Query Optimization on Local Area Networks

ALAN R. HEVNER, O. QI WU, and S. BING YAO
University of Maryland

Local area networks are becoming widely used as the database communication framework for sophisticated information systems. Databases can be distributed among stations on a network to achieve the advantages of performance, reliability, availability, and modularity. Efficient distributed query optimization algorithms are presented here for two types of local area networks: *address ring networks* and *broadcast networks*. Optimal algorithms are designed for *simple queries*. Optimization principles from these algorithms guide the development of effective heuristic algorithms for general queries on both types of networks. Several examples illustrate distributed query processing on local area networks.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed databases*; C.2.5 [**Computer-Communication Networks**]: Local Networks—*access schemes*; H.2.4 [**Database Management**]: Systems—*distributed systems; query processing*

General terms: Algorithms, Design

Additional Key Words and Phrases: Distributed query optimization

1. INTRODUCTION

The advantages of decentralization in organizations are a driving force in the design of today's sophisticated information systems [13]. Advanced technical developments in microcomputer workstations and data communications have provided the means to achieve these advantages via *local area networks* (LANs) [15]. High bandwidth LANs provide a framework for tying together information resources within an organization.

An important factor in information system design is the distribution of databases on a network. Databases are distributed in order to achieve the advantages of performance, reliability, availability, and modularity [4]. In order to be effective, distributed database systems must include a user interface that is efficient while hiding the data distribution details from the user. A relational query refers only to the description of data that the user wants, not to where the data are located. The strategy that executes a query in the network will be determined by the query optimization algorithm. The process of retrieving data from different sites in the network is known as distributed query processing.

The performance of a distributed database system is critically dependent on the ability of the query optimization algorithm to derive efficient query processing

Authors' address: Database Systems Research Center, College of Business and Management, University of Maryland, College Park, MD 20742.

strategies. A considerable body of research on the distributed query optimization problem has been performed. (An excellent survey of this research area can be found in [4].) For the most part, the results of this previous research are applicable to distributed systems on point-to-point networks. Local area networks, however, exhibit quite different performance behaviors.

Query optimization on local area networks has received some attention. Gouda and Dayal studied the complexity of performing semijoin schedules on a bus [8]. They developed a method of executing semijoins in order to minimize data communication time. More recently, Sacco [14] and Wah and Lien [16] have analyzed the problems of query optimization on LANs. Wah and Lien proposed a heuristic algorithm for query optimization on a broadcast bus. Their algorithm is dynamic in that the next operation to be performed in the query strategy is not chosen until the previous operation is complete. The effectiveness of the algorithm is based on the ability to broadcast the status of the query execution to all sites involved in the query processing.

It is the purpose of this paper to propose query optimization algorithms especially designed for queries in local-area-network environments. Two types of LANs are studied: an address ring network and a broadcast network. The characteristics of each network are modeled in terms of a distributed database environment (Section 2).

In our previous research on distributed query optimization for point-to-point networks [1, 9, 11], we used the approach of restricting the query environment in order to find provably optimal query strategies. We extended these optimal algorithms into heuristic algorithms that minimized the *response time* and *total time* of general query execution. These algorithms have been implemented and found to be effective in a prototype distributed database system [7]. In this paper we use a similar research approach for LAN query optimization. For *simple queries*, in which a single, common joining attribute connects all query relations, we design optimal query strategies. Then, using the optimization principles in the optimal algorithms, we develop effective heuristic algorithms for both types of LANs in a general query environment.

2. DISTRIBUTED DATABASE SYSTEM MODEL

A distributed database system (DDBS) is characterized by the distribution of system resources: hardware, software, and data. For our analysis on local area networks, a DDBS is defined as a collection of sites, S_i, that are connected on a common, high-bandwidth local area network. Each site may contain a processing capability, a data storage capacity, or a combination of both. For example, different sites may be computer systems, personal workstations, database machines, or intelligent terminals in an integrated information system. Queries can only be processed at sites that contain a version of the distributed database management system.

The database is viewed logically in the relational data model [5]. The database is allocated across system nodes in units of relations (without loss of generality, relation fragments may be considered as relations for distribution). The relation distribution allows a general manner of redundancy. The only allocation constraint is that all data must be accessible from any system site. The distribution of data on the network is invisible to the user.

We assume that the following information about the relations is maintained in a distributed data dictionary. Each query processing site is assumed to contain a copy of the data dictionary.

For each relation R_i, $i = 1, 2, \ldots, m$,

n_i: number of tuples,
a_i: number of attributes,
s_i: size (e.g., in bytes).

For each attribute d_{ij}, $j = 1, 2, \ldots, a_i$ of relation R_i,

p_{ij}: attribute density (defined below),
w_{ij}: size (e.g., in bytes) of the data item in attribute d_{ij},
b_{ij}: projected size (e.g., in bytes) of attribute d_{ij} with no duplicate values.

The density, p_{ij}, of attribute d_{ij} is defined as the number of different values occurring in the attribute divided by the number of all possible values in the domain of the attribute. So, $0 \leq p_{ij} \leq 1$.

To process a query in a relational database, we only need the operations selection, projection, and join [6]. In a distributed database we may need to compute joins on relations that are located at different sites. Instead of computing these joins by moving relations on the network, it is beneficial first to reduce the sizes of the relations wherever possible by selections and projections. This phase of the query processing strategy is known as *initial local processing*. Once relations are reduced locally, the remainder of the query consists of intersite joins that require data transmission on the network and subsequent projections of join attributes not required for output.

The *semijoin* operation [2] has been found to be a beneficial technique for performing distributed join operations. A semijoin from relation R_i to relation R_k on attributes d_{ij} and d_{kl}, respectively, would be performed by projecting the unique attribute values of d_{ij} (with size b_{ij}) and transmitting them to the site containing R_k. Tuples of R_k are selected where the d_{kl} value matches one of the set of d_{ij} values. The *selectivity* of the semijoin is defined as the attribute density of the transmitted joining attribute, p_{ij}. By assuming a uniform distribution of values in R_k from the domain of d_{kl}, the parameters of relation R_k are then changed in the following way:

$$
\begin{aligned}
n'_k &\leftarrow n_k * p_{ij}, \\
s'_k &\leftarrow s_k * p_{ij}, \\
p'_{kl} &\leftarrow p_{kl} * p_{ij}, \\
b'_{kl} &\leftarrow b_{kl} * p_{ij}.
\end{aligned}
$$

The size and density of other attributes in relation R_k are also changed on the basis of their dependence on the joining attribute d_{kl}. These changes can be estimated using formulas that analyze dependencies among attributes [3, 9].

In our optimization analysis, data transmission costs are modeled as follows for the two types of local area networks.

Address Ring. An address ring using a token protocol allows one site to send a message to one or more other sites by placing the selected site addresses in the

message header. Data flows in one direction. Each site actively monitors the ring and accepts the message if its address is detected. The final addressed site reenters the token onto the ring. The cost of each data transmission is

$$t_a + u_{ij}c_a x, \tag{2.1}$$

where

t_a: average time (in seconds) for sending site to gain control of the token ring.

u_{ij}: distance (in units) on the ring between the sending site S_i and the final receiving site S_j. If a ring has n sites, then assuming sites are equidistant around the ring with one unit separation,

$$u_{ij} = \begin{cases} j - i & \text{if } j \geq i, \\ n + j - i & \text{if } j < i. \end{cases}$$

c_a: average time for each byte of data (in seconds/byte) to travel a unit distance on the ring.

x: amount of data (in bytes) transmitted.

Broadcast Network. A broadcast network can be either a bus or ring topology with various protocols. A message placed on the network is transmitted to all sites. Further processing of the message at each site will determine the actions the site performs (e.g., discard message, perform a semijoin). The cost of each data transmission is

$$t_b + c_b x, \tag{2.2}$$

where

t_b: average time (in seconds) for sending site to gain control of the network,

c_b: average time for each byte of data (in seconds/byte) to traverse the network,

x: amount of data (in bytes) transmitted.

In the subsequent sections we develop distributed query optimization algorithms for both local area network environments. The algorithms produce a query strategy to be executed on the distributed database system. The algorithms are *static* in that they base the optimization on the state of the system at one point in time. State variables should be updated regularly. For example, the variables t_a and t_b should be updated to reflect the present level of contention on the network. The principal advantage of static optimization is the ability to compile and store optimized query strategies for later execution. Queries are reoptimized if the current system state is sufficiently different from the state under which the current strategy was derived. Dynamic algorithms require optimization processing during execution of the query. A disadvantage of the static method is error propagation during the estimation of semijoin effects on size and selectivity parameters. These errors may lead to the derivation of a less beneficial query strategy [16].

Only one site at a time may control a local area network. Thus, a distributed query strategy consists of a sequence of data transmissions interspersed with

query processing at the sites. The query result will be formed at a specified result site, S_Δ. We denote two types of data transmissions. An *intermediate transmission* is a transmission in which one or more joining attributes are transmitted to one or more relations to effect semijoins. The intermediate transmission, $(d_{i1}, d_{i2}:R_j, R_k)$, represents the move of attributes d_{i1} and d_{i2} to the sites containing relations R_j and R_k. A *final transmission* is a transmission in which a relation is transmitted to the result site $(R_k:S_\Delta)$. An example distributed query strategy may be defined as an *ordered* set of intermediate and final transmissions,

$$\{(d_{11}:R_2), (d_{21}:R_3), (R_1:S_\Delta), (R_2:S_\Delta), (R_3:S_\Delta)\}.$$

In this example, attribute d_{11} is transmitted and a semijoin is performed on R_2. Then the reduced d_{21} is transmitted and a semijoin is performed on R_3. Finally, the reduced relations R_1, R_2, and R_3 are transmitted to the result site for final processing. Note that parallel data transmissions are not possible on a LAN; thus, *response time* minimization is equivalent to *total time* minimization.

