of local semijoins and those developed for the case of remote semijoins. The latter type of algorithms (Algorithms 1, 2, and 3) were found to be generally superior to the first type (Algorithms A and B).

The importance of horizontal partitioning has been stressed in Section 1. Given that type of data topology, 2-way joins are important operations, since it is reasonable to assume that many of the queries will refer to two relations. The number of fragments in a horizontally partitioned database system might be large for a geographically dispersed organization. In such a case the problem's complexity will render the use of optimal procedures too costly, and good heuristic procedures will have to be employed. The results presented in this paper have important practical implications. When a system designer is confronted with an actual system and a need to choose an optimization procedure, he or she can base the decision on the results of computational experiments, similar to the ones reported in this paper. For example, it has been demonstrated that for large values of the selectivity factor and/or the size of joining attributes, many of the fragments will be transferred directly to the query site. Hence, based on simulations of a particular system, it may be appropriate to establish cut-off points for the values of the parameters, above which there is no benefit in using a semijoin algorithm. The lower bounds developed in this paper can also provide a system designer with valuable help in evaluating the performance of a particular algorithm.

The model developed for the case of 2-way joins was based on a set of assumptions, one of which was the assumption that the results of all semijoin operations, which are applied to the same fragment, are disjoint. This assumption has been introduced to simplify the introduction of the mathematical model. Its practical implication is that the policies generated by optimal procedures could be conservative (i.e., decisions could be made not to restrict fragments, though their restriction by semijoin operations is beneficial). However, it is possible to overcome this drawback in the following way. In the case of local semijoin operations [30], the models are concerned with only the sum of the selectivity factors for a fragment. Hence $\sum_j \alpha_{ij}$ could be replaced with an estimate β_i that takes into account the fact that the sets of tuples, resulting from the restriction of a fragment by semijoin operations, are not disjoint. Note that $\beta_i \leq \sum_j \alpha_{ij}$ because the results of the semijoins may overlap. In the case of remote semijoin operations both α_{ij}'s and β_i's should be used by the heuristic procedures. β_i will

be used to evaluate the cost of transferring the restricted fragments to the query site, and α_{ij} will be used to evaluate the cost of semijoin operations.

Some of the assumptions made in this paper are more restrictive, and their elimination changes the nature of the optimization model. Dealing with the removal of these assumptions is the subject of future research. One of the restrictive assumptions was about nonreplicated data. This assumption, however, is more of a restriction on the underlying data allocation than a restriction on the optimality of the model.

The other restrictive assumption which has an implication on the optimality of the model is the selection of the query site as the join site. It is interesting to note that this assumption became more restrictive after another restriction—constant communication cost rates—had been removed. If the communication cost rates are identical for every pair of sites, then the mathematical expression of the net benefit of a semijoin is independent of the assembly site chosen (assuming, of course, that the fragment stored at the assembly site need not be semijoined). If the model developed in this paper is augmented such that the selection of the join site is regarded as a decision variable, then the algorithms that do not use semijoins become a special case of semijoin algorithms because the latter algorithms also consider non-semijoin strategies.

Additional future-research topics include the problem of multiple join attributes and relations, and other objective functions.

APPENDIX 1. A Detailed Description of the Heuristic Procedures

Algorithm 1

1. (*Initialization*). $M = \emptyset$; $N = T - M$; $L_k = \{k\}$, $\forall k \in T$; $\bar{C} = 0$.
2. For every $i \in N$ evaluate B_i:
   $a_i = F_i(1 - \sum_{j \in \Gamma_i} \alpha_{ij})C_{iq}$,
   $b_i = \sum_{k \in \Gamma_i} (\text{Min}\{\text{Min}_{j \in L_k}\{D_k C_{ji}\}, \text{Min}_{j \in L_k}\{D_i(C_{ij} + \alpha_{ik}C_{ji})\}\})$,
   $B_i = b_i - a_i$.
3. Let $\tilde{i}$ be the index i which achieves $\text{Min}_{i \in N}\{B_i\}$. If $B_{\tilde{i}} \geq 0$, go to step 8.
4. $\bar{C} = \bar{C} + B_{\tilde{i}}$.
5. For every $k \in \Gamma_{\tilde{i}}$,
   if $\text{Min}_{j \in L_k}\{D_k C_{ji}\} < \text{Min}_{j \in L_k}\{D_i(C_{\tilde{i}j} + \alpha_{\tilde{i}k}C_{j\tilde{i}})\}$,
   then $L_k = L_k + \{\tilde{i}\}$.
   Otherwise, $L_{\tilde{i}} = L_{\tilde{i}} + \{k\}$.
6. $M = M + \{\tilde{i}\}$; $N = N - \{\tilde{i}\}$.
7. If $N \neq \emptyset$, go to step 2.
8. Total cost = $\sum_{i \in T} F_i C_{iq} + \bar{C}$.

