The 1st International Conference on Distributed Computing Systems, October 1-5, 1979, Huntsville, Alabama.

THE PLACEMENTS OF RELATIONS ON A DISTRIBUTED RELATIONAL DATA BASE

C. V. Ramamoorthy and Benjamin W. Wah
Computer Science Division, Department of
Electrical Engineering and Computer Sciences and the
Electronics Research Laboratory,
University of California, Berkeley, CA 94720.

ABSTRACT

In this paper, the problem of optimal placements of relations on a distributed relational data base is studied. It is found that this problem can be decomposed into multiple sub-problems of optimizing the placements of individual relations. A technique is proposed to introduce redundant information onto the distributed data base so that non-decomposable queries can be made decomposable. As a result, the operational costs are found to decrease when sufficient redundant information is added. A simple example is used to illustrate the technique.

I. *INTRODUCTION*

The recent advances in large-scale integrated logic and communication technology, coupled with the explosion in size and complexity of the application areas, have led to the design of distributed architectures. Basically, a *Distributed Computer System (DCS)* is considered as an interconnection of digital systems called *Processing Elements (PE's)*, each having certain processing capabilities, communicating with each other through an interconnection network and working on a set of jobs, which may be related or unrelated [RAM76, AND75]. This definition encompasses a wide range of configurations from an uniprocessor system with different functional units to a multiplicity of general purpose computers (e.g. ARPANET).

Data on a DCS are managed through a *Data Base (DB)* which is a collection of stored operational data used by the application systems of some particular enterprise [DAT77]. A *Distributed Data Base (DDB)* can be regarded as the data stored at different locations of a DCS. It can be considered to exist only when data elements at multiple locations are interrelated and/or there is a need to access data stored at some locations from another location.

Because of the availability of many parallel resources on a DCS, and the increasing need for larger data bases, the design of efficient coordination schemes for the management of data on a DCS is a very critical problem. In this paper, the problem of optimal placements of relations on a distributed relational DB is studied. The objective of the problem is to place multiple copies of a relation on the DCS so as to minimize the total operational costs of the system which may include storage cost, multiple update cost, retrieval cost, query processing cost and file migration cost (if the assignments are dynamic). The theme of this paper is to demonstrate that the placements of multiple relations on a distributed relational DB can be optimized for each relation independently. It is assumed that a technique exists to find the optimal placements of multiple copies of a single file on a DDB. There are many techniques available, e.g. [CAS72, LEV74, WAH79]. We have shown in [WAH79, RAM79b] that the placements of multiple copies of a single file is isomorphic to the single commodity warehouse location problem. Based on this property, many techniques developed for both problems are interchangeable. Among these are algorithms developed in the warehouse location problem, such as the add-drop algorithm, the branch and bound algorithms, the probabilistic branch and bound algorithm, the integer programming technique, the steepest ascent algorithm and the dynamic programming methods. These algorithms can be applied to solve the (dynamic) file allocation problem. On the other hand, there are algorithms developed in the file allocation problem which can be used to solve the warehouse location problem. These include the hyper-cube technique, the clustering technique, the dynamic programming methods and the max-flow min-cut network flow technique.

A technique is proposed in this paper in which redundant information is added to the DDB which further reduces the operational costs of the system. It is shown by an example DDB that under certain conditions, the technique can reduce the total operational costs of the system. A relational data model is chosen in this paper because it is very popular and the results obtained would be more specific. However, the technique proposed in this paper can be generalized to any type of data model and file system.

II. *QUERIES ON A RELATIONAL DB*

In a relational DB [COD70], data is viewed as relations of varying degree, the degree being the number of distinct domains participating in the relation. Each instance of a relation is known as a tuple, which has a value for each domain of the relation. Thus a relation can simply be represented in tabular form with columns as domains and rows as tuples.

A *Query* is an access request made by a user or a program, in which one or more relations have to be accessed. A query on a relational DB consists of two parts: the part specifying the domains of the relation to be retrieved and the part specifying the predicate which is a quantification representing the defining properties of the set to be accessed. Let S be a relation of domains s#, sname, city, inventory; and SP be a relation of domains s#, p# (Figure 1). The queries on a relational DB can be classified into the following categories [DAT77]:

(1) Retrieval Operations
   (a) Single Relation Retrieval: The predicate representing the defining property of the set to be retrieved is defined on the same relation as the set.

Research was supported partially by Ballistic Missile Defense Contract DASG60-77-C-0138 and ARO contract DAAG29-78-G-0189. Preparation of this paper was supported in part by National Science Foundation under grant MCS 78-07291.