For the purposes of the analysis in this paper, queries are assumed to be in disjunctive normal form [6]. Each conjunct subquery is optimized separately, and the union of the subquery results forms the query result. The selected data used in each subquery are determined before the query strategy is optimized by the algorithms presented in the next sections. We assume that the single most beneficial copy of redundant data is chosen for processing.

Simple Queries. A *simple query* is defined as a query in which each relation contains only a common joining attribute. This definition allows us to make the following observations:

(1) Since each query relation, R_i, has one attribute, $a_i = 1$, we will use a single subscript for simple query analysis (e.g., p_i, s_i).
(2) A relation contains no duplicate values, $s_i = b_i$.
(3) The size of a relation, s_i, is directly proportional to the density, p_i. Let v_i represent the total number of distinct values in the domain of the joining attribute. Then, $v_i * w_i * p_i = s_i$.
(4) The values for v_i and w_i are assumed to be constant over all simple query relations. Therefore,

$$p_i \leq p_j \quad \text{if and only if} \quad s_i \leq s_j,$$

and

$$p_i * s_j = p_j * s_i.$$

These observations will be used in the simple query proofs in the next section.

3. ADDRESS RING QUERY OPTIMIZATION

The topology of an address ring is illustrated in Figure 1. There are n sites on the ring labeled S_i, $i = 1, \ldots, n$. A data transmission can only travel in one direction (assume clockwise) on the ring. A transmission from S_i to S_j can be monitored and received by any site between the two sites.

In our analysis we use an *ordering relationship* among sites on a ring. We denote $S_i \ll S_j \ll S_k$ (i, j, and k distinct) when site S_j is between sites S_i and S_k

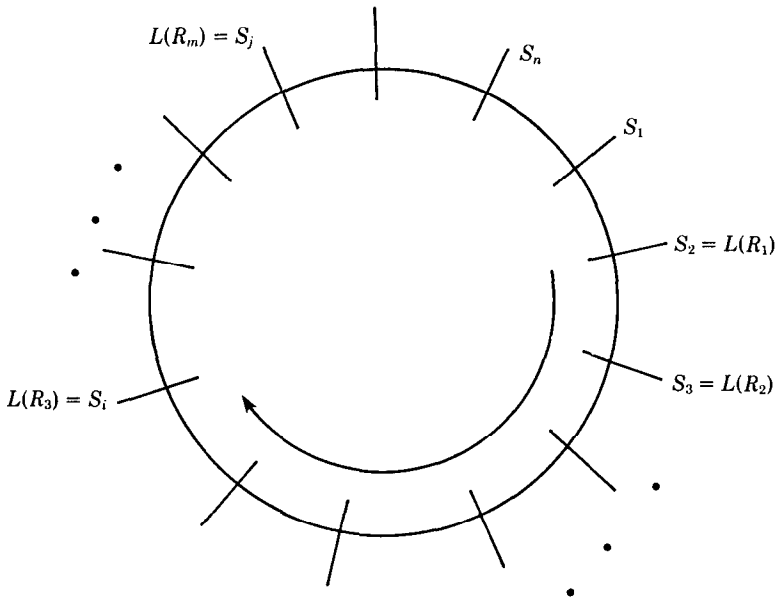

Fig. 1. An address ring.

and the path from S_i to S_j to S_k does not circumnavigate the ring. Thus, in Figure 1, the relationship $S_i \ll S_j \ll S_n \ll S_1$ holds.

Relations involved in a distributed query can be located at any site. Each site that has query data is considered to contain a single integrated relation. The relations at each site remain distinct, however. Without loss of generality, this assumption provides a distribution abstraction to the problem and simplifies the understanding of the following optimization algorithms. When necessary to identify the site location of a particular relation, we use a *location function*, $L(R_i) = S_k$, where R_i is located at site S_k. The inverse function, $L'(S_k) = R_i$ returns the integrated query relation at that site. In the following, both sites and relations are numbered sequentially clockwise, with S_1 and R_1 assigned arbitrarily.

3.1 Simple Query Optimization

A *legal strategy* for a query is one in which the correct query result is formed at the result site. A legal strategy must satisfy the following two properties.

Property 1. A legal strategy must include a data transmission from each site containing a query relation, except for the result site.

Property 2. A legal strategy must be connected, ending at one result site. In other words, data from all query relations must be either transmitted eventually to the result site or used in a semijoin that selects the data to be transmitted eventually to the result site.

For simple query analysis we make two straightforward assumptions on legal strategies. First, if one query relation has a density less than 1.0, then relations with a density of 1.0 need not be included in the query. It can be assumed that all such relations are identified and eliminated from the query execution strategy.

Thus, we assume that $0 \leq p_i < 1$, for all query relations, R_i, $i = 1, \ldots, m$. The second assumption restricts the range of sites on which simple query processing can be performed. Only sites containing query relations and the result site are considered as processing sites. If all sites have equivalent processing capabilities, it is easily seen that the use of sites with no resident query data provides no cost benefit on LANs over the use of sites containing query data. In general, however, this may not be true (e.g., networks with varying processor speeds [12]).

For analysis, we represent a query execution strategy with an acyclic directed graph called a query execution graph (QEG). Given a query strategy, it is straightforward to construct a QEG. Starting with S_Δ as the root, draw an arc for each direct transmission into S_Δ. From the end to the beginning of the strategy, then, add directed arcs based on each data transmission into the connected graph. Since the query strategy is by definition ordered, cycles are not possible. Any transmissions that do not result in a connected arc are extraneous, since the data does not end up at the result site. Note that a single transmission $(R_i:R_j, R_k, \ldots)$ will produce a separate arc from R_i into each receiver relation. Also note that this construction does not preclude multiple transmissions to and from any query relation.

For example, consider the following strategy for a simple query involving four relations,

$$\{(R_1:R_2)(R_2:R_3, R_4)(R_3:S_\Delta)(R_4:S_\Delta)\}.$$

The resulting QEG is shown in Figure 2.

Transmission costs can be viewed on a QEG in the following way. Each sending node represents a cost of t_a for accessing the LAN. The cost of all arcs starting from a node is represented by the maximum of the costs of each arc taken individually (i.e., $\max(c_a x u_{ij})$, where x is the amount of data transmitted and u_{ij} is the distance from S_i to S_j.) On the basis of Properties 1 and 2 above, a QEG for a legal strategy must include transmissions from all query relations (except a relation at the result site), and the graph must be connected with S_Δ as the directed root. In the following, we only consider legal strategies for simple queries.

A *linear strategy* is defined as query strategy that is represented by a linear QEG.

LEMMA 3.1. *An arbitrary query strategy for a simple query on an address ring can be transformed into a linear strategy with lesser or equal transmission cost.*

PROOF. By definition, any legal query strategy can be represented by a directed acyclic graph ending at one node, S_Δ. Consider the cost of all transmissions coming into an arbitrary node in the graph in Figure 3.

Let $L(R_{i_a}) = S_{k_a}$. For the special case where the receiving node is S_Δ, consider R_{i_j} to be a dummy relation residing at S_Δ. The relations in the graph segment must be unique, else one of the duplicate arcs can be eliminated from the graph. The site subscripts can be ordered as

$$S_{k_1} \ll S_{k_2} \ll \cdots \ll S_{k_{j-1}} \ll S_{k_j}.$$

Clearly this can be done, since if the relations are unique, then by definition the sites are unique also. Figure 4 shows the notation on the ring.

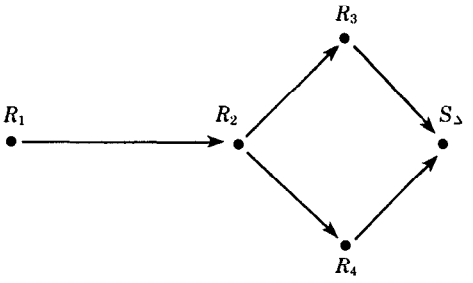

Fig. 2. A query execution graph (QEG).

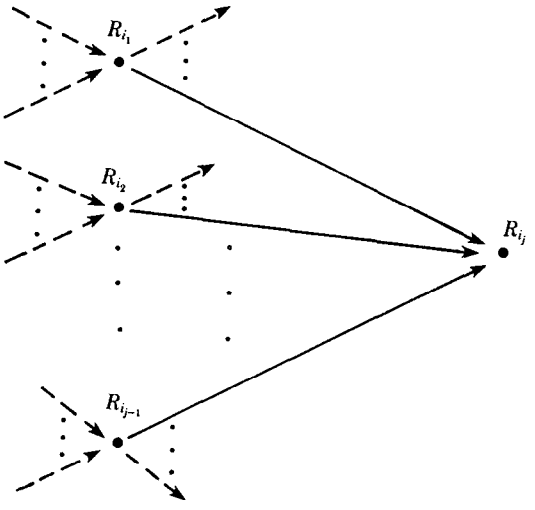

Fig. 3. Transmissions to a graph node.

The cost of this graph segment is

$$\sum_{\alpha=1}^{j-1} (t_\alpha + c_\alpha x_{i_\alpha} u_{k_\alpha k_j}), \tag{3.1}$$

where x_{i_α} is the size of relation R_{i_α} in the transmission and $u_{k_\alpha k_j}$ is the unit length from site S_{k_α} to site S_{k_j}.

The above segment can be transformed into a linear segment with lesser or equal cost. Consider the linear segment shown in Figure 5. The cost of this segment is

$$\sum_{\alpha=1}^{j-1} (t_\alpha + c_\alpha x'_{i_\alpha} u_{k_\alpha k_{\alpha+1}}), \tag{3.2}$$

where x'_{i_α} is the size of R_{i_α} after the potential size reductions from the previous relations in the linear segment, R_{i_β}, where $\beta < \alpha$. Thus, $x'_{i_\alpha} \leq x_{i_\alpha}$. It is also clearly seen that $u_{k_\alpha k_{\alpha+1}} \leq u_{k_\alpha k_j}$ because of the relation ordering (see Figure 4). Since all other factors are constant, the cost of the segment in Figure 5 (expression (3.2)) is less than or equal to the cost of the segment in Figure 3 (expression (3.1)).

