On Data Allocation With the Minimum Overall Communication Costs In Distributed Database Design

Xuemin Lin, Maria Orlowska, and Yanchun Zhang

Department of Computer Science
University of Queensland
Australia 4067

ABSTRACT

This paper investigates the problem of allocation of database fragments to a network, so that the overall communication cost for processing a given set of transactions is minimized. Presented first is a data allocation algorithm with respect to a "simple strategy" to process transactions. Secondly, we present a dynamic data allocation algorithm which is guaranteed to produce a "locally optimal" data allocation.

1. Introduction

In distributed database design, extensive study [1, 4, 8] has been carried out on the problem of how to allocate *fragments* [1, 6, 10] of a pre-existing database to the sites (individual computers) of a given network to enhance the performance of transaction processing. This is called the *data allocation* problem. Usually two objective criteria are applied to a data allocation problem:

- Given a set of transactions, a network of individual computers and a database, allocate the data such that the total response time of the execution of the transactions in T is minimized.
- Given a set of transactions, a network of individual computers and a database, allocate the database such that the overall communication costs are minimized.

Those two problems are inherently different, and are both important in distributed database design. In this paper, we shall study only the data allocation problem with the requirement of minimizing the overall communication cost.

In general, each of the following factors will substantially impact on the quality and overall performance of the system:

1. How do we split a relation into several *fragments*?
2. What kind of strategy is used to process each transaction?
3. What kind of approach is used to find a data allocation, based on the solutions of 1 and 2, to minimize the overall communication costs?

There are a number of works on how to split a relation into fragments [1, 6, 10]. Among them, the result in [1] is the one which most directly and effectively addresses the data allocation problem. In this paper, we assume that a *fragmentation* has been automatically obtained using the information about a set of most frequently used transactions.

For a given fragmentation, the relationship between factor 2 and factor 3 makes the data allocation problem logically intractable:

The quality of a data allocation can be evaluated through transaction processing strategies. Different strategies may lead to different minimum overall communication costs. On the other hand, without specific data allocation it is generally difficult to justify what strategies should be chosen.

This logical intractability is mainly caused by the consideration of the inter-relationship between the fragments (relations or objects), while processing a transaction.

A simple way [4, 8] to overcome this hard issue is to simplify the inter-relationship between fragments by the application of a "simple strategy" (see Section 3) when a query transaction is processed. For example, when performing a *join*, the fragments are sent to the *result site* - the site which requires the result of this transaction. The join operation is done at that site. Thus, the optimal model of the data allocation problem, under this simplification, is similar to the *file allocation* problem in classical distributed computing [1].

Further, in [1], general schedules to process the given transactions are pre-assumed. Following this a data allocation with the minimum overall "transmission cost"

[1] under the given general schedule is sought. A framework of a data allocation process in dynamic co-operation with an existing distributed optimizer is presented in [1].

The research in [1] corresponds only to the problem of minimizing overall communication cost on a "uniform network" (defined in Section 2). The generality of a network makes the data allocation more difficult. Also, it is generally difficult to judge whether or not a given pre-assumed strategy to process the given transactions is better than a simple strategy.

In this paper, our work concentrates on a general network. First, we present a polynomial time heuristic algorithm for finding a data allocation of given database fragments under a simple query strategy. The update strategy between the duplicated copies of a fragment adopted in this paper is different and better than that in [1, 4, 8] (see Section 2). The performance - optimality estimation results - of this algorithm and the computational intractability are also illustrated.

Second, we propose an approach to refine the data allocation obtained from the above algorithm. This approach starts from an initial data allocation, and then iteratively and greedily reduces the overall communication cost by either adding or dropping a copy of a fragment based on an employed optimizer.

This paper is organized as follows. In Section 2, we present some necessary preliminaries which include the necessary formalizations. In Section 3, a heuristic algorithm is presented for obtaining a data allocation under given simple query strategies, together with its theoretical performance guarantee. The computational intractability of the related optimization problems is shown in Section 3. In Section 4, we present a dynamic approach for obtaining a data allocation in co-operation with an existing optimizer. This is followed by the conclusion.

2. Preliminaries

This section includes a background discussion with respect to networks, fragments, and transaction management strategies.

2.1. Networks

Let I denote the set of all positive integers. A *physical network* $N=(V, E, p)$ consists of a *simple connected undirected graph* (V, E) [3] and a mapping p: $E \rightarrow I$, where each node in V represents an individual computer, each edge represents a link, and p is assigned so that the communication cost of a unit data volume through a link e is $p(e)$. The communication cost of sending a data volume U through a link e in N is $Up(e)$. Here (V, E) is called the *underlying graph* of N.