(a) Relation S

S	s#	sname	city	inventory
	1	Supplier A	New York	1500
	3	Supplier B	San Francisco	700
	5	Supplier C	Chicago	2500

(b) Relation SP

SP	s#	p#
	1	A1
	2	A1
	3	A2
	4	A2
	5	P2

Figure 1 Relations S and SP

E.g. GET (S.sname): (S.city="Paris" AND S.inventory>1000)

(b) Multiple Relation Retrieval: The predicate, as well as the set to be retrieved, may be defined over multiple relations.
E.g. GET (S.sname): (S.s#=SP.s# AND SP.p#="P_2")
Relation S and SP must be available simultaneously before the retrieval can be processed.

(2) Storage Operations
(a) Single Relation Update;
(b) Multiple Relation Update;
(c) Insertion;
(d) Deletion.

(3) Library Functions
These represent more complicated operations on the predicate than the equality operations, e.g. counting the number of occurences, selecting the maximum/minimum etc.

Single relation queries can be processed very easily on a distributed relational DB. When the relation is geographically distributed, the query can be sent to a node in which a copy of the relation resides and be processed there. The results after the processing can be sent back to the requesting node. It is generally true that the amount of communications needed to transmit the results is much smaller than the amount needed to transmit the relations.

On the other hand, the processing of a mult-relation query is more complicated. When multiple relations are accessed by the same query on a DDB, these relations usually have to reside at a common location before the query can be processed. Substantial communication overhead may be involved if these relations are geographically distributed and a copy of each relation has to be transferred to a common location. It is therefore necessary to decompose the query into sub-queries so that each sub-query accesses a single relation. This technique has been proposed in the design of the centralized version of INGRES [WON76], and is extended to the design of SDD-1 [WON77] and distributed INGRES [EPS78]. Specifically, the technique consists of two steps. The first step is to select a site with the minimum amount of data movements to that site before the query can be processed. This is used as a starting point for the second step of the algorithm which determines the sequence of moves that results in a minimum cost. The algorithm used is a greedy algorithm and only local optima can result in such an algorithm. Hevner and Yao [HEV79] have followed a similar approach and have developed two optimal algorithms for arranging data transmissions and local data processing with minimal response time and minimal total time, for a special class of queries. These optimal algorithms are used as a basis to develop a general query processing algorithm for a general query in which each required relation may have any number of joining domains and output domains and each node may have any number of required relations. This general algorithm is a heuristic which uses an improved exhaustive search to find efficient distribution strategies. Ghosh also proposed a model of data distribution on a DB which facilitates query processing [GHO76]. Specifically, the model consists of a DB with multiple target segment types and there are queries with multiple target segment types. The objective is to distribute the segments on the DB so as to maximize the number of segments that the queries can retrieve in parallel from different nodes. The model only looks at the problem from a retrieval point of view and no cost is associated with retrieving a segment from a node.

Most of the previous work address the problem from a viewpoint of what are the processing sequence of the query and where it should be processed. However, they do not consider the distribution of files which can make the processing of queries more efficient. Further, there exists queries which are non-decomposable. For example, the query:

GET (S.sname): (S.s#=SP.s# AND SP.p#="P_2")

is not decomposable into single relation retrievals because there is a logical relation "=" which is defined over a common domain s# of the relation S and SP. These relations must be available simultaneously at a common location before the retrieval or update operations can be performed. Instead of solving the problem of decomposing the queries, we study a technique to reduce the processing and communication costs for non-decomposable queries in this paper. It is shown later, by the introduction of some redundant information on the DB, non-decomposable queries may be made decomposable, (see also [RAM79a]). The basic assumption made over here is that all the required relations are moved to the node at which the query originates, before the processing of the query begins. It is possible to consider a sequence of moves which will minimize the total amount of data transferred. However the problem will be very complicated and the intention of this paper is to demonstrate the usefulness of the technique of using redundant information.

Before the technique is discussed, the problem of placements of relations on a DDB is first formulated.

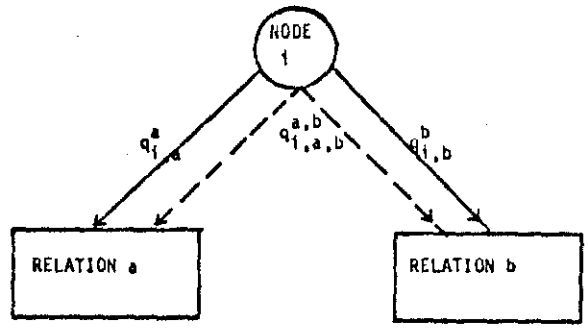
III. *THE PLACEMENTS OF RELATIONS ON A DDB*

In this section, a model for the placements of multiple relations on a DDB is formulated. The model is shown for the special case of two relations and is generalized later to the case of more than two relations.

