MINIMUM DELAY SEMIJOIN SCHEDULES
FOR LOCAL AREA DISTRIBUTED DATABASE SYSTEMS

Shigeru Masuyama, Shojiro Muro, Toshihide Ibaraki
Tadashi Mizutani and Toshiharu Hasegawa
Department of Applied Mathematics and Physics
Faculty of Engineering, Kyoto University

1. Introduction.

In a relational database system, users submit queries expressed in
terms of relations (i.e., tables or files), which might be time consuming to
answer. Especially in a distributed relational database system, a query
may necessitate to consult different files, consuming time and communication
cost. Thus how to reduce the query processing time and cost is one of the
major issues in designing a practical database system.

To enhance the query processing in a distributed relational database
system, the concept of semijoin is introduced by [WONG 77], [BERN 81b], as
a means to reduce the amount of data transmission among the sites. By
exchanging certain information before answering a query, the table sizes
of files to be transmitted can be often reduced drastically. The properties
of semijoin have been intensively studied under two criteria, the communica-
tion cost and the response time, putting emphasis usually on the former in
the environment of global public communication networks. Bernstein and
Chiu [BERN 81a] introduced the concept of a tree query for which an efficient
semijoin processing is possible, as well as a necessary and sufficient
condition stated in a graph theoretic context for a query to be a tree query.
Chiu [CHIU 81] also developed an efficient algorithm to obtain optimal semi-
join schedules for some special classes of queries under the communication cost criterion.

Recently, local area distributed systems become of great concern in conjunction with office automation. In a local area system, the building cost of a communication network is not very high compared with other equipments in the system, and the network tends to have enough bandwidth, making the communication speed rather fast and the communication cost almost negligible. In this new environment, optimality criterion and constraints imposed on the semijoin schedules are different from those discussed so far.

An attempt to clarify the role of semijoins in local area systems was first made by [GOUD 81]. We adopt here a much more simplified model. In this model, we impose no constraint on the transmission line capacity and the communication processing capability at each site, assuming the development of hardware technologies in the near future. In other words, any number of transmission can be made in parallel among the sites connected to the network. For this model, the minimum delay semijoin schedule is defined as the one requiring the shortest time, assuming that any semijoin operation from one relation to others requires exactly one unit of time. For this model, an efficient algorithm to obtain such schedules is developed for tree queries.

Our model is different from the one discussed by [GOUD 81] in the sense that [GOUD 81] uses a more detailed cost criterion taking into account the time delays caused by preprocessing, post processing and transmission of data. For each simplified model as employed in this paper, there is a simple and efficient algorithm to compute shortest semijoin schedules, whereas most problems concerning the model of [GOUD 81] turn out to be NP-hard (i.e., computationally intractable) as proved therein. Considering also that
the exact knowledge about data, such as transmission delays, is usually
difficult to measure in advance, our model may serve as an approximate
model useful for practical purposes.

Finally we mention that our model is also applicable to some kinds of
database machines in which multiprocessors handling different files are
mutually connected via multibusses. Semijoins in such machines may become
realistic if the number of processors becomes considerably large.

2. Definitions.

Terminologies for the subsequent discussion are briefly sketched here
by assuming the familiarity of the reader with relational database
terminologies (see [BERN 81a]). Let $\mathbf{A}$ denotes the set of all attributes.
A relation $R_i$ is a set of tuples $t_k$ over a set of attributes $A_{i} \subseteq \mathbf{A}$, where
$t_k[A]$ denotes the value of $t_k$ for an attribute $A \subseteq A_i$. $D=\{R_1, R_2, \ldots, R_n\}$
is a database. We consider only equijoin queries over $D$. As is well known,
such a query $q$ is characterized by $\{S_1, S_2, \ldots, S_a\}$, where $S_\ell$'s are disjoint
subsets of $\mathbf{A}$. The result $D'=q(D)$ of the query $q$ is given by $D'=\{R'_1, R'_2, \ldots, R'_n\}$ where $t_k \in R_\ell$ belongs to $R'_\ell$ if and only if
$\exists t \in R_1 \times R_2 \times \cdots \times R_n$ such that
$t[A]=t[A']$ for any two attributes $A, A' \subseteq S_\ell$ for every $\ell$ with $1 \leq \ell \leq a$ and $t_k[B]=t[B]$ for all $B \subseteq A_i$. The notation $R'_\ell=q(R_\ell)$ is used when $D'=q(D)$ holds.

When $q(R_1)$ is obtained, $R_1$ is said to be fullreduced (with respect to $q$).
As a semijoin involves two relations at different sites, we assume that any
two attributes in $S_\ell$ belong to different $A_i$.

