数理解析研究所講究録
第 494 巻 1983 年 13-24

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 necessiate 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 communication 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 semijoin shcedules 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 familiarlity of the reader with relational database terminologies (see [BERN 81a]). Let $\mathbb{A}$ denotes the set of all attributes. A relation R_i is a set of tuples t_k over a set of attributes $\mathbb{A}_i \subset \mathbb{A}$, where $t_k[A]$ denotes the value of t_k for an attribute $A \in \mathbb{A}_i$. $D=\{R_1, R_2, \cdots, R_n\}$ is a database. We consider only equijoin queries over D. As is well known, such a query q is characterized by $\{S_1, S_2, \cdots, S_a\}$, where S_ℓ's are disjoint subsets of $\mathbb{A}$. The result $D'=q(D)$ of the query q is given by $D'=\{R'_1, R'_2, \cdots, R'_n\}$ where $t_k \in R_i$ belongs to R'_i 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 $A, A' \in S_\ell$ for every ℓ with $1 \leq \ell \leq a$ and $t_k[B]=t[B]$ for all $B \in \mathbb{A}_i$. The notation $R'_i=q(R_i)$ is used when $D'=q(D)$ holds. When $q(R_i)$ is obtained, R_i 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_ℓ belong to different $\mathbb{A}_i$.

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

$$V_q=\{R_i \mid \mathbb{A}_i \cap (S_1 \cup S_2 \cup \cdots \cup S_a) \neq \phi\}$$

$$E_q=\{(R_i, R_j) \mid i \neq j \text{ and } \exists S_\ell, \exists A \in \mathbb{A}_i, \exists B \in \mathbb{A}_j \text{ such that } A, B \in S_\ell\}.$$

For simplicitity, 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 difinition, each maximal clique of G_q corresponds to one S_ℓ. 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 $R_i\langle A=B]R_j$, where $A \in \mathbb{A}_i$ and $B \in \mathbb{A}_j$, is defined as follows:

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

Example 1. Consider a database $D=\{R_1, R_2, \cdots, R_{14}\}$, 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

$F_q=\{T_1, T_2, \cdots, T_m\}$

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 $\{R_{i_1}, R_{i_2}, \cdots, R_{i_m}\}(m\geq 2)$ of G_q.

Output: The schedule forest $F_q=\{T_1, T_2, \cdots, T_m\}$.

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

Step 2. Obtain F_q by constructing trees $T_1, T_2, \cdots, T_m$ which are subgraphs of G'_q such that any node $R_i\in V_q$ is connected to one of $R_{i_1}, R_{i_2}, \cdots, R_{i_m}$ via the shortest path. Such trees can be obtained by scanning each component of G'_q 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 $L_k=\max\{d(R_{i_k}, R_j) | R_j \text{ is in } T_k\}$. Then assign label $\ell(e)=L_k-d(R_{i_k}, R_i)$ to each edge $e=(R_i, R_j)$ in T_k, where R_i is closer to R_{i_k} 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 $D=\{R_1, R_2, \cdots, R_n\}$.

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

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

Step 2. After executing Step 1, execute all the semijoins $R_{i_k}\langle A=B]R_{i_\ell}$, $k\neq\ell$, $1\leq k, \ell\leq m$, simultaneously (i.e., broadcast from each R_{i_ℓ} the set of values of $A\in\mathbb{A}_{i_\ell}$ to all other R_{i_k}).

Step 3. Execute semijoins $R_j\langle B=A]R_i$ corresponding to the edges $e=(R_i, R_j)$ in T_k, where R_i is closer to the root R_{i_k} 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

$$L_q=\begin{cases} 2L_1, & \text{if } m=1, \\ 2\max_{1\leq k\leq m} L_k+1, & \text{if } m>1, \end{cases}$$

where L_k is defined in Step 3 of FOREST.

The following two lemmas prove that $q(R_i)$ $(i=1, 2, \cdots, n)$ 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 $O(n^2|A|)$.

Algortihm MINIDELAY

Input: A closure graph G_q.