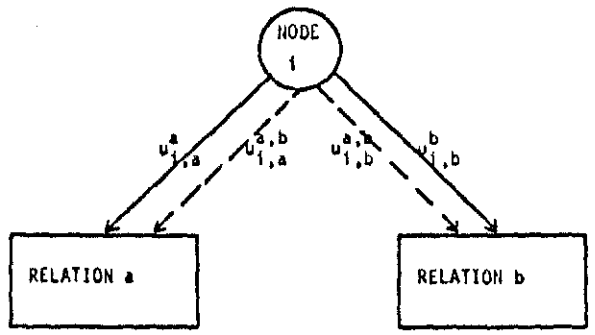
Consider two relations a and b, the retrieval and update rate at node i are (see Figure 2)[1]:

$q_{i,a}^{a}(q_{i,b}^{b})$ = rate of access at node i for a single relation retrieval accessing relation a(b);

$q_{i,a,b}^{a,b}$ = rate of access at node i for a multi-relation retrieval accessing both relations a and b;



a) RETRIEVALS



b) UPDATES

⟶ SINGLE RELATION ACCESSES
⟶ MULTI-RELATION ACCESSES

Figure 2 Retrieval and Update Rate on a 2-Relation DB from Node i

$u_{i,a}^{a}(u_{i,b}^{b})$ = rate of update at node i for a single relation query updating relation a(b);

$u_{i,a}^{a,b}(u_{i,b}^{a,b})$ = rate of update at node i for a multi-relation query accessing both relations a and b before updating relation a(b).

The costs for each unit of access are:

$S_{i,j}^{a}(S_{i,j}^{b})$ = communication and processing cost per unit query of accessing relation a(b) from node i to node j;

$M_{i,j}^{a}(M_{i,j}^{b})$ = communication and processing cost per unit update of multiple updating relation a(b) from node i to node j.

We differentiate between the costs of retrievals and updates because in some applications, retrievals are more important than updates and therefore would have a higher cost (e.g. inventory system); while in other real time applications, updates may be more frequent and therefore more critical (e.g. airline reservation system).

[1] The convention of the symbols used are as follows: i,j represent indexes for nodes; a,b represent indexes for relations; the superscripts represent the list of relations that the query must access before the query can be processed; the subscripts represent the nodes concerned and the target list of relations for the query.

Let:

n = number of nodes on the DCS;

$l_a(l_b)$ = length of relation a(b);

$f_{i,a}(f_{i,b})$ = per unit cost of storing relation a(b) at node i.

We define from the characteristics of the queries initiated from node i, the following symbols:
(1) Single relation retrievals:

$\alpha_{i,a}^{a}(\alpha_{i,b}^{b})$ = fraction of relation a(b) that is put into the result relation due to the execution of a single relation retrieval on a(b);

(2) Multi-relation retrievals:

$\alpha_{i,a}^{a,b}(\alpha_{i,b}^{a,b})$ = fraction of relation a(b) that is needed to process a multi-relation retrieval on a and b;

(3) Single relation updates:

$\beta_{i,a}^{a}(\beta_{i,b}^{b})$ = fraction of relation a(b) that will be updated by a single relation update;

(4) Multi-relation updates:

$\nu_{i,a}^{a,b}(\nu_{i,b}^{a,b})$ = fraction of relation a(b) that is needed to process a multi-relation update before the updates can be performed;

$\beta_{i,a}^{a,b}(\beta_{i,b}^{a,b})$ = fraction of relation a(b) that will be updated by a multi-relation update after relations a and b have been accessed.

In processing a multi-relation update, the relations a and b must be accessed first in order to determine what are the actual updates that have to be made. This is measured by the parameters $\nu_{i,a}^{a,b}$ and $\nu_{i,b}^{a,b}$. The fraction of relations a and b to be updated after they have been determined are measured by the parameters $\beta_{i,a}^{a,b}$ and $\beta_{i,b}^{a,b}$.

The parameters defined above can be estimated from the characteristics of the different types of queries that can be made on the DDB and the distribution of the data stored in the relations.

The control variables governing the file locations and the routing discipline are defined as follows:

$$Y_i^a(Y_i^b) = \begin{cases} 0 & \text{if } \textit{relation } a\,(b) \textit{ does not exist at node } i \\ 1 & \textit{otherwise} \end{cases}$$

$X_{i,j}^{a}(X_{i,j}^{b})$ = fraction of queries made at node i on relation a(b) that are routed to node j.