This transformation will not increase the transmission costs in the remainder of the QEG. All incoming transmissions into the R_{i_α} relations ($\alpha = 1, 2, \ldots, j-1$) are not affected. Outgoing transmissions may be affected in that

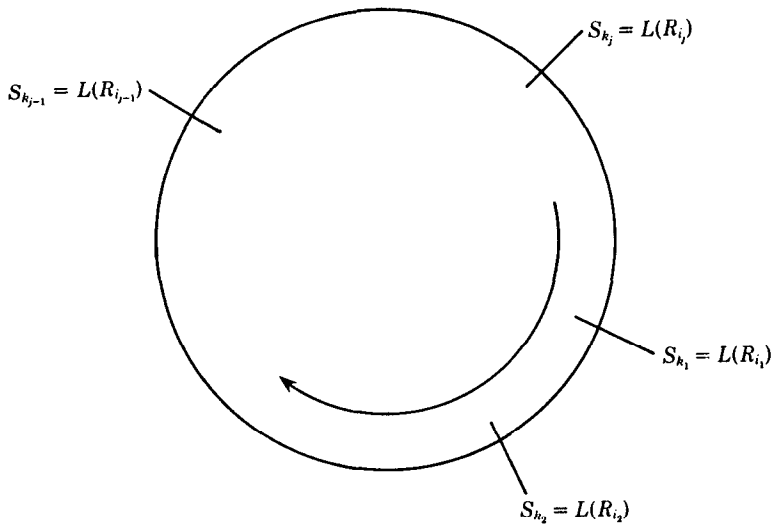

Fig. 4. State of the address ring.

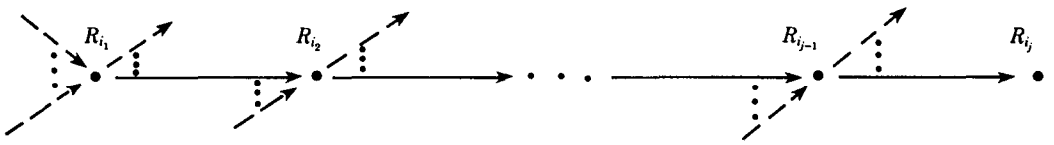

Fig. 5. Linear graph segment.

the size of the R_{i_α} may be further reduced by the selectivities of the relations in the preceding linear segment. Thus, the transmission cost of outgoing arcs may be less. The size and selectivity effects of the two segments at R_{i_j} remain the same; outgoing arcs are not affected.

The linear transformation of any node in the QEG with two or more incoming arcs produces a new QEG with lesser or equal cost. By iteratively performing such linear transformations on all nodes in a QEG, the final result will be a *linear strategy* for a simple query with lesser or equal cost than the original query strategy. Q.E.D.

We define a *nonredundant linear strategy* as a linear strategy in which each relation is transmitted at most once.

LEMMA 3.2. *A linear strategy for a simple query on an address ring can be transformed into a nonredundant linear strategy with lesser or equal transmission cost.*

PROOF. Consider a linear strategy as in Figure 6. Assume $i_\alpha = i_\beta$. Since the selectivity of R_{i_α} is already included in the strategy, the inclusion of the R_{i_β} transmission adds no further size reductions. Thus, an additional transmission of a relation adds no cost benefit to the strategy but adds, at the least, a cost of t_a to access the ring.

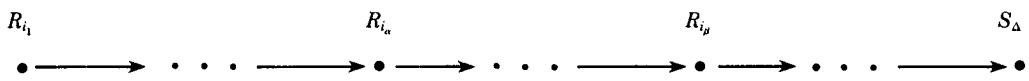

Fig. 6. Redundant linear strategy.

By iteratively eliminating all cases of duplicate transmission of the same relation, the resulting linear strategy has a cost lesser than or equal to that of the strategy in Figure 6. Q.E.D.

We define a *serial strategy* for a simple query on an address ring as one in which every query relation is transmitted exactly once (except, perhaps, for a relation at the result site which may not be transmitted), and the transmission order of the relations is based on the order of the sites on the ring. If $S_{k_1} \ll S_{k_2} \ll \cdots \ll S_{k_\alpha}$, then the serial strategy would be

$$\{(R_{i_1}, R_{i_2}), \ldots, (R_{i_\alpha}, S_\Delta)\},$$

where $L(R_{i_j}) = S_{k_j}$ and $\alpha = m$ or $m - 1$, depending on whether a relation at S_Δ is in the strategy or not.

LEMMA 3.3. *A nonredundant linear strategy for a simple query on an address ring can be transformed into a serial strategy with lesser or equal transmission cost.*

PROOF. Consider the nonredundant linear strategy in Figure 7a, where $S_{k_\alpha} \ll S_{k_\beta} \ll S_{k_\gamma}$ (see Figure 8). The segments L_1 and L_2 represent the beginning and ending portions of the strategy. Let ρ be the accumulated selectivity from the L_1 segment. The transmission cost for this strategy is

$$\begin{aligned}\text{Cost}(L_1) + (t_a + c_a\rho s_{i_\alpha} u_{k_\alpha k_\gamma}) + (t_a + c_a\rho p_{i_\alpha} s_{i_\gamma} u_{k_\gamma k_\beta}) \\ + (t_a + c_a\rho p_{i_\alpha} p_{i_\gamma} s_{i_\beta} u_{k_\beta k_\delta}) + \text{Cost}(L_2).\end{aligned} \tag{3.3}$$

A lesser or equal cost can be gained by transforming the segment from R_{i_α} to R_{i_δ} into the strategy as found in Figure 7b. The cost of the new strategy is

$$\begin{aligned}\text{Cost}(L_1) + (t_a + c_a\rho s_{i_\alpha} u_{k_\alpha k_\beta}) + (t_a + c_a\rho p_{i_\alpha} s_{i_\beta} u_{k_\beta k_\gamma}) \\ + (t_a + c_a\rho p_{i_\alpha} p_{i_\beta} s_{i_\gamma} u_{k_\gamma k_\delta}) + \text{Cost}(L_2).\end{aligned} \tag{3.4}$$

It can be seen that expression (3.4) is less than or equal to (3.3) in the following way. In comparing the two expressions, first eliminate the common terms $\text{Cost}(L_1)$, $\text{Cost}(L_2)$, and $3t_a$; then divide all terms by the common factor, $c_a\rho$. Thus, we need to show that

$$\begin{aligned}s_{i_\alpha} u_{k_\alpha k_\gamma} + p_{i_\alpha} s_{i_\gamma} u_{k_\gamma k_\beta} + p_{i_\alpha} p_{i_\gamma} s_{i_\beta} u_{k_\beta k_\delta} \\ \geq s_{i_\alpha} u_{k_\alpha k_\beta} + p_{i_\alpha} s_{i_\beta} u_{k_\beta k_\gamma} + p_{i_\alpha} p_{i_\beta} s_{i_\gamma} u_{k_\gamma k_\delta}.\end{aligned} \tag{3.5}$$

Since $S_{k_\alpha} \ll S_{k_\beta} \ll S_{k_\gamma}$, we have $u_{k_\alpha k_\gamma} = u_{k_\alpha k_\beta} + u_{k_\beta k_\gamma}$, and expression (3.5) becomes

$$s_{i_\alpha} u_{k_\beta k_\gamma} + p_{i_\alpha} s_{i_\gamma} u_{k_\gamma k_\beta} + p_{i_\alpha} p_{i_\gamma} s_{i_\beta} u_{k_\beta k_\delta} \geq p_{i_\alpha} s_{i_\beta} u_{k_\beta k_\gamma} + p_{i_\alpha} p_{i_\beta} s_{i_\gamma} u_{k_\gamma k_\delta}. \tag{3.6}$$

By the definition of a simple query, $s_{i_\alpha} \geq p_{i_\alpha} s_{i_\beta}$, since a join result is always smaller than or equal to either of the join relations. Therefore, in eq. (3.6),

$$s_{i_\alpha} u_{k_\beta k_\gamma} \geq p_{i_\alpha} s_{i_\beta} u_{k_\beta k_\gamma}. \tag{3.7}$$

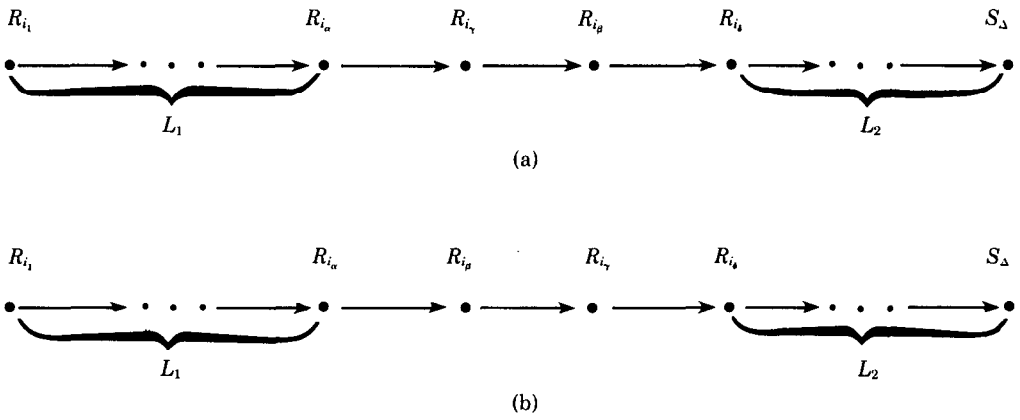

Fig. 7. Serial strategy transformation.