Output: A schedule forest F_q 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_q). With the set of centers $\{R_{i_1}, R_{i_2}, \cdots, R_{i_m}\}$ 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 [HAND 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 $\{R_{i_1}, R_{i_2}, \cdots, R_{i_m}\}$ 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 $\{R_{i_1}\} \leftarrow$ {the middle point of the shortest path from R_j to R_f}.

(ii) If $d=2\ell+1$, then let $\{R_{i_1}, R_{i_2}, \cdots, R_{i_m}\} \leftarrow V_C$, where V_C is the set of nodes of the maximal clique C such that two nodes R_{s1} and R_{s2} on the shortest path with distance ℓ 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_i 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 $\{R_{i_1}, R_{i_2}, \cdots, R_{i_m}\}=\{R_4, R_7, R_{10}\}$ 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 essencial 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 LT_i facilitates the execution of the breadth-first search.

The clique tree $T_q=(V_C \cup V_J, E_J)$ is a tree where

$V_C=\{S_i | S_i \in q\}$

$V_J=\{R_j | \exists S_\ell, \exists S_{\ell'} \in q(\ell \neq \ell'), \exists A, \exists A' \in \mathbb{A}_j$ such that $A \in S_\ell, A' \in S_{\ell'}\}$

$E_J=\{(S_i, R_j) | R_j \in V_J$ and $\exists A \in S_i$ such that $A \in \mathbb{A}_j\}$.

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

$LT_i=\{R_j | R_j \in V_q-V_J$ and $\exists A \in \mathbb{A}_j$ such that $A \in S_i\}$.

T_q and LT_i'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_i in T_q is first visited during the search, the list LT_i is referred and all the nodes in LT_i are connected to R, where R is the node in T_q which is adjacent to S_i 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_i 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.

6. Discussions.

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, necessiating 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

[BERN 81a] Bernstein, P. A., and Chiu, D. M., "Using semijoins to solve relational queries", JACM, Vol. 28, No. 1, pp. 25-50, 1981.

[BERN 81b] Bernstein , P. A., Goodman, N., Wong, E. , Reeve, C. and Rothnie, J. B., "Query processing in a system for distributed databases (SDD-1)", ACM Trans. on Database Systems, Vol. 6, No. 4, pp.602-605, 1981.

[BERN 81c] Bernstein, P. A. and Goodman, N., "Power of natural semijoins", SIAM J. Computing, Vol. 10, No. 4, pp. 751-771. 1981.

[CHIU 81] Chiu, D. M., Bernstein P. A. and Ho, Y. C., "Optimizing chain queries in a distributed database system", Technical Report, Center for Research in Computing Technology, Aiken Computation Laboratory, Harvard University, TR-01-81, 1981.

[CODD 70] Codd, E. F., " A relational model of data for large shared data banks", Comm. ACM, Vol. 13, No. 6, pp. 377-387, 1970.

[GOUD 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.

[HAND 73] Handler, G. Y., "Minimax location of a facility in an undirected tree graph", Transportation Science, Vol. 7, pp. 287-293, 1973.

[HEVN 79] Hevner, A. R. and Yao, S. B., "Query processing in distributed database systems", IEEE Trans. Software Eng., SE-5, 3, pp. 178-187, 1979.

[WONG 77] Wong, E., "Retrieving dispersed data from SDD-1: A system for distributed databases", Proc. of Second Berkeley Workshop on Distributed Data Management and Computer Networks, pp. 217-235, 1977.