An undirected graph is *complete* if it is simple, and each pair of distinct vertices is connected by an edge. A physical network is *fully connected* if its underlying graph is complete. A physical network is *uniform* if it is fully connected and $p(e)=c$ for each link e, where c is a constant integer.

A network $N=(V, E, p)$ is *metric* if it is fully connected, and for a triple u, v, w of distinct vertices:

$$p((u, v)) \leq p((u, w)) + p((w, v)).$$

A physical network $G=(V, E, p)$ has a corresponding metric network $G'=(V, E', p')$ by the assignment of $p'((u,v))$ to the length of the *shortest path* [3] between u and v - a pair of distinct vertices. Here G' is called the *metric map* of G.

In this paper, a necessary restriction is added to the data allocation problem, that is, the communication between two computers on a physical network is through the shortest path in a physical network. Thus, finding a data allocation with the minimum overall communication cost on a physical network is equivalent to finding a data allocation with the minimum overall communication cost on its metric map.

In the rest of the paper, we consider only metric networks. "Metric network" is abbreviated to "network".

2.2. Fragmentations, Data Allocation, and Transactions

A *primary* fragmentation of relational database is a set $F=\{f_i: 1 \leq i \leq m\}$ of fragments with the property that $f_i \neq f_j$ if $i \neq j$. For the correctness issue of a fragmentation, we refer the reader to [6, 10]. A *duplicated* fragmentation is induced by a primary fragmentation so that each fragment may have several copies.

For the remainder of this paper, an element in a primary fragmentation is always called a fragment, while each element in a duplicated fragmentation is called a *copy* of a fragment. A fragmentation is always referred to as a primary fragmentation; and "primary fragmentation" is abbreviated to "fragmentation".

A *data allocation* L of F on a network $N=\{V, E, p\}$ is a map such that L: $F \rightarrow 2^V$.

Without loss of generality, we assume that the transactions are either query only or update only, and transactions are expressed by fragments.

2.3. Transaction Processings

Suppose that a fragmentation F, a network $N=(V, E, p)$, a data allocation L of F on N, and a transaction set T, are given. Further, let $F=\{f_i: 1 \leq i \leq m\}$ and $V=\{j: 1 \leq j \leq n\}$. In this section, we discuss strategies to process the transactions in T.

An Update Strategy: MST-Strategy

Note that in this paper, a transaction causing an update of several different fragments is always assumed to be expressed by those fragments. Further, the following update strategy called MST-strategy is applied to update all copies of a fragment located on N. Say a user at node j of the network N issues an update of a fragment f_i:

- the route for processing this transaction is a *minimum spanning tree* [3] of $L(f_i) \cup \{j\}$ in N.

For example, a given network, as illustrated in Figure 1, has the fragment f_1 located at site 1 and site 2. A transaction issued at site 3 requires an update of f_1. Using MST-strategy, the update route is from site 3 to site 1 to site 2. In early works, the *broadcasting strategy* [1] was applied to update copies of a fragment. In the above example, the route is from site 3 to site 1 and to site 2 by broadcasting strategy. Usually, MST-strategy leads to a smaller communication cost than broadcasting strategy, since the route of the broadcasting strategy is also a spanning tree.

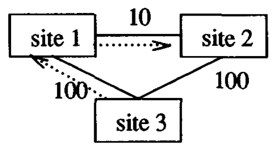

Figure 1

Let U_{ij} denote the total data volume required by the transactions in T issued at node j to update the fragment f_i. The overall communication cost for update f_i using MST-strategy is:

(2.1) $$\sum_{j=1}^{n} U_{ij}\, vmsp(j,\ L(f_i)),$$

where $vmsp(j,\ L(f_i))$ is the summation of the weights of the links of the minimum spanning tree of $\{j\} \cup L(f_i)$ in N. The overall communication cost to process the update type transactions of T is:

(2.2) $$\sum_{i=1}^{m} \sum_{j=1}^{n} U_{ij}\, vmsp(j,\ L(f_i)).$$

We should note that MST-strategy is not optimal; to find an optimal strategy (with the minimum communication cost) for a particular update among the sites containing copies of a fragment and the issuing site, is equivalent to the problem of finding the *minimum steiner tree* which is NP-hard [2].

Minimum Steiner Tree Problem

INSTANCE: Metric network $N=(V,\ E,\ p)$, a subset $V' \subseteq V$.