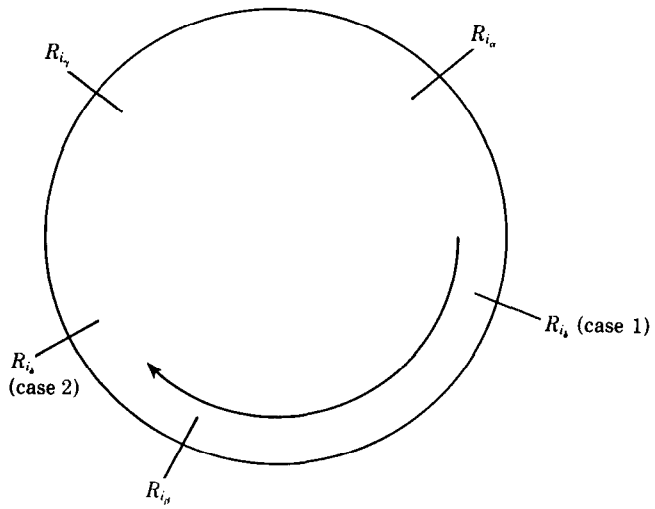

Fig. 8. Address ring for Lemma 3.3.

To see that

$$p_{i_\alpha}s_{i_\gamma}u_{k_\gamma k_\beta} + p_{i_\alpha}p_{i_\gamma}s_{i_\beta}u_{k_\beta k_\delta} \geq p_{i_\alpha}p_{i_\beta}s_{i_\gamma}u_{k_\gamma k_\delta}, \tag{3.8}$$

first recognize that $p_{i_\alpha}p_{i_\gamma}s_{i_\beta} = p_{i_\alpha}p_{i_\beta}s_{i_\gamma}$, since the join operation for simple queries is commutative.

Consider two cases, depending on where R_{i_δ} is located on the ring (see Figure 8).

Case 1. $S_{k_\gamma} \ll S_{k_\delta} \ll S_{k_\beta}$. For this case, $u_{k_\gamma k_\beta} \geq u_{k_\gamma k_\delta}$, and eq. (3.8) is clearly true, since $p_{i_\alpha}s_{i_\gamma} \geq p_{i_\alpha}p_{i_\beta}s_{i_\gamma}$.

Case 2. $S_{k_\beta} \ll S_{k_\delta} \ll S_{k_\gamma}$. For this case, $u_{k_\gamma k_\delta} = u_{k_\gamma k_\beta} + u_{k_\beta k_\delta}$ and from eq. (3.8),

$$p_{i_\alpha}s_{i_\gamma}u_{k_\gamma k_\beta} + p_{i_\alpha}p_{i_\gamma}s_{i_\beta}u_{k_\beta k_\delta} \geq p_{i_\alpha}p_{i_\beta}s_{i_\gamma}u_{k_\gamma k_\beta} + p_{i_\alpha}p_{i_\beta}s_{i_\gamma}u_{k_\beta k_\delta}. \tag{3.9}$$

Since $p_{i_\alpha}s_{i_\gamma} \geq p_{i_\alpha}p_{i_\beta}s_{i_\gamma}$, this proves that eq. (3.8) is true for this case.

By iterating this linear segment transformation, any nonredundant linear strategy is transformed into a serial strategy with lesser or equal cost. Q.E.D.

The above three lemmas allow the following theorem to be proved.

THEOREM 3.1. *A minimum-cost strategy for a simple query on an address ring can be found by comparing the costs of all serial strategies.*

PROOF. Given an arbitrary legal strategy for a simple query on an address ring:

(1) Lemma 3.1 proves that the strategy can be transformed into a linear strategy with lesser or equal cost.
(2) Lemma 3.2 proves that a linear strategy can be transformed into a nonredundant linear strategy with lesser or equal cost.
(3) Lemma 3.3 proves that a nonredundant linear strategy can be transformed into a serial strategy with lesser or equal cost.

Thus, any legal strategy can be transformed into a legal serial strategy with lesser or equal cost. By testing all possible serial strategies a minimum-cost query execution strategy for simple queries can be found. Q.E.D.

From this theorem we obtain Algorithm AR-SERIAL, an optimization algorithm for simple queries that generates optimal strategies.

Algorithm AR-SERIAL

1. Do all initial processing. Number the query relations R_i, $i = 1, \ldots, m$. Assign R_1 arbitrarily, and number sequentially clockwise around the ring.
2. For each query relation, R_i, $i = 1, \ldots, m$, calculate the transmission cost of a serial strategy starting at R_i and ending with all data at S_Δ.
3. For each query relation, R_i, $i = 1, \ldots, m$, if a query relation is at site S_Δ, calculate the transmission cost of a serial strategy from R_i that does not include a transmission to or from the relation at S_Δ.
4. Select the minimum-cost strategy generated in steps 2 and 3.

End of Algorithm AR-SERIAL.

The complexity of Algorithm AR-SERIAL is $O(m^2)$. At most, the costs of $2m - 1$ strategies are compared. The computation of each strategy's cost requires the summation of m transmissions. The algorithm is illustrated on the following simple query example.

Example 3.1. Figure 9 shows an address ring with 15 sites located equidistantly on the network. A simple query is placed on the network with result site S_7. After local processing, four query relations remain with state parameters shown in Table I. The data transmission cost parameters are

$$t_a = 2.0 \text{ seconds}, \qquad c_a = 10^{-3} \text{ second/byte},$$
$$u_{ij} = \begin{cases} j - i & \text{if } j \geq i, \\ 15 + j - i & \text{if } j < i, \end{cases}$$

where the distance from one site to an adjacent site is one unit.

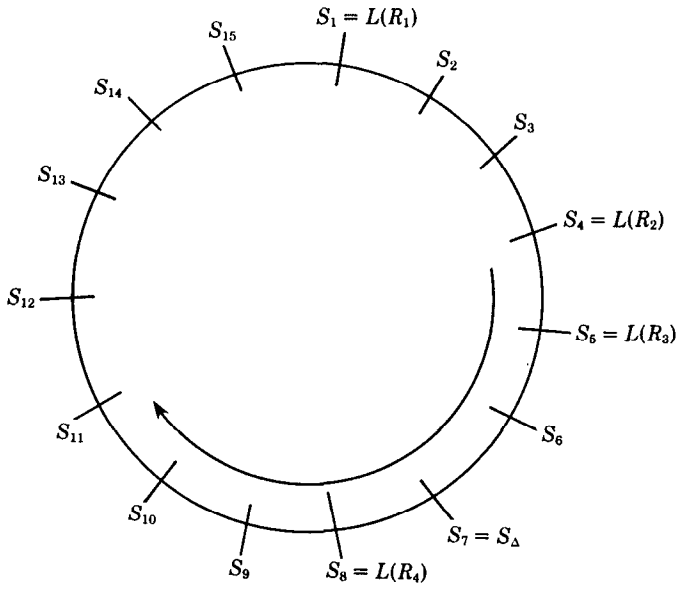

Fig. 9. Address ring simple query example.

Table I. Simple Query Parameters (Address Ring)

Relation: R_i	R_1	R_2	R_3	R_4
Location: S_i	S_1	S_4	S_5	S_8
Size: s_i (bytes)	8000	5000	8000	4000
Selectivity: p_i	0.8	0.5	0.8	0.4

Since the result site does not contain a query relation, there are four serial strategies to be tested.

Strategy 1: $\{(R_1:R_2), (R_2:R_3), (R_3:R_4), (R_4:S_7)\}$.

Transmission cost = 26.0 + 6.0 + 11.6 + 19.9 = 63.5 seconds.

Strategy 2: $\{(R_2:R_3), (R_3:R_4), (R_4:R_1), (R_1:S_7)\}$.

Transmission cost = 7.0 + 14.0 + 14.8 + 9.7 = 45.5 seconds.

Strategy 3: $\{(R_3:R_4), (R_4:R_1), (R_1:R_2), (R_2:S_7)\}$.

Transmission cost = 26.0 + 27.6 + 9.7 + 5.8 = 69.1 seconds.

Strategy 4: $\{(R_4:R_1), (R_1:R_2), (R_2:R_3), (R_3:S_7)\}$.

Transmission cost = 34.0 + 11.6 + 3.6 + 4.2 = 53.4 seconds.

The minimum cost strategy is strategy 2.

3.2 General Query Optimization

A *general query* is characterized by each query relation having an arbitrary number of output attributes and one or more joining attributes that connect it

with the other relations. Let δ represent the number of distinct joining attributes in the query. Therefore, a relation can be reduced in size by semijoins on different joining attributes.

The general query optimization problem on an address ring is NP-hard [10]. Thus, we present a heuristic optimization algorithm for general queries. Algorithm AR-GENERAL is based on extending the serial strategy principles found optimal for simple queries.

In an address ring, a single intermediate transmission can send data to several sites for semijoins. It may or may not be cost beneficial to perform semijoin processing at each of these sites, however, because of local processing costs. A general algorithm must include a method for selecting processing sites for intermediate transmissions. In Algorithm AR-GENERAL we consider local processing costs as defined in the following.

Definition. A *cost-beneficial semijoin* for relation R_k at site S_γ is one in which the cost of performing the semijoin, SJ-cost(R_k), is less than the reduction in transmission cost for the reduced relation to the result site, S_Δ. Thus, if

$$\text{SJ-cost}(R_k) < c_a(s_k - s_k')u_{\gamma\Delta}, \tag{3.10}$$

where s_k' is the reduced size of R_k after the semijoin, then S_γ is selected as a processing site for that intermediate transmission.

Definition. A *cost-beneficial intermediate transmission* is one in which the transmission cost of sending data for semijoin operations, $t_a + c_a x u_{\alpha\beta}$ (where x is the size of the data), is less than the summation of the benefits gained by all beneficial semijoins effected by the transmission (eq. 3.10).