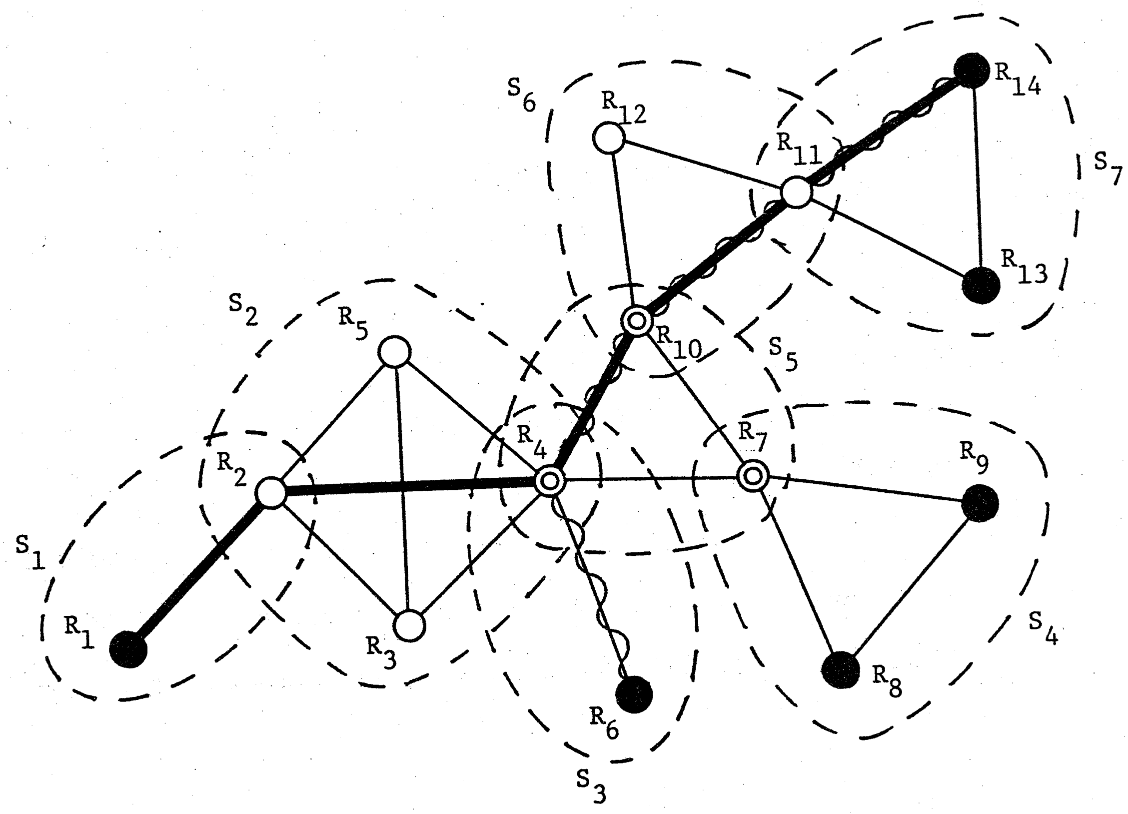

Fig. 1. The closure graph G_q for a query q of Example 1 (● denotes a terminal node and ◎ denotes the centers obtained by Procedure CENTER).

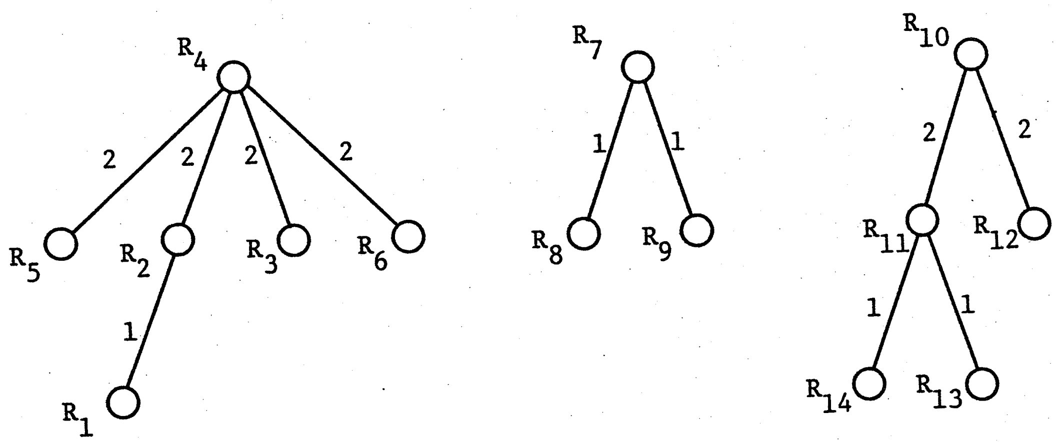

Fig. 2. The schedule forest F_q constructed by Procedure MINIDELAY for the query q of Example 1.