QUESTION: How can we find a subtree of N that includes all the vertices of V' and such that the sum of the weights of the edges in the subtree is minimized?

However, from [2], it follows that the communication cost by using MST-strategy is at most twice of the minimum communication cost.

Query Strategies

There are a number of ways to process a query. According to the query optimisation technique [12], we may always assume that selection and projection operations have been pushed down while processing a query, that is, it is necessary to first process the relevant selections and projections on the required fragments at the sites where those fragments are located. Then, a *simple strategy* to process a query is to send the contents of all fragments, which are required to access, to the query result site to implement the query; meanwhile if a fragment has several copies in the location L then the closest copy to the result site is chosen.

For example, assume that $f_1(A,B)$ and $f_2(B,C)$ are two fragments located at site 1 and site 2 of the network illustrated in Figure 1. A query, resulted at site 3, is represented in SQL [11] as following:

```
SELECT A, B, C
FROM f1, f2
WHERE f1.B = f2.B and f1.A≤50
```

The simple strategy to process this is to do a selection on f_1 at site 1 with the condition $f_1.A \leq 50$, and then to send the result of the selection and fragment f_2 to site 3 to implement the join.

Let Q_{ij} denote the total data volume of f_i required to send to site j to process the queries in T which resulted at site j by the simple strategy. Then the overall communication cost to process the queries in T is:

(2.3) $$\sum_{i=1}^{m} \sum_{j=1}^{n} Q_{ij}\, d(j,\ L(f_i)),$$

where $d(j,\ L(f_i)) = \min\{p((j,l)) : l \in L(f_i)\}$ and $d(j,\ L(f_i)) = 0$ if $j \in L(f_i)$. The overall communication cost to process the queries in T *with respect to* f_i is defined as:

(2.4) $$\sum_{j=1}^{n} Q_{ij}\, d(j,\ L(f_i))$$

To observe the computational intractability of processing a join by the minimum communication cost in nesting order, an efficient (polynomial time) optimization query algorithm [7] has been presented with the restriction to a given *chain* order. This algorithm executes any query in a chain order by the minimum communication cost for a given data allocation on a metric network.

3. A Data Allocation Algorithm Under the Simple Strategy

Suppose that $F=\{f_i:\ 1\le i\le m\}$ is a fragmentation, N is a network with node set $\{j:\ 1\le j\le n\}$, and T is a transaction set. In this section, we study the problem of finding a data allocation L of F on N so that the overall communication cost to process the transactions in T, by the simple query strategy and MST-strategy, is minimized. This optimization problem can be precisely stated as the following problem.

Simple Data Allocation Problem (SDAP)

Find an allocation L of F on N so that the following value is minimized:

$$(3.1)\quad \sum_{i=1}^{m}\sum_{j=1}^{n} Q_{ij}d(j,\ L(f_i))+\sum_{i=1}^{m}\sum_{j=1}^{n} U_{ij}vmsp(j,\ L(f_i)).$$

The SDAP is equivalent to finding an allocation L of f_i for each fragment so that

$$(3.2)\quad \sum_{j=1}^{n} Q_{ij}d(j,\ L(f_i))+\sum_{j=1}^{n} U_{ij}vmsp(j,\ L(f_i))$$

is minimized.

3.1. NP-hardness

In this section, we prove that SDAP is NP-hard. To prove the NP-hardness of SDAP, we need only to show the NP-hardness of the following problem.

Allocation Problem (AP)

INSTANCE: Given a network N with n nodes, a fragment f_i, two integers U_{ij} and Q_{ij} for each node j in N, and an integer K.

QUESTION: Is there a data allocation L of f_i such that the value of (3.2) is not greater than K?

Theorem 1: AP is NP-hard.

To prove Theorem 1, we first show a property (Lemma 1) for an allocation. For $1\le i\le m$, let $U_i=\sum_{j=1}^{n}U_{ij}$; and for $1\le i\le m$ and $1\le j\le n$, let $B_{ij}=Q_{ij}+U_{ij}-U_i$; and for an allocation L of f_i, let

$$cost(L(f_i))=\sum_{j=1}^{n} Q_{ij}d(j,\ L(f_i))+\sum_{j=1}^{n} U_{ij}vmsp(j,\ L(f_i)).$$

Lemma 1: Given a fragment f_i, suppose that L is an arbitrary allocation of f_i. Then $cost(L_1(f_i))\le cost(L(f_i))$ where $L_1(f_i)=L(f_i)\cup\{j\}$ for a node j with $B_{ij}\ge 0$.