Algorithm AR-GENERAL

1. Do all initial local processing. Let Q represent the set of all query relations remaining. Let QS (initially empty) represent the query strategy. Within each relation, order joining attributes d_{ij} by ascending p_{ij} value, $j = 1, \ldots, \delta_i$ = the number of joining attributes in R_i.
2. For each relation R_i in Q find the most cost-beneficial intermediate transmission from R_i as follows:
   a. Perform step 2b for data transmissions of (d_{i1}), (d_{i1}, d_{i2}), $\ldots$, $(d_{i1}, d_{i2}, \ldots, d_{i\delta_i})$, where the attribute ordering is as found in step 1.
   b. For each relation $R_j \neq R_i$, where all attributes of R_j have not been transmitted in QS, calculate the cost benefit of an intermediate data transmission from R_i to R_j, where all sites between R_i and R_j can receive the data. Processing is only performed at sites that have a cost-beneficial semijoin.
   c. From 2b, select the most beneficial intermediate transmission for R_i, and label it R_i^*.
3. From 2, select the most beneficial intermediate transmission among the R_i^* candidates. Add it to QS as the next transmission, and calculate the resulting database state. If no intermediate transmissions are beneficial, go to 5.
4. Eliminate the selected sending relation in 3 from Q. If Q is not empty, go to 2.
5. Add final transmissions to QS from all query relations to the final site. Do not include relations whose data have been completely transmitted as part of an intermediate transmission.

End of Algorithm AR-GENERAL.

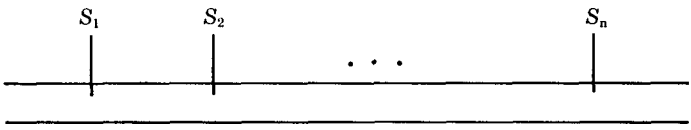

Fig. 10. Broadcast network.

The complexity of Algorithm AR-GENERAL is, in the worst-case analysis, $O(m^4\delta)$. In step 2 each relation in Q (worst-case size of m) is tested for cost-beneficial transmissions to the other (possibly, $m - 1$) relations. The testing is done for up to δ groupings of join attributes. The calculation of each strategy cost requires the summation of, in the worst case, $m - 1$ semijoin benefits. The algorithm iterates step 2, in the worst case, m times.

The principal optimization technique in the algorithm involves analyzing serial transmissions of join attribute groupings. The groupings are treated as simple queries and tested for cost-beneficial transmissions to other relations on the ring. When no further beneficial intermediate transmissions can be found, the essential final transmissions are added to the final execution strategy. Section 5 illustrates general query optimization on an address ring using Algorithm AR-GENERAL.

4. BROADCAST NETWORK QUERY OPTIMIZATION

In a broadcast network (Figure 10), multiple processors are connected by a common communication channel. Any site can place a message on the network. The transmission is broadcast so that all other sites can receive it. Ordering and distance among sites are irrelevant for transmission cost.

4.1 Simple Query Optimization

A simple query strategy on broadcast network will be represented as an ordered set of transmissions,

$$\{(R_{i_1}), (R_{i_2}), \ldots, (R_{i_a})\},$$

where each relation is broadcast to all other sites. Information within the message will tell each site what to do with the data transmitted. Thus, a site that contains a query relation may or may not perform a local semijoin with an incoming relation. Note that since our optimization criterion only involves minimizing transmission cost, the selection of processing sites does not affect the following simple query analysis. However, we will discuss the selection of sites for local processing and incorporate a selection step into the query optimization algorithms.

The definition of a *legal strategy* is the same as for address rings. The strategy must contain a transmission from every query relation site (except the result site), and the strategy must be connected. For broadcast networks, we number the query relations, R_i, $i = 1, \ldots, m$, based on the density value of the common joining attribute,

$$p_1 \leq p_2 \leq \cdots \leq p_m.$$

Relations can be located arbitrarily on the network.

On a broadcast network, any legal strategy for a simple query can be considered a *linear strategy*, in which each sending relation can be received at all other network sites. We define a *nonredundant linear strategy* as a linear strategy in which each relation is transmitted at most once.

LEMMA 4.1. *A linear strategy for a simple query on a broadcast network can be transformed into a nonredundant linear strategy with lesser or equal transmission cost.*

PROOF. Consider a linear strategy in which R_j is transmitted more than once:

$$\{(R_i), \ldots, (R_j), \ldots, (R_j), \ldots, (R_k)\}.$$

Since the selectivity of R_j is already included in the strategy, the inclusion of the second R_j transmission adds no further size reduction. Thus, an additional transmission of a relation adds no cost benefit, but adds, at the least, a cost of t_b to access the network.

By iteratively eliminating all cases of duplicate transmission of a relation, the resulting linear strategy has cost lesser than or equal to that of the original strategy. Q.E.D.

We define a *serial strategy* for a simple query on a broadcast network as a nonredundant linear strategy in which each relation (except, perhaps, for a relation at the result site) is transmitted in order of increasing selectivity value. Thus, if a relation is at the result site, $L(R_k) = S_\Delta$, then there are two possible serial strategies:

Strategy 1: $\{(R_1), (R_2), \ldots, (R_m)\}$.

Strategy 2: $\{(R_1), \ldots, (R_{k-1}), (R_{k+1}), \ldots, (R_m)\}$.

If no relation is at the result site, then strategy 1 is the only serial strategy.

LEMMA 4.2. *A nonredundant linear strategy for a simple query on a broadcast network can be transformed into a serial strategy with lesser or equal transmission cost.*

PROOF. Consider the nonredundant linear strategy where $p_\alpha \leq p_\beta$:

$$\{(R_i), \ldots, (R_\beta), (R_\alpha), \ldots, (R_k)\}.$$

Let L_1 represent the beginning of the strategy until the R_β transmission, and let L_2 represent the ending of the strategy after the R_α transmission. Let ρ be the accumulated selectivity from the L_1 segment. The transmission cost for this strategy is

$$\text{Cost}(L_1) + (t_b + c_b \rho s_\beta) + (t_b + c_b \rho p_\beta s_\alpha) + \text{Cost}(L_2). \tag{4.1}$$

A lesser or equal cost can be gained by reversing the transmission from R_β to R_α:

$$\{(R_i), \ldots, (R_\alpha), (R_\beta), \ldots, (R_k)\}.$$

The cost of the transformed strategy is

$$\text{Cost}(L_1) + (t_b + c_b \rho s_\alpha) + (t_b + c_b \rho p_\alpha s_\beta) + \text{Cost}(L_2). \tag{4.2}$$

It can be seen that expression (4.2) is less than or equal to (4.1) in the following way. In comparing the two equations, first eliminate common terms Cost(L_1), Cost(L_2), and $2t_b$; then divide all terms by the common factor, $c_b\rho$. Thus, we need to show that

$$s_\beta + p_\beta s_\alpha \geq s_\alpha + p_\alpha s_\beta. \tag{4.3}$$

Equation (4.3) is clearly true, since $s_\beta \geq s_\alpha$ is given and $p_\beta s_\alpha = p_\alpha s_\beta$ by the definition of a simple query.

It can also be seen that this transformation in no way affects the size and selectivity changes brought about by semijoins in segments L_1 or L_2. By iterating this linear transformation, any nonredundant linear strategy is transformed into a serial strategy with lesser or equal cost. Q.E.D.

These two lemmas allow the following theorem to be proved.

THEOREM 4.1. *A minimum-cost strategy for a simple query on a broadcast network can be found by comparing the costs of the two possible serial strategies.*

PROOF. Given an arbitrary legal strategy for a simple query on a broadcast network:

(1) Lemma 4.1 proves that the strategy can be transformed into a nonredundant linear strategy with lesser or equal cost.
(2) Lemma 4.2 proves that a nonredundant linear strategy can be transformed into a serial strategy with lesser or equal cost.

Thus, any legal strategy can be transformed into a legal serial strategy with lesser or equal cost. By testing both possible serial strategies a minimum-cost query execution strategy for simple queries can be found. Q.E.D.

From this theorem we obtain Algorithm BN-SERIAL, an optimization algorithm for simple queries that generates optimal strategies.

Algorithm BN-SERIAL

1. Do all initial processing. Number the query relations R_i, $i = 1, \ldots, m$ on the basis of increasing density value, $p_1 \leq p_2 \leq \cdots \leq p_m$.
2. Calculate the cost of the serial strategy that contains all relations in order (strategy 1).
3. If a relation is at the result site, calculate the cost of the serial strategy that does not include this relation (strategy 2).
4. Select the minimum-cost strategy in steps 2 and 3.

End of Algorithm BN-SERIAL.

The complexity of Algorithm BN-SERIAL is $O(m)$, since the calculation of a strategy cost requires the summation of m transmissions.

Each relation transmission in the simple query strategy sends data to all other relations. It can easily be observed that it is only necessary for the next relation in serial order to actually perform the semijoin. Since each transmission carries the accumulated selectivities of all previous relations in the strategy, the query relations need only perform one semijoin in the strategy. This is represented by

Table II. Simple Query Parameters (Broadcast Network)

Relation: R_i	R_1	R_2	R_3	R_4
Size: s_i (bytes)	3000	5000	8000	9000
Selectivity: p_i	0.3	0.5	0.8	0.9

intermediate transmission $(R_1 : R_2)$, where the right side of a transmission denotes the relations to perform the semijoin.

The algorithm is illustrated by the following simple query example.

Example 4.1. A broadcast network must execute a simple query that, after local processing, requires data from four relations at separate sites. The relations are ordered on the basis of their selectivities. The relation parameters are shown in Table II.

Let R_2 be stored at the result site. The data transmission cost parameters are

$$t_b = 3.0 \text{ seconds}, \qquad c_b = 5 * 10^{-3} \text{ second/byte}.$$

The two strategies to be tested are

Strategy 1: $\{(R_1 : R_2), (R_2 : R_3), (R_3 : R_4), (R_4 : S_\Delta)\}$.