For a query $q$, closure graph $G_q=(V_q, E_q)$ is an undirected graph defined
by

$V_q=\{R_1|A_i \cap (S_1 \cup S_2 \cup \cdots \cup S_a) \neq \emptyset\}$

$E_q=\{(R_i, R_j)|i \neq j \text{ and } \exists S_\ell, \exists A_i, \exists B_j \text{ such that } A, B \subseteq S_\ell\}$. 

3
For simplicity, we usually do not distinguish a relation $R_i$ and a node $R_i$. A clique of $G_q$ is a complete subgraph of $G_q$. A clique is maximal if there is no clique properly containing it. As obvious from the definition, each maximal clique of $G_q$ corresponds to one $S_q$. A terminal clique in $G_q$ is a maximal clique which shares at most one $R_i$ with other maximal cliques. Each node in a terminal clique, except such $R_i$ also included in another clique, is called a terminal node. The distance between $R_i$ and $R_j$, denoted by $d(R_i, R_j)$, is the length of a shortest path between $R_i$ and $R_j$ in $G_q$. Let

$$L(R_i) = \max_{R_j \in V_q} d(R_i, R_j).$$

The center of $G_q$ is the node $R_i \in V_q$ with the smallest $L(R_i)$.

A tree query is a query whose closure graph has no cycle except within a clique. Finally, a semijoin $R_i \triangleleft A \bowtie B R_j$, where $A \subseteq A_i$ and $B \subseteq A_j$, is defined as follows:

$$R_i \triangleleft A \bowtie B R_j = \{t_k \in R_i \mid \exists t' \in R_j \text{ such that } t_k[A] = t'[B] \}.$$  

Example 1. Consider a database $D = \{R_1, R_2, \ldots, R_{14}\}$, where $A_1 = \{A\}$, $A_2 = \{B, C\}$, $A_3 = \{D\}$, $A_4 = \{E, F, G\}$, $A_5 = \{H\}$, $A_6 = \{I\}$, $A_7 = \{J, K\}$, $A_8 = \{L\}$, $A_9 = \{M\}$, $A_{10} = \{N, O\}$, $A_{11} = \{P, Q\}$, $A_{12} = \{R\}$, $A_{13} = \{S\}$ and $A_{14} = \{T\}$, and a query $q = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$, where $S_1 = \{A, B\}$, $S_2 = \{C, D, E, H\}$, $S_3 = \{F, I\}$, $S_4 = \{J, L, M\}$, $S_5 = \{G, K, N\}$, $S_6 = \{O, P, R\}$ and $S_7 = \{Q, S, T\}$. The closure graph for $q$ is illustrated in Fig. 1. □

3. Description of the Model,

Simplifying assumptions on our model of a local area distributed database system are listed below.

(A1) There is one relation at each site.

(A2) At each data transmission, the set of all values of one attribute of a relation can be sent with a constant transmission delay (one
unit of time) regardless of the length of data. Thus a semijoin operation is completed in a unit of time.

(A3) The time delay to process the data at each site is neglected.

(A4) There is no capacity constraint on the communication lines, and any number of data can be transmitted in parallel among the sites at the same time. In particular, broadcast to all sites from one site is possible.

(A5) There is no restriction on the communication capacity at each site, i.e., each site can send and receive data simultaneously to and from any number of different sites.

4. Semijoin Schedule for a Given Query \( q \).

In this section, we define a semijoin schedule for our model by means of a schedule forest. A semijoin schedule describes the execution order of semijoins to obtain the result \( q(D) \) for a query \( q \) and a database \( D \).

A schedule forest for a given tree query \( q \) is a labeled forest \( F_q = \{T_1, T_2, \ldots, T_m\} \) constructed by the following procedure, where \( T_k \) is a tree with labeled edges.

Procedure FOREST

Input: A closure graph \( G_q \) for a tree query \( q \), and a set of one node \( \{R_{i_1}\} \) of \( G_q \) (i.e., \( m=1 \)) or a maximal clique \( \{R_{i_1}, R_{i_2}, \ldots, R_{i_m}\} (m \geq 2) \) of \( G_q \).

Output: The schedule forest \( F_q = \{T_1, T_2, \ldots, T_m\} \).

Step 1. If \( m=1 \), then let \( E'=E \) else let \( E'=E - \{(R_{i_j}, R_{i_\ell}) | 1 \leq j, \ell \leq m, j \neq \ell \} \). The resulting graph \( G' = (V_q, E') \) consists of \( m \) components.

Step 2. Obtain \( F_q \) by constructing trees \( T_1, T_2, \ldots, T_m \) which are subgraphs of \( G' \) such that any node \( R_{i_k} \) is connected to one of \( R_{i_1}, R_{i_2}, \ldots, R_{i_m} \) via the shortest path. Such trees can be obtained by scanning each component of \( G' \) from \( R_{i_k} \) in breadth-first manner and choosing edges in this
order unless the resulting graph contains a cycle.