Proof: Without loss of generality, we may assume that $j\notin L(f_i)$. We may immediately verify the following properties (3.3), (3.4) and (3.5) according to the definitions.

(3.3) $d(j,V_1)\le d(j,V_2)$ if $V_2\subseteq V_1$;

(3.4) $vmsp(k,V_1\cup\{j\})\le d(j,V_1)+vmsp(k,V_1)$;

(3.5) $vmsp(k,V_1\cup\{j\})=vmsp(j,V_1)$ if $j=k$,

where V_1 and V_2 are subsets of the node set of the network.

From the above properties, it follows that:

$$\begin{aligned}(3.6)\quad cost(L(f_i))\ge &\sum_{k\notin L_1(f_i)} Q_{ik}d(k,L_1(f_i))\\ &+Q_{ij}d(j,L(f_i)))\\ &+\sum_{k=1}^{n}U_{ik}vmsp(k,L(f_i)),\end{aligned}$$

$$\begin{aligned}(3.7)\quad cost(L_1(f_i))\le &\sum_{k\notin L_1(f_i)} Q_{ik}d(k,L_1(f_i))\\ &+U_{ij}vmsp(j,L(f_i))\\ &+\sum_{k\ne j}U_{ij}(vmsp(k,L(f_i))+d(j,L(f_i))).\end{aligned}$$

Take (3.6) from (3.7):

$cost(L_1(f_i))-cost(L(f_i))\le -B_{ij}d(j,L(f_i))\le 0.$ □

Proof of Theorem 1: Note that the following problem has been shown NP-hard in [5, 2].

Steiner Tree Problem (STP)

INSTANCE: Given a metric network $N=(V,\ E,\ p)$, a subset $V'\subseteq V$, and an integer l.

QUESTION: Is there a subtree $(V_1,\ E_1)$ of N so that $V'\subseteq V_1$ and $\sum_{e\in E_1} p(e)\le l$?

For each instance $I_1=(N,\ V',\ l)$ of STP where $N=(V,\ E,\ p)$ is a graph and $V=\{j:\ 1\le j\le n\}$ and $V'\subseteq V$, we now construct an instance $I_2=(\overline{N},\ \{U_{ij},Q_{ij},\ 1\le j\le n\},\ K)$ of AP as follows, where $\overline{N}=(\overline{V},\ \overline{E},\ \overline{p})$ is a network:

- $\overline{N}=N$;
- for $1\le j\le n$, $U_{ij}=c$ and $Q_{ij}=nc$ where c is a constant integer if $j\in V'$, and both U_{ij} and Q_{ij} are zero if $j\notin V'$;
- $K=cl|V'|$.

It can be immediately shown that for a solution $(V_1,\ E_1)$ of STP, L such that $L(f_i)=V_1$ is a solution of AP. Also from Lemma 1, it follows that for a solution L of AP, the minimum spanning tree of $L(f_i)\cup V'$ is a solution of STP. Hence the Theorem holds [5]. □

Corollary 1: SDAP is NP-hard. □

3.2. Algorithm SIMPLE

In this section, we present an approximation algorithm, SIMPLE, for SDAP. From Lemma 2, it follows that an allocation L of a fragment f_i with the minimum value of $cost(L(f_i))$ can be viewed as an extension of the data allocation which allocates f_i to those nodes j with

$B_{ij} \geq 0$. This is the basic idea for the development of the algorithm SIMPLE.

For each fragment f_i, the algorithm SIMPLE starts to allocate copies of f_i to those j with $B_{ij} \geq 0$. Then it finds other nodes to allocate copies of f_i to greedily reduce the overall communication cost.

```
Algorithm SIMPLE($\{Q_{ij}, U_{ij}: 1 \leq j \leq n\}, f_i, N$; Var $L$);
Input: $\{Q_{ij}, U_{ij}: 1 \leq j \leq n\}, N, f_i$;
Output: $L$ is an allocation of $f_i$;
{ if $U_i > 0$ then
    { $V_0 := \{j: B_{ij} \geq 0\}$;
    if $V_0 = \emptyset$ then $cost(V_0) = \infty$;
    $V' := V - V_0$;
    for $j = 1$ to $|V| - |V_0|$ do
      { choose a node $l$ in $V'$ such that
        $cost(V_{j-1} \cup \{l\})$ is minimized;
        $V_j := V_{j-1} \cup \{l\}$;
        $V' := V' - \{l\}$; }
    Choose a $V_j$ such that $cost(V_j)$ is minimized;
    $L(f_i) := V_j$; }
  else $L(f_i) = \{j: Q_{ij} > 0\}$; }
```

3.3. Analysis of the Algorithm SIMPLE

Suppose that N is a given network, C is the largest weight of an edge in N, and c is the smallest weight of an edge in N. Further, suppose that F is a given fragment, T is a given transaction set, and for each fragment L_{opt}, L_{opt} is the data allocation of f_i with the minimum value of $cost(L_{opt}(f_i))$.