Algorithm 2

1. (*Initialization*). $M = \emptyset$; $N = T$; $L_k = \{k\}$, $\forall k \in T$; $\bar{C} = 0$.
2. For every $i \in T$ do steps 3 and 4.
3. Calculate: $a_i = F_i(1 - \sum_{j \in \Gamma_i} \alpha_{ij})C_{iq}$.
   $b_i = \sum_{k \in \Gamma_i} (\text{Min}\{\text{Min}_{j \in L_k}\{D_k C_{ji}\}, \text{Min}_{j \in L_k}\{D_i(C_{ij} + \alpha_{ik}C_{ji})\}\})$.
4. If $a_i > b_i$, then do:
   $\bar{C} = \bar{C} + (b_i - a_i)$,
   $M = M + \{i\}$; $N = N - \{i\}$.
   For every $k \in \Gamma_i$,
   if $\text{Min}_{j \in L_k}\{D_k C_{ji}\} < \text{Min}_{j \in L_k}\{D_i(C_{ij} + \alpha_{ik}C_{ji})\}$,
   then $L_k = L_k + \{i\}$.
   Otherwise, $L_i = L_i + \{k\}$.
5. Total cost = $\sum_{i \in T} F_i C_{iq} + \bar{C}$.

Algorithm 3.2 (used as step 2 of Algorithm 3)

1. (*Initialization*). Let i be a fragment to be semijoined, $M = \{i\}$, $N = \Gamma_i$, $C_M = \sum_{j \in N} D_j C_{ji}$.
2. For every $j \in N$ calculate C_j as follows:
   a. The cost of transmitting join attribute i (the condition set) is the cost of a minimum spanning tree on the complete graph $G = (D, A)$, where the node set is $D = M + \{j\}$, the arc set is $A = \{a_{ij}\}$, and arc costs are $C(a_{ij}) = C_{ij} D_i$.
   b. The transmission cost of join attributes $k \in N$, $k \neq j$ (the sets), and of the restricted versions of join attribute i is

$$\sum_{\substack{k \in N \\ k \neq j}} \operatorname{Min}_{t \in M + \{j\}} \{D_k C_{kt} + D_i \alpha_{ik} C_{ti}\} + \sum_{\substack{t \in M + \{j\} \\ t \neq i}} D_i \alpha_{it} C_{ti}.$$

   C_j is the sum of costs a. and b.
3. Let $\bar{j}$ be the index j that achieves $\operatorname{Min}_{j \in N}\{C_j\}$. If $C_{\bar{j}} > C_M$, stop.
4. $C_M = C_{\bar{j}}$; $M = M + \{\bar{j}\}$; $N = N - \{\bar{j}\}$.
5. If $N = \emptyset$, stop. Otherwise go to step 2.

ACKNOWLEDGMENTS

I would like to thank Professor Bezalel Gavish, Dr. Arie Shoshani, and the anonymous referees for their helpful comments and suggestions.

REFERENCES

1. APERS, P. M. G. Distributed query processing: Minimum response time schedules for relations. IR 53, Vrije Univ., Amsterdam, Nov. 1979.
2. APERS, P. M. G., HEVNER, A. R., AND YAO, S. B. Optimization algorithms for distributed queries. *IEEE Trans. Softw. Eng. SE-9*, 1 (Jan. 1983), 57–68.
3. BERNSTEIN, A. P., ET AL. Query processing in a system for distributed databases (SDD-1). *ACM Trans. Database Syst. 6*, 4 (Dec. 1981), 602–625.
4. BERNSTEIN, A. P., AND CHIU, D W. Using semijoins to solve relational queries. *J. ACM 28*, 1 (Jan. 1981), 25–40.
5 CHANG, J.-M. A heuristic approach to distributed query processing. In *Proceedings of the 8th International Conference on Very Large Data Bases* (1982), 54–61.
6. CODD, E. F. A relational model for large shared data banks. *Commun. ACM 13*, 6 (June 1970), 909–917.
7. CORNUEJOLS, G., FISHER, M. L., AND NEMHAUSER, G. L. Location of bank accounts to optimize float. *Manage. Sci. 23*, 8 (Apr. 1977), 789–810.
8. DANIELS, D. Query compilation in a distributed database system. IBM Res. Rep. RJ3423, Mar. 1982.
9. DANIELS, D., SELINGER, P., HAAS, L., LINDSAY, B., MOHAN, C., WALKER, A., AND WILMS, P. An introduction to distributed query compilation in R*. IBM Res. Rep. RJ3497, June 1982.
10. DATE, C. J. *An Introduction to Database Systems*. Addison-Wesley, Reading, Mass., 1981.
11. EPSTEIN, R. S., STONEBRAKER, M., AND WONG, E. Distributed query processing in a relational database system. In *Proceedings ACM-SIGMOD* (May 1979), ACM, New York, 169–180.
12. ERLENKOTTER, D. A dual-based procedure for uncapacitated facility location. *Oper. Res. 26*, 1 (1978), 992–1009.
13. GAVISH, B., AND SRIKANTH, S. K. Optimal solution methods for large-scale multiple traveling salesman problem. Working Paper, Graduate School of Management, Univ. of Rochester, Rochester, N.Y., 1980.
14. GAVISH, B. Topological design of centralized computer networks—formulations and algorithms. *Networks 12* (1982), 355–377.
15. GAVISH, B., AND SEGEV, A. Set query optimization in horizontally partitioned distributed database systems. Working Paper QM 8304, Graduate School of Management, Univ. of Rochester, Rochester, N.Y., 1983.
16. GEOFFRION, A. M. Lagrangian relaxation and its uses in integer programming. *Math. Program. Stud. 2* (1974), 82–114.