Transmission cost = 18.0 + 10.5 + 9.0 + 8.4 = 45.9 seconds.

Strategy 2: $\{(R_1 : R_3), (R_3 : R_4), (R_4 : S_\Delta)\}$.

Transmission cost = 18.0 + 15.0 + 13.8 = 46.8 seconds.

The minimum cost strategy is strategy 1. Note that the algorithm is responsive to changes in the network state. For example, if the network access cost, t_b, is changed from 3.0 seconds to 6.0 seconds to show increased contention, the algorithm would find the strategy costs to be Cost(strategy 1) = 57.9 seconds and Cost(strategy 2) = 55.8 seconds. Thus, the second strategy, requiring fewer transmissions, would be selected on the basis of increased network access time.

4.2 General Query Optimization

Similarly to the address ring, the general query optimization problem on a broadcast network is NP-hard [10]. The following heuristic optimization algorithm for general queries is developed using the same definitions, with modified cost formulas, for *cost beneficial semijoins* and *cost beneficial intermediate transmissions* as presented in Section 3.2 for address rings.

Algorithm BN-GENERAL

1. Do all initial local processing. Let Q represent the set of all query relations remaining. Let QS (initially empty) represent the query strategy. Within each relation, order joining attributes d_{ij} by ascending p_{ij} value, $j = 1, \ldots, \delta_i$ = the number of joining attributes in R_i.
2. For each relation R_i in Q find the most cost-beneficial intermediate transmission from R_i as follows:
   a. Perform step 2b for data transmissions of (d_{i1}), (d_{i1}, d_{i2}), $\ldots$, $(d_{i1}, d_{i2}, \ldots, d_{i\delta_i})$, where the attribute ordering is as found in step 1.

b. Calculate the cost benefit of the intermediate transmission. Processing is only performed at sites that have a cost-beneficial semijoin.

c. From 2b select the most cost-beneficial intermediate transmission for R_i and label it R_i^*.

3. From 2 select the most cost-beneficial intermediate transmission among the R_i^* candidates. Add it to QS as the next transmission, and calculate the resulting database state. If no intermediate transmissions are beneficial, go to 5.

4. Eliminate the selected sending relation in 3 from Q. If Q is not empty, go to 2.

5. Add final transmissions to QS from all query relations. Do not include relations whose data have been completely transmitted as part of an intermediate transmission.

End of Algorithm BN-GENERAL.

The complexity of Algorithm BN-GENERAL is, in the worst-case analysis, $O(m^3\delta)$. In step 2, each relation in Q (worst-case size of m) is tested for cost-beneficial transmissions. The testing is done for up to δ groupings of join attributes. The calculation of each transmission cost requires the summation of, in the worst case, $m - 1$ semijoin benefits. The algorithm iterates step 2, in the worst case, m times.

The similarities between the algorithms for the broadcast network and the address ring are clear. In both algorithms the principle optimization technique is to group join attribute data in each query relation and find the best serial transmission strategy for the group that provides the greatest cost benefit on the network. Hill-climbing optimization selects the transmission with the greatest benefit, the database state is recalculated, and the process is reiterated until no more beneficial intermediate transmissions can be found. The necessary final transmissions are added to the strategy to complete the algorithm. An example of optimization processing using Algorithm BN-GENERAL is presented in the next section.

5. A GENERAL QUERY EXAMPLE

Consider the following organizational database:

R_1: Supplier(S#, sname, saddress)
R_2: S-P-J(S#, P#, J#)
R_3: Warehouse(W#, S#, P#, qty)

Suppliers (S#) supply parts (P#) that are stored in warehouses (W#). The parts are used in projects (J#). In a ten-site local area network, each of the relations is stored at a separate site. The query represented in Figure 11 is entered into the LAN at site S_5. The query can be stated: "List the S# and sname of all suppliers who supply parts to project 5, along with the parts they supply and the total quantity of those parts."

We now illustrate how this general distributed query would be optimized on an address ring with Algorithm AR-GENERAL and on a broadcast network with Algorithm BN-GENERAL. Both optimization algorithms start (step 1) by performing all local processing at the relation sites. Table III contains selected query parameter values after this initial processing. The join attributes for each relation

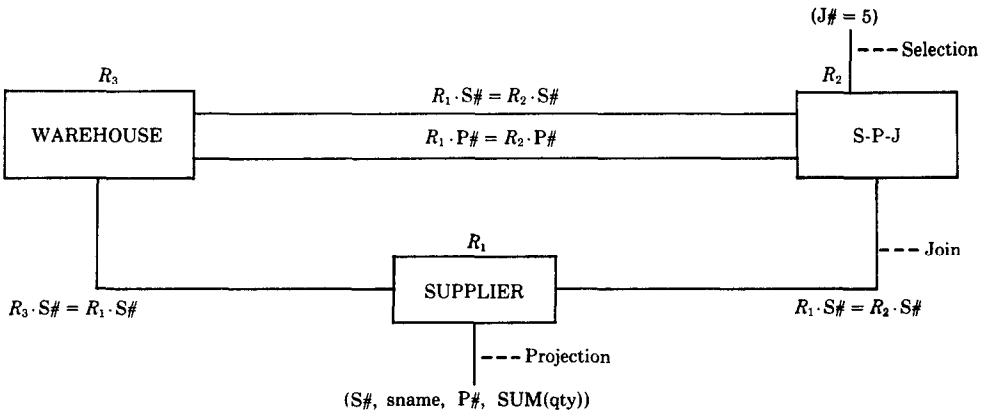

Fig. 11. General query example.

Table III. General Query Parameters

Relation: R_i	Location: S_i	Size: s_i (bytes)	Join attribute d_{i1} = P#		Join attribute d_{i2} = S#	
			b_{i1} (bytes)	p_{i1}	b_{i2} (bytes)	p_{i2}
R_1: Supplier	S_2	12000	—	—	2700	0.9
R_2: S-P-J	S_6	9000	2000	0.8	1500	0.5
R_3: Warehouse	S_9	4000	500	0.2	1200	0.4

are ordered on the basis of selectivity values. Note that attribute d_{22} will be ordered before d_{21} for relation R_2.

5.1 Address Ring Optimization

The cost parameters on the address ring are

$$t_a = 2.0 \text{ seconds}, \quad c_a = 10^{-3} \text{ second/byte},$$
$$u_{ij} = \begin{cases} j - i & \text{if } j \geq i, \\ 10 + j - i & \text{if } j < i. \end{cases}$$

Recording each step of Algorithm AR-GENERAL, the optimization proceeds as follows.

Step 1. Local processing and attribute ordering are done.

$Q = \{R_1, R_2, R_3\}$ and QS is empty.

Step 2—*Pass* 1. Find the best intermediate transmission from each relation in Q.

R_1: Supplier

(1) Test the intermediate transmission of d_{12} to R_2. The benefit is the cost of transmitting R_2 without this semijoin minus the cost of transmitting R_2 with this semijoin (eq. (3.10)). Thus,

$$\text{benefit} = 10^{-3} * 9 * 900 = 8.1 \text{ seconds}.$$

The cost of the transmission is

$$\text{cost} = 2.0 + 10^{-3} * 4 * 2700 = 12.8 \text{ seconds.}$$

Thus, this is not a beneficial transmission for further consideration.

(2) Test the intermediate transmission of d_{12} to R_2 and to R_3:

$$\text{benefit} = 10^{-3} * 9 * 900 + 10^{-3} * 6 * 400 = 8.1 + 2.4 = 10.5 \text{ seconds,}$$
$$\text{cost} = 2.0 + 10^{-3} * 7 * 2700 = 20.9 \text{ seconds.}$$

This is not a beneficial transmission for further consideration. Therefore, R_1^* is empty.

R_2: S-P-J

(1) Test the intermediate transmission of d_{22} to R_3:

$$\text{benefit} = 10^{-3} * 6 * 2000 = 12.0 \text{ seconds,}$$
$$\text{cost} = 2.0 + 10^{-3} * 3 * 1500 = 6.5 \text{ seconds.}$$

This is a beneficial transmission with total benefit of 5.5 seconds.

(2) Test the intermediate transmission of d_{22} to R_3 and to R_1:

$$\text{benefit} = 10^{-3} * 6 * 2000 + 10^{-3} * 3 * 6000$$
$$= 12.0 + 18.0 = 30.0 \text{ seconds,}$$
$$\text{cost} = 2.0 + 10^{-3} * 6 * 1500 = 11.0 \text{ seconds.}$$

This is a beneficial transmission with total benefit of 19.0 seconds.

(3) Test the intermediate transmission of (d_{21}, d_{22}) to R_3:

$$\text{benefit} = 10^{-3} * 6 * 2400 = 14.4 \text{ seconds,}$$
$$\text{cost} = 2.0 + 10^{-3} * 3 * 3500 = 13.5 \text{ seconds.}$$

This is a beneficial transmission with total benefit of 0.9 second.

(4) Test the intermediate transmission of (d_{21}, d_{22}) to R_3 and to R_1:

$$\text{benefit} = 10^{-3} * 6 * 2400 + 10^{-3} * 3 * 7200$$
$$= 14.4 + 21.6 = 36.0 \text{ seconds,}$$
$$\text{cost} = 2.0 + 10^{-3} * 6 * 3500 = 23.0 \text{ seconds.}$$

This is a beneficial transmission with total benefit of 13.0 seconds.

The most beneficial intermediate transmission for R_2 is (d_{22}:R_3, R_1).

R_3: Warehouse

(1) Test the intermediate transmission of d_{31} to R_1:

$$\text{benefit} = 10^{-3} * 3 * 9600 = 28.8 \text{ seconds,}$$
$$\text{cost} = 2.0 + 10^{-3} * 3 * 500 = 3.5 \text{ seconds.}$$

This is a beneficial transmission with total benefit of 25.3 seconds.