Clearly, the algorithm SIMPLE may run in polynomial time. Further, we have the following performance guarantee.

Theorem 2: For each fragment f_i, suppose that L is a data allocation given by the algorithm SIMPLE. Then

- $\dfrac{cost(L(f_i))}{cost(L_{opt}(f_i))} \leq \dfrac{C}{c}$.

Proof: Clearly, if $U_i = 0$ then the algorithm SIMPLE will output a data allocation with the minimum communication cost. Below, we prove that this Theorem is true for $U_i > 0$.

Let $V_0 = \{j: B_{ij} \geq 0\}$; and let $L_0(f_i) = V_0$ if $V_0 \neq \emptyset$, otherwise let $L_0(f_i) = \{j\}$ where j is an arbitrary element in $L_{opt}(f_i)$. From Lemma 1, it follows that $cost(L_1(f_i)) = cost(L_{opt}(f_i))$ where $L_1(f_i) = L_{opt}(f_i) \cup V_0$. It is clear that $L_0(f_i) \subseteq L_1(f_i)$.

From the algorithm SIMPLE, we have that $cost(L(f_i)) \leq cost(L_0(f_i))$. We now prove that $\dfrac{cost(L_0(f_i))}{cost(L_1(f_i))} \leq \dfrac{C}{c}$.

Suppose that $|L_0(f_i)| = K_0 + 1$ and $|L_1(f_i)| = K_1 + 1$. By elementary calculation, we have that:

$$cost(L_0(f_i)) \leq \sum_{j \notin L_1(f_i)} (Q_{ij} + U_{ij})C + \sum_{j \in L_1(f) - L_0(f)} (Q_{ij} + U_{ij})C + K_0 U_i C.$$

From the fact that $L_0(f_i)$ contains all nodes j with $B_j(f_i) \geq 0$ and above, it follows that:

$$cost(L_0(f_i)) \leq \sum_{j \notin L_1(f_i)} (Q_{ij} + U_{ij})C + K_1 U_i C.$$

Similarly,

$$cost(L_1(f_i)) \geq \sum_{j \notin L_1(f_i)} (Q_{ij} + U_{ij})c + K_1 U_i c.$$

Hence $\dfrac{cost(L_0(f_i))}{cost(L_1(f_i))} \leq \dfrac{C}{c}$.

Corollary 2: Suppose that the algorithm SIMPLE is applied to each fragment f_i in F to obtain the allocation L_i of f_i. Then in the data allocation L of F such that $L(f_i) = L_i(f_i)$ for each f_i, we have that $\dfrac{\sum_{i=1}^{m} cost(L(f_i))}{\sum_{i=1}^{m} cost(L_{opt}(f_i))} \leq \dfrac{C}{c}$. Further, if the network is uniform then L has the minimum communication cost under the simple strategy. □

Theoretically, the algorithm SIMPLE has a good performance for a *local network*, the network such that $\dfrac{C}{c}$ is small. From our experiments, in practice, this algorithm performs well in a general network.

4. A Heuristic Algorithm: REFINEMENT

The algorithm in [7] usually outputs a better strategy to process a query than the simple strategy; that is, if the simple strategy is the optimal then the algorithm will take the simple strategy. The data allocation output by the algorithm SIMPLE needs to be refined because the strategies to process queries are not necessarily simple. In this section, we present a framework of a refinement algorithm on the data allocation L_0 obtained by the algorithm SIMPLE based on an employed distributed optimizer OP. Suppose that $cost(L, T, OP)$ is the overall communication cost to process T by OP on the data allocation L.

A *local modification* of a data allocation L of F is either:

- for a fragment f_i, drop a copy of f_i from a node j with $j \in L(f_i)$; or

- for a fragment f_i, add a copy of f_i to a node j with $j \notin (f_i)$; or
- for a fragment f_i, remove it from a node j with $j \in L(f_i)$ to a node l with $l \notin L(f_i)$.