**Step 3.** For each $T_k$, let $L_k = \max\{d(R_{ik}, R_j) | R_j \text{ is in } T_k\}$. Then assign $\ell(e) = L_k - d(R_{ik}, R_j)$ to each edge $e = (R_{i}, R_j)$ in $T_k$, where $R_i$ is closer to $R_{ik}$ than $R_j$.

**Example 2.** Fig. 2 shows an example of a schedule forest for the query $q$ of Example 1.

For a given query $q$, and a schedule forest $F_q$, the semijoin schedule is defined by the following procedure.

**Procedure SCHEDULE**

**Input:** A schedule forest $F_q$ and a database $D = \{R_1, R_2, \ldots, R_n\}$.

**Output:** $q(R_i)$ $(i=1, 2, \ldots, n)$ (at each site $i$).

**Step 1.** For each edge $e = (R_i, R_j)$ in $T_k$, where $R_i$ is closer to $R_{ik}$ of $T_k$ than $R_j$, execute semijoin $R_i < A = B \land R_j$ with $A, B \in S_\ell$ for some $\ell$, at the time of the label associated with $e$.

**Step 2.** After executing Step 1, execute all the semijoins $R_{ik} < A = B \land R_j$, $k \neq \ell$, $1 \leq k, \ell \leq m$, simultaneously (i.e., broadcast from each $R_{i\ell}$ the set of values of $A \in S_{i\ell}$ to all other $R_{ik}$).

**Step 3.** Execute semijoins $R_j < B = A \land R_i$ corresponding to the edges $e = (R_i, R_j)$ in $T_k$, where $R_i$ is closer to the root $R_{ik}$ than $R_j$, in the reverse order of the labels associated with edges.

The total time units $L_q$ (which is called to be the **schedule length**, or **delay**) required for the above semijoin schedule is

$$L_q = \begin{cases} 
2L_i, & \text{if } m=1, \\
2 \max_{1 \leq k \leq m} L_k + 1, & \text{if } m>1,
\end{cases}$$

where $L_k$ is defined in Step 3 of FOREST.

The following two lemmas prove that $q(R_i)$ $(i=1, 2, \ldots, n)$ is in fact obtained at each site $i$ after Procedure Schedule is executed.
Lemma 4.1. ([BERN 81a]) Given a tree query \( q \), \( q(D) \) is obtained by a semijoin schedule if and only if it contains a sequence of semijoins along a path of \( G_q \) from \( R_j \) to \( R_i \) for any pair of relations \( R_i \) and \( R_j \) in \( G_q \). □

Lemma 4.2. \( q(D) \) is obtained by Procedure SCHEDULE.

(Proof). In the semijoin schedule given by Procedure SCHEDULE, there is a sequence of semijoins along a path for each pair of nodes \( R_i \) and \( R_j \) in \( G_q \). □

5. Linear Time Algorithm to Obtain the Shortest Semijoin Schedule from \( G_q \)

In this section, an algorithm MINIDELAY is developed to obtain a schedule forest which gives the shortest (i.e., minimum delay) semijoin schedule from a given closure graph \( G_q \). MINIDELAY uses two subroutines CENTER and FOREST. Note that it is straightforward to construct the closure graph \( G_q \) from a given query \( q \) ([BERN 81c]). The required time is \( O(n^2|\Delta|) \).

Algorithm MINIDELAY

Input: A closure graph \( G_q \).

Output: A schedule forest \( F \) which gives the shortest semijoin schedule.

Step 1. (Finding the centers of \( G_q \)). Call Procedure CENTER.

Step 2. (Construction of the schedule forest \( F \)). With the set of centers \( \{R_{i1}, R_{i2}, \ldots, R_{im}\} \) obtained in Step 1, call Procedure FOREST.

Step 3. Halt. □

Procedure CENTER computes the centers of \( G_q \). Since \( G_q \) corresponding to a tree query \( q \) is very close to a tree, i.e., \( G_q \) has no cycle except within a clique, the algorithm proposed by [HAN D 73] for trees can be used with a minor modification.

Procedure CENTER

Input: Closure graph \( G_q \) for a tree query \( q \).
Output: The set of centers \( \{R_i, R_1, \ldots, R_m\} \) of \( F_q \).

Step 1. Take a terminal node \( R_i \) of \( G_q \) and let \( R_j \) be the farthest node from \( R_i \) (i.e., the shortest path from the chosen \( R_i \) to \( R_j \) is the longest).

Step 2. Let \( R_f \) be the farthest node from \( R_j \) in \( G_q \) and \( d \) denote the length between \( R_j \) and \( R_f \).

(i) If \( d=2\ell \), then let \( \{R_i, R_1\} \rightleftharpoons \{ \text{the middle point of the shortest path from } R_j \text{ to } R_f \} \).