(2) Test the intermediate transmission of d_{31} to R_1 and to R_2:

$$\text{benefit} = 10^{-3} * 3 * 9600 + 10^{-3} * 9 * 7200$$
$$= 28.8 + 64.8 = 93.6 \text{ seconds,}$$
$$\text{cost} = 2.0 + 10^{-3} * 7 * 500 = 5.5 \text{ seconds.}$$

This is a beneficial transmission with total benefit of 88.1 seconds.

Table IV. General Query Parameters after Pass 1 (Address Ring and Broadcast Network)

Relation: R_i	Location: S_i	Size: s_i (bytes)	Join attribute $d_{i1} = P\#$		Join attribute $d_{i2} = S\#$	
			b_{i1} (bytes)	p_{i1}	b_{i2} (bytes)	p_{i2}
R_1: Supplier	S_2	960	—	—	1080	0.36
R_2: S-P-J	S_6	720	400	0.16	600	0.20
R_3: Warehouse	S_9	4000	500	0.20	1200	0.40

(3) Test the intermediate transmission of (d_{31}, d_{32}) to R_1:

$$\text{benefit} = 10^{-3} * 3 * 11040 = 33.1 \text{ seconds},$$
$$\text{cost} = 2.0 + 10^{-3} * 3 * 1700 = 7.1 \text{ seconds}.$$

This is a beneficial transmission with total benefit of 26.0 seconds.

(4) Test the intermediate transmission of (d_{31}, d_{32}) to R_1 and to R_2:

$$\text{benefit} = 10^{-3} * 3 * 11040 + 10^{-3} * 9 * 8280$$
$$= 33.1 + 74.5 = 107.6 \text{ seconds},$$
$$\text{cost} = 2.0 + 10^{-3} * 7 * 1700 = 13.9 \text{ seconds}.$$

This is a beneficial transmission with total benefit of 93.7 seconds.

The most beneficial intermediate transmission for R_3 is $(d_{31}, d_{32}: R_1, R_2)$.

Step 3—*Pass* 1. Select the most beneficial intermediate transmission.

R_3^* is the most cost-beneficial transmission. QS = $\{(d_{31}, d_{32}: R_1, R_2)\}$. The estimated database state after this transmission is calculated in Table IV.

Step 4—*Pass* 1. Eliminate sending relation.

R_3 is removed from Q. Since Q is not empty, return to step 2 for another pass.

Step 2—*Pass* 2. Find best intermediate transmission for each relation in Q.

R_1: Supplier

As in pass 1, R_1^* is empty; no beneficial transmissions from R_1 can be found.

R_2: S-P-J

(1) Test the intermediate transmission of d_{22} to R_3:

$$\text{benefit} = 10^{-3} * 6 * 2000 = 12.0 \text{ seconds},$$
$$\text{cost} = 2.0 + 10^{-3} * 3 * 600 = 3.8 \text{ seconds}.$$

This is a beneficial transmission with total benefit of 8.2 seconds.

(2) Test the intermediate transmission of d_{22} to R_3 and to R_1:

$$\text{benefit} = 10^{-3} * 6 * 2000 + 10^{-3} * 3 * 480$$
$$= 12.0 + 1.4 = 13.4 \text{ seconds},$$
$$\text{cost} = 2.0 + 10^{-3} * 6 * 600 = 5.6 \text{ seconds}.$$

This is a beneficial transmission with total benefit of 7.8 seconds.

Table V. General Query Parameters after Pass 2 (Address Ring)

Relation: R_i	Location: S_i	Size: s_i (bytes)	Join attribute $d_{i1} = P\#$		Join attribute $d_{i2} = S\#$	
			b_{i1} (bytes)	p_{i1}	b_{i2} (bytes)	p_{i2}
R_1: Supplier	S_2	960	—	—	1080	0.36
R_2: S-P-J	S_6	720	400	0.16	600	0.20
R_3: Warehouse	S_9	1600	400	0.16	600	0.20

(3) Test the intermediate transmission of (d_{21}, d_{22}) to R_3:

$$\text{benefit} = 10^{-3} * 6 * 2400 = 14.4 \text{ seconds},$$
$$\text{cost} = 2.0 + 10^{-3} * 3 * 1000 = 5.0 \text{ seconds}.$$

This is a beneficial transmission with total benefit of 9.4 seconds.

(4) Test the intermediate transmission of (d_{21}, d_{22}) to R_3 and to R_1:

$$\text{benefit} = 10^{-3} * 6 * 2400 + 10^{-3} * 3 * 576$$
$$= 14.4 + 1.7 = 16.1 \text{ seconds},$$
$$\text{cost} = 2.0 + 10^{-3} * 6 * 1000 = 8.0 \text{ seconds}.$$

This is a beneficial transmission with total benefit of 8.1 seconds.

The most beneficial intermediate transmission for R_2 is $(d_{21}, d_{22}: R_3)$.

Step 3—*Pass* 2. Select the most beneficial intermediate transmission.

R_2^* is the most cost-beneficial transmission. QS = $\{(d_{31}, d_{32}: R_1, R_2), (d_{21}, d_{22}: R_3)\}$. The estimated database state after this transmission is calculated in Table V.

Step 4—*Pass* 2. Eliminate sending relation.

R_2 is removed from Q. Since Q is not empty, return to step 2 for another pass.

Step 2—*Pass* 3. Find best intermediate transmission from each relation in Q.

R_1: Supplier

Again, R_1^* is empty.

Step 3—*Pass* 3. Select the most beneficial intermediate transmission.

There are no beneficial intermediate transmission; therefore, go to step 5.

Step 5. Add final transmissions to QS.

Since R_2 is completely transmitted already in QS, only R_1 and R_3 need to be transmitted to the result site, S_5. QS = $\{(d_{31}, d_{32}: R_1, R_2), (d_{21}, d_{22}: R_3), (R_1: S_5), (R_3: S_5)\}$.

To demonstrate the cost benefit gained by executing the optimized strategy QS, consider its total cost against an initial feasible strategy of sending the

relations after initial processing to the result site; IFS = $\{(R_1:S_5), (R_2:S_5), (R_3:S_5)\}$.

$$\begin{aligned}\text{Cost(IFS)} &= (2.0 + 10^{-3} * 3 * 12000) + (2.0 + 10^{-3} * 9 * 9000) \\ &\quad + (2.0 + 10^{-3} * 6 * 4000) \\ &= 38.0 + 83.0 + 26.0 = 147.0 \text{ seconds}, \\ \text{Cost(QS)} &= (2.0 + 10^{-3} * 7 * 1700) + (2.0 + 10^{-3} * 3 * 1000) \\ &\quad + (2.0 + 10^{-3} * 3 * 960) + (2.0 + 10^{-3} * 6 * 1600) \\ &= 13.9 + 5.0 + 4.9 + 11.6 = 35.4 \text{ seconds}.\end{aligned}$$

The estimated cost savings from this optimization is 111.6 seconds, or 76 percent of the cost of the initial feasible solution.

5.2 Broadcast Network Optimization

The cost parameters on the broadcast network are

$$t_b = 3.0 \text{ seconds}, \qquad c_b = 5 * 10^{-3} \text{ second/byte}.$$

Recording each step of Algorithm BN-GENERAL, the optimization proceeds as follows.

Step 1. Local processing and attribute ordering are done.

$Q = \{R_1, R_2, R_3\}$ and QS is empty.

Step 2—*Pass* 1. Find the best intermediate transmission from each relation in Q.

R_1: Supplier
Test the intermediate transmission of d_{12}.

Each possible semijoin from the transmitted attribute is tested separately and added only if beneficial.

$$\begin{aligned}\text{benefit on } R_2 &= 5 * 10^{-3} * 900 = 4.5 \text{ seconds}, \\ \text{benefit on } R_3 &= 5 * 10^{-3} * 400 = 2.0 \text{ seconds}, \\ \text{cost} &= 3.0 + 5 * 10^{-3} * 2700 = 16.5 \text{ seconds}.\end{aligned}$$

Thus, this is not a beneficial transmission for further consideration. R_1^* is empty.

R_2: S-P-J

(1) Test the intermediate transmission of d_{22}:

$$\begin{aligned}\text{benefit on } R_1 &= 5 * 10^{-3} * 6000 = 30.0 \text{ seconds}, \\ \text{benefit on } R_3 &= 5 * 10^{-3} * 2000 = 10.0 \text{ seconds}, \\ \text{cost} &= 3.0 + 5 * 10^{-3} * 1500 = 10.5 \text{ seconds}.\end{aligned}$$

This is a beneficial transmission with total benefit of 29.5 seconds.

(2) Test the intermediate transmission of (d_{21}, d_{22}):

$$\begin{aligned}\text{benefit on } R_1 &= 5 * 10^{-3} * 7200 = 36.0 \text{ seconds}, \\ \text{benefit on } R_3 &= 5 * 10^{-3} * 2400 = 12.0 \text{ seconds}, \\ \text{cost} &= 3.0 + 5 * 10^{-3} * 3500 = 20.5 \text{ seconds}.\end{aligned}$$

This is a beneficial semijoin with total benefit of 27.5 seconds.

The most beneficial intermediate transmission for R_2 is $(d_{22}: R_3, R_1)$.

R_3: Warehouse

(1) Test the intermediate transmission of d_{31}:

$$\text{benefit on } R_1 = 5 * 10^{-3} * 9600 = 48.0 \text{ seconds},$$
$$\text{benefit on } R_2 = 5 * 10^{-3} * 7200 = 36.0 \text{ seconds},$$
$$\text{cost} = 3.0 + 5 * 10^{-3} * 500 = 5.5 \text{ seconds}.$$

This is a beneficial transmission with total benefit of 78.5 seconds.