A data allocation L of F on N is *locally optimal* with respect to a distributed optimizer *OP* if no local modification will reduce the overall communication cost to process T by *OP*.

The algorithm REFINEMENT iteratively refines L_0 through the choice of a local modification, so that the overall communication cost is greedily reduced, until there is no reduction.

```
Algorithm REFINEMENT($L_0, F, N, OP$; Var $L$);
Input: $F$ is a fragmentation, $N$ is a network with node set $V$,
    $OP$ is a distributed optimizer,
    $L_0$ is a data allocation of $F$ on $N$;
Output: $L$ is a data allocation;
{ $L:=L_0$;
  REPEAT
    $c_0:=cost(L_0, T, OP)$; $c:=c_0$;
    for each fragment $f_i$ do
      { for each $j \in L_0(f_i)$ do
          for each node $k \neq j$ do
            { Locally modify $L_0$ to produce $L_1$ so that
              $L_1(f_i):=L_0(f_i)-\{j\} \cup \{k\}$,
              and so that $L_1(f_l):=L_0(f_l)$ for $f_l \neq f_i$;
              $c_1:=cost(L_1, T, OP)$;
              if $c_1 < c$ then
                { $L:=L_1$; $c:=c_1$ } };
        for $j \in V-L_0(f_1)$ do
          { Locally modify $L_0$ to produce $L_1$
            so that $L_1(f_i):=L_0(f_i) \cup \{j\}$,
            and so that $L_1(f_l):=L_0(f_l)$ for $f_l \neq f_i$;
            $c_1:=cost(L_1, T, OP)$;
            if $c_1 < c$ then
              { $L:=L_1$; $c:=c_1$ } } };
    $L_0:=L$;
  Until $c:=c_0$ (no reduction on $c_0$); }
```

It is clear that in the algorithm REFINEMENT, each iteration runs in polynomial time if *OP* runs in polynomial time for each transaction. Each iteration never increases the overall communication cost, and the algorithm will stop if there is no reduction on the overall communication cost. In practice, we can choose a fixed number as the maximal iteration times. Let T_i be a subset of the given transaction set in which all transactions are required to access fragment f_i. To implement the algorithm REFINEMENT efficiently, for each local modification of fragment f_i, we only need to run the optimizer again for the transactions in T_i instead of the whole T.

From the algorithm REFINEMENT, it immediately follows that:

Theorem 3: The algorithm REFINEMENT produces a locally optimal data allocation of F. □

5. Remarks and Conclusion

This paper studies the data allocation problem with emphasis on minimizing overall communication cost. We have developed an iterative algorithm considering both the generality of a physical network (like other classical research in the file allocation problem) and the generality of transaction processing strategy (like the work in [1] for a uniform network). Several important properties of our algorithms are shown, while the computational intractability of the optimization problems is illustrated.

References

[1] P. M. G. Apers, "Data Allocation in Distributed Database Systems", *ACM Transactions on Database Systems*, Vol. 13, No. 3, page 263-304, 1988.

[2] M. W. Bern and R. L. Graham, "The Shortest-Network Problem", *Scientific American*, page 66-71, January 1989.

[3] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, The Macmillan, 1978.

[4] W. W. Chu, "Optimal File Allocation in a Multiple Computer System", *IEEE Transaction on Computers*, Vol. C-13, No. 10, October 1969.

[5] M. R. Garey and D. S. Johnson, *Computer and Intractability-A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1978.

[6] S. Navathe, etc., "Vertical Partitioning Algorithms for Database Design", *ACM Transactions on Database systems*, Vol. 9, No.4, page 680-303 1984.

[7] M. W. Orlowski, "On Optimisation of Joins in Distributed Database System", *Future Databases 92*, World Scientific, page 106-114, 1992.

[8] D. Sacca and G. Wiederhold, "Database partitioning in a cluster of processors", *ACM Transactions on Database Systems*, Vol. 10, No. 1, page 28-56, 1985.

[9] R. Sedgewick, *Algorithms*, Addision-Wesley, 1988.

[10] Y. Zhang, M. Orlowska and B. Colomb, "An Efficient Test for the Validity of Hybrid Knowledge Fragmentation in Distributed Databases", *Intl. J. of Software Engineering and Knowledge Engineering*, Vol. 2, No. 4, 1992.

[11] C. J. Date, *An Introduction to Database System*, Vol 2. Addition-Wesley, 1982.

[12] D. Maier, *Theory of Relational Databases*, Computer Science Press, 1983.