17. GEOFFRION, A. M., AND GRAVES, G. W. Multicommodity distribution system design by Benders decomposition. *Manage. Sci. 20* (1974), 822–844.
18. GEOFFRION, A. M., AND MCBRIDGE, R. Lagrangian relaxation applied to capacitated facility location problems. *AIIE Trans. 10* (1978), 40–47.
19. GOODMAN, N., ET AL. Query processing in SDD-1: A system for distributed databases. Computer Corporation of America, Tech. Rep. CCA-79-06, 1979.
20. HELD, M., AND KARP, R. M. The traveling salesman problem and minimum spanning trees. *Oper. Res. 18* (1970), 1138–1162.
21. HELD, G. D., STONEBRAKER, M., AND WONG, E. INGRES—a relational DBMS. In *Proceedings of AFIPS 1975 National Computer Conference* (Arlington, Va., 1975), 409–416.
22. HEVNER, A. R., AND YAO, S. B. Query processing in distributed database systems. *IEEE Trans. Softw. Eng. SE-5*, 3 (May 1979), 177–187.
23. HOROWITZ, E., AND SAHNI, S. *Fundamentals of Computer Algorithms*. Computer Science Press, Rockville, Md., 1978.
24. MULGREW, T. Standard Oil of California. Seminar at the University of California, Berkeley, 1984.
25. PAIK, I.-S., AND DELOBEL, C. A strategy for optimizing the distributed query processing. In *Proceedings of the 1st International Conference on Distributed Computing Systems* (Huntsville, Ala., Oct. 1979).
26. PEEBLES, R. W., AND MANNING, E. D. A computer architecture for large (distributed) databases. In *Proceedings of the Conference on Very Large Databases* (Framingham, Mass., Sept. 1975).
27. PELAGATTI, G., AND SCHREIBER, F. A. A model of an access strategy in a distributed database system. In *Proceedings of the IFIP-TC2, Database Architecture* (Venice, 1979).
28. ROTHNIE, J. B., AND GOODMAN, N., ET AL. Query processing in SDD-1: a system for distributed databases. Computer Corporation of America, Tech. Rep. 79-06, Oct. 1979.
29. SANDI, C. Subgradient optimization. In *Combinatorial Optimization*, N. Christofides, A. Mingizzi, P. Toth, and C. Sandi, Eds., Wiley, New York, 1979.
30. SEGEV, A. Optimizing 2-way joins in fragmentd database systems. Working Paper MS-2, School of Business Administration, Univ. of California, Berkeley, 1983.
31. SEGEV, A. Algorithms for distributed data processing. Working Paper, School of Business Administration, Univ. of California, Berkeley, 1985.
32. SELINGER, P. G., AND ADIBA, M. E. Access path selection in distributed database management systems. In *Proceedings of the International Conference on Databases* (July 1980), 204–215.
33. STONEBRAKER, M., AND NEUHOLD, E. A distributed database version of INGRES. In *Proceedings of the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks* (1977).
34. WILLIAMS, R., ET AL. R*: An overview of the architecture. IBM Res. Rep. RJ3325, Dec. 1981.
35. WONG, E., AND YOUSSEFI, K. Decomposition—a strategy for query processing. *ACM Trans. Database Syst. 1*, 3 (Sept. 1976), 223–241.
36. WONG, E. Retrieving dispersed data from SDD-1: A system for distributed databases. In *Proceedings of the 3rd Berkeley Workshop on Distributed Data Management and Computer Networks* (May 1977), 217–235.
37. YU, C. T., AND CHANG, C. C. On the design of a query processing strategy in a distributed database environment. In *Proceedings of the ACM SIGMOD Database Week* (1983), ACM, New York, 30–39.

Received November 1983; revised August 1985; accepted September 1985