(2) Test the intermediate transmission of (d_{31}, d_{32}):

$$\text{benefit on } R_1 = 5 * 10^{-3} * 11040 = 55.2 \text{ seconds},$$
$$\text{benefit on } R_2 = 5 * 10^{-3} * 8280 = 41.4 \text{ seconds},$$
$$\text{cost} = 3.0 + 5 * 10^{-3} * 1700 = 11.5 \text{ seconds}.$$

This is a beneficial transmission with total benefit of 85.1 seconds.

The most beneficial intermediate transmission for R_2 is $(d_{31}, d_{32}: R_1, R_2)$.

Step 3—*Pass* 1. Select the most beneficial intermediate transmission.

R_3^* is the most cost-beneficial transmission. QS = $\{(d_{31}, d_{32}: R_1, R_2)\}$. The estimated database state after this transmission is calculated in Table IV.

Step 4—*Pass* 1. Eliminate sending relation.

R_3 is removed from Q. Since Q is not empty, return to step 2 for another pass.

Step 2—*Pass* 2. Find the best intermediate transmission from each relation in Q.

R_1: Supplier

Test the intermediate transmission of d_{12}:

$$\text{benefit on } R_2 = 5 * 10^{-3} * 72 = 0.4 \text{ second},$$
$$\text{benefit on } R_3 = 5 * 10^{-3} * 400 = 2.0 \text{ seconds},$$
$$\text{cost} = 3.0 + 5 * 10^{-3} * 1080 = 8.4 \text{ seconds}.$$

This is not a beneficial transmission for further consideration. Therefore, R_1^* is empty.

R_2: S-P-J

(1) Test the intermediate transmission of d_{22}:

$$\text{benefit on } R_1 = 5 * 10^{-3} * 480 = 2.4 \text{ seconds},$$
$$\text{benefit on } R_3 = 5 * 10^{-3} * 2000 = 10.0 \text{ seconds},$$
$$\text{cost} = 3.0 + 5 * 10^{-3} * 600 = 6.0 \text{ seconds}.$$

This is a beneficial transmission with total benefit of 6.4 seconds.

(2) Test the intermediate transmission of (d_{21}, d_{22}):

$$\text{benefit on } R_1 = 5 * 10^{-3} * 576 = 2.9 \text{ seconds},$$
$$\text{benefit on } R_3 = 5 * 10^{-3} * 2400 = 12.0 \text{ seconds},$$
$$\text{cost} = 3.0 + 5 * 10^{-3} * 1000 = 8.0 \text{ seconds}.$$

Table VI. *General Query Parameters after Pass 2 (Broadcast Network)*

Relation: R_i	Location: S_i	Size: s_i (bytes)	Join attribute d_{i1} = P#		Join attribute d_{i2} = S#	
			b_{i1} (bytes)	p_{i1}	b_{i2} (bytes)	p_{i2}
R_1: Supplier	S_2	384	—	—	216	0.18
R_2: S-P-J	S_6	720	400	0.16	600	0.20
R_3: Warehouse	S_9	1600	400	0.16	600	0.20

This is a beneficial transmission with total benefit of 6.9 seconds.

The most beneficial intermediate transmission for R_2 is $(d_{21}, d_{22}:R_1, R_3)$.

Step 3—*Pass* 2. Select the most beneficial intermediate transmission.

R_2^* is the most cost-beneficial transmission. QS = $\{(d_{31}, d_{32}:R_1, R_2), (d_{21}, d_{22}:R_1, R_3)\}$. The estimated database state after this transmission is calculated in Table VI.

Step 4—*Pass* 2. Eliminate sending relation.

R_2 is removed from Q. Since Q is not empty, return to step 2 for another pass.

Step 2—*Pass* 3. Find best intermediate transmission from each relation in Q.

R_1: Supplier

Test the intermediate transmission of d_{12}. Since all attributes of R_2 are included in QS, there is no possible benefit to be gained by performing a semijoin on R_2. Therefore,

$$\text{benefit on } R_3 = 5 * 10^{-3} * 160 = 0.8 \text{ second},$$
$$\text{cost} = 3.0 + 5 * 10^{-3} * 216 = 4.1 \text{ seconds}.$$

This is not a beneficial transmission for further consideration. Therefore, R_1^* is empty.

Step 3—*Pass* 3. Select the most beneficial intermediate transmission.

There are no beneficial intermediate transmissions; therefore, go to step 5.

Step 5. Add final transmissions to QS.

Since R_2 is completely transmitted already in QS, only R_1 and R_3 need to be transmitted to the result site, S_5. QS = $\{(d_{31}, d_{32}:R_1, R_2), (d_{21}, d_{22}:R_1, R_3), (R_1:S_5), (R_3:S_5)\}$.

To demonstrate the cost benefit gained by executing the optimized strategy QS, consider its total cost against an initial feasible strategy of sending the relations after initial processing to the result site; IFS = $\{(R_1:S_5), (R_2:S_5), (R_3:S_5)\}$.

$$\begin{aligned}\text{Cost(IFS)} &= (3.0 + 5 * 10^{-3} * 12000) + (3.0 + 5 * 10^{-3} * 9000) \\ &\quad + (3.0 + 5 * 10^{-3} * 4000) \\ &= 63.0 + 48.0 + 23.0 = 134.0 \text{ seconds},\end{aligned}$$

$$\begin{aligned}\text{Cost(QS)} &= (3.0 + 5 * 10^{-3} * 1700) + (3.0 + 5 * 10^{-3} * 1000) \\ &\quad + (3.0 + 5 * 10^{-3} * 384) + (3.0 + 5 * 10^{-3} * 1600) \\ &= 11.5 + 8.0 + 4.9 + 11.0 = 35.4 \text{ seconds.}\end{aligned}$$

The estimated cost savings from this optimization is 98.6 seconds, or 74 percent of the cost of the initial feasible solution.

6. CONCLUSIONS AND FUTURE RESEARCH

Local area networks are the backbone of many current and future information systems. LANs support the decentralization of hardware, data, and control to achieve important organizational advantages such as performance, reliability, availability, and modularity. A critical performance problem is the derivation of data retrieval (query) strategies on the local area network.

The goal of this paper has been to present effective query optimization algorithms for local area networks. Two types of networks have been studied; address rings and broadcast networks. Provably optimal strategies for simple queries guided the development of heuristic optimization algorithms for general distributed queries on both network types. These algorithms included the consideration of both data transmission and local processing costs. Several examples illustrated the algorithms.

Future research plans include the implementation of these algorithms in an office information system being built as a research testbed. The system will include a distributed database system that supports query entry and optimization at any site. Of particular interest for future study will be the performance evaluation of the static method of optimization versus the dynamic method as proposed by Wah and Lien [16]. Also, techniques for optimizing query processing over redundant data copies will be investigated.

ACKNOWLEDGMENTS

We acknowledge a reviewer's comments that led to several improvements in the final version of this paper.

REFERENCES

1. APERS, P.M.G., HEVNER, A.R., AND YAO, S.B. Algorithms for distributed query optimization. *IEEE Trans. Softw. Eng. SE-9*, 1 (Jan. 1983), 57–68.
2. BERNSTEIN, P., AND CHIU, D.W. Using semijoins to solve relational queries. *J. ACM 28*, 1 (Jan. 1981), 25–40.
3. BERNSTEIN, P., GOODMAN, N., WONG, E., REEVE, C.L., AND ROTHNIE, J.B., JR. Query processing in a system for distributed databases (SDD-1). *ACM Trans. Database Syst. 6*, 4 (Dec. 1981), 602–625.
4. CERI, S., AND PELAGATTI, G. *Distributed Databases: Principles and Systems.* McGraw-Hill, New York, 1984.
5. CODD, E. A relational model of data for large shared data banks. *Commun. ACM 13*, 6 (Jun. 1970), 377–387.
6. DATE, C. *An Introduction to Database Systems*, 3rd ed. Addison-Wesley, Reading, Mass., 1981.
7. DWYER, P. A study of materialization and access planning. Tech. Rep. CSC-84-10, Computer Sciences Cent., Honeywell, Inc., Bloomington, Minn., Feb. 1984.
8. GOUDA, M., AND DAYAL, U. Optimal semijoin schedules for query processing in local distributed

database systems. *Proceedings of the International Conference on Management of Data* (Ann Arbor, Mich., Apr. 29–May 1), ACM, New York, 1981.

9. HEVNER, A.R. The optimization of query processing on distributed database systems. Ph.D. dissertation, Computer Science Dept., Purdue Univ., West Lafayette, Ind., Dec. 1979.
10. HEVNER, A.R. A complexity analysis of the distributed query optimization problem. Working paper, Coll. of Business and Management, Univ. of Maryland, Feb. 1985.
11. HEVNER, A.R., AND YAO, S.B. Query processing in distributed database systems. *IEEE Trans. Softw. Eng. SE-5*, 3 (May 1979), 177–187.
12. KERSCHBERG, L., TING, P.D., AND YAO, S.B. Query optimization in Star computer networks. *ACM Trans. Database Syst. 7*, 4 (Dec. 1982), 678–711.
13. KING, J.C. Centralized versus decentralized computing: Organizational considerations and management options. *ACM Comput. Surv. 15*, 4 (Dec. 1983), 319–349.
14. SACCO, G. Distributed query evaluation in local area networks. In *Proceedings of the International Conference on Data Engineering* (Los Angeles, Calif., Apr.) 1984, pp. 510–516.
15. TANENBAUM, A. *Computer Networks.* Prentice-Hall, Englewood Cliffs, N.J., 1981.
16. WAH, B. AND LIEN, Y. The file-assignment and query-processing problems in local multiaccess networks. In *Proceedings of the International Conference on Data Engineering* (Los Angeles, Calif., Apr.). 1984, pp. 228–235.

Received September 1984