(ii) If \( d=2\ell+1 \), then let \( \{R_i, R_1, \ldots, R_m\} \rightleftharpoons \{ V_C \} \), where \( V_C \) is the set of nodes of the maximal clique \( C \) such that two nodes \( R_s \) and \( R_t \) on the shortest path with distance \( \ell \) from \( R_j \) to \( R_f \), respectively, belong to \( C \).

Step 3. Return.

Example 3. In Fig. 1, take a terminal node \( R_6 \) as \( R_1 \) of Step 1. Then \( R_{14} \) is the farthest node from \( R_6 \), and the farthest node from \( R_{14} \) is \( R_1 \). The waved line in Fig. 1 shows the shortest path from \( R_6 \) to \( R_{14} \), while the bold line shows the shortest path from \( R_{14} \) to \( R_1 \). Double circles denote the set \( \{R_i, R_1, \ldots, R_m\} \rightleftharpoons \{R_4, R_7, R_{10}\} \) obtained by Case (ii) of Step 2. The schedule forest \( F_q \) obtained from \( \{R_4, R_7, R_{10}\} \) by Procedure FOREST is already given in Fig. 2, as discussed in Example 2.

Note 5.1. The farthest node from a node \( R_i \) is obtained by scanning \( G_q \) from \( R_i \) in a breadth-first manner.

Theorem 5.1. The algorithm MINIDELAY constructs the shortest semijoin schedule in \( O(n) \) time.

Only the time complexity required by the above computation is briefly analysed. As noted in the algorithms, the essential part of FOREST and CENTER is the breadth-first search of \( G_q \) or subgraphs of \( G_q \). The following tree \( T_q \) and lists \( L_t \) facilitates the execution of the breadth-first search.

The **clique tree** \( T_q = (V_C \cup V_J, E_J) \) is a tree where
\[ V_C = \{ S_1 \mid S_1 \in \mathcal{E} \} \]
\[ V_J = \{ R_j \mid \exists S_{j'}, \exists S_{j''}, \epsilon \exists L_{j}, A \in \mathcal{A}_{j'} \text{ such that } A \in S_{j'}, A' \in S_{j''}, \} \]
\[ E_j = \{ (S_1, R_j) \mid R_j \in V_J \text{ and } \exists A \in S_1 \text{ such that } A \in \mathcal{A}_j. \} \]

Note that \(|q| = |V_C| \leq n \) because \( q \) is a tree query, and \(|V_J| \leq |V_q| = n \) is obvious, where \( V_q \) is the set of nodes in \( G_q \). Thus \(|V_C \cup V_J| = O(n)\). We also use the following lists
\[ LT_1 = \{ R_j \mid R_j \in V - V_J \text{ and } \exists A \in \mathcal{A}_j \text{ such that } A \in S_1. \} \]

\( T_q \) and \( LT_1 \)'s are also constructed at the time of constructing \( G_q \).

With these data structures, the breadth-first search on \( G_q \) is executed in \( O(n) \) time as shown below. It is simulated by the breadth-first search on \( T_q \) due to the obvious correspondence between nodes in \( G_q \) and \( T_q \). The breadth-first search on \( T_q \) starts from the node corresponding to the starting node in \( G_q \) specified by FOREST and CENTER respectively. When node \( S_1 \) in \( T_q \) is first visited during the search, the list \( LT_1 \) is referred and all the nodes in \( LT_1 \) are connected to \( R \), where \( R \) is the node in \( T_q \) which is adjacent to \( S_1 \) and has already been visited. Note that this \( R \) is uniquely determined because of the way of the execution of the breadth-first search.

The breadth-first search executed in this way is obviously done in \( O(n) \) time, because each node in \( V_q - V_J \) appears exactly once in lists \( LT_1 \) and each list is scanned only once. This shows that FOREST and CENTER requires \( O(n) \) time. The construction of semijoin is also \( O(n) \) time.


In the current model, each site is assumed to be able to send and/or receive any number of data simultaneously. However, this assumption is not adequate if the communication processing capability at each site is limited and/or enough bandwidth of transmission line is not available, necessitating
the study of the models with restrictions on the communication processing capability at each site and/or the line capacity of the communication network. Some results along this line are already obtained and will be reported elsewhere.

Acknowledgement

Thanks are due to Prof. T. Kameda of Simon Fraser University, Canada, for introducing the authors to these topics.

References


[GOUĐ 81] Gouda, M.G. and Dayal, U., "Optimal Semijoin schedules for query processing in local distributed database systems", Proc. of ACM-SIGMOD International Conf. on Management of Data,
pp. 164-175, 1981.


Fig. 1. The closure graph $G_q$ for a query $q$ of Example 1
(● denotes a terminal node and ○ denotes the centers
obtained by Procedure CENTER).

Fig. 2. The schedule forest $F_q$ constructed by Procedure
MINIDELAY for the query $q$ of Example 1.