It is true that if $X_{i,j}^{r}>0$, then $Y_j^r=1$ for r=a,b.

The optimization problem of placing relations a and b on the DDB can be formulated in the following linear program:

min (1)

$$\sum_{r=a,b}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{i,r}^{r}\alpha_{i,r}^{r}l_r X_{i,j}^{r}S_{i,j}^{r} \qquad (1a)$$

$$+\sum_{r=a,b}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{i,a,b}^{a,b}\alpha_{i,r}^{a,b}l_r X_{i,j}^{r}S_{i,j}^{r} \qquad (1b)$$

$$+\sum_{r=a,b}\sum_{i=1}^{n}\sum_{j=1}^{n} u_{i,r}^{r}\beta_{i,r}^{r}l_r M_{i,j}^{r}Y_j^r \qquad (1c)$$

$$+ \sum_{r=a,b} \sum_{i=1}^{n} \sum_{j=1}^{n} u_{i,r}^{a,b} \left[\sum_{s=a,b} \nu_{i,j}^{s,b} l_s X_{i,j}^{s} S_{i,j}^{s} + \beta_{i,r}^{a,b} l_r M_{i,j} Y_j^r \right] \quad (1d)$$

$$+ \sum_{r=a,b} \sum_{i=1}^{n} f_{i,r} l_r Y_i^r \quad (1e)$$

subject to the following constraints:

$$\sum_{i=1}^{n} Y_i^r \geq 1 \qquad r=a,b \quad (1f)$$

$$\sum_{j=1}^{n} X_{i,j}^r = 1, \qquad r=a,b,\ i=1,2,\ldots,n \quad (1g)$$

$$nY_j^r \geq \sum_{i=1}^{n} X_{i,j}^r \geq 0, \quad r=a,b,\ j=1,2,\ldots n \quad (1h)$$

$$Y_i^r = 0,1, \quad r=a,b,\ i=1,2,\ldots,n \quad (1i)$$

Eq. 1a represents the access cost for single relation retrievals; Eq. 1b represents the access cost for multi-relation retrievals; Eq. 1c represents the update cost for single relation updates; Eq. 1d represents the update cost for multi-relation updates and Eq. 1e represents the storage cost of relations on the DDB. Condition 1f assures that at least one copy of the relation exists; condition 1g assures that all the queries are serviced; condition 1h assures that the relation must exist at a node if a route is defined to access it at that node and condition 1i assures that the control variables Y_i^r are integral.

LEMMA 1
The above optimization problem can be partitioned into two independent optimization sub-problems for each relation:

(a) min (2)

$$\sum_{i=1}^{n} \sum_{j=1}^{n} Q_i^a X_{i,j}^a S_{i,j}^a + \sum_{i=1}^{n} \sum_{j=1}^{n} U_i^a M_{i,j}^a Y_j^a + \sum_{i=1}^{n} F_i^a Y_i^a$$

where

$$Q_i^a = (q_{i,a}^a \alpha_{i,a}^a + q_{i,a,b}^{a,b} \alpha_{i,a}^{a,b} + u_{i,a}^{a,b} \nu_{i,a}^{a,b} + u_{i,b}^{a,b} \nu_{i,a}^{a,b}) l_a$$

$$U_i^a = (u_{i,a}^a \beta_{i,a}^a + u_{i,a}^{a,b} \beta_{i,a}^{a,b}) l_a$$

$$F_i^a = f_{i,a} l_a$$

subject to:

$$\sum_{i=1}^{n} Y_i^a \geq 1$$

$$\sum_{j=1}^{n} X_{i,j}^a = 1 \qquad i=1,\ldots,n$$

$$nY_j^a \geq \sum_{i=1}^{n} X_{i,j}^a \geq 0 \qquad j=1,\ldots,n$$

$$Y_i^a = 0,1 \qquad i=1,\ldots,n$$

(b) min (3)

$$\sum_{i=1}^{n} \sum_{j=1}^{n} Q_i^b X_{i,j}^b S_{i,j}^b + \sum_{i=1}^{n} \sum_{j=1}^{n} U_i^b M_{i,j}^b Y_j^b + \sum_{i=1}^{n} F_i^b Y_i^b$$

where

$$Q_i^b = (q_{i,b}^b \alpha_{i,b}^b + q_{i,a,b}^{a,b} \alpha_{i,b}^{a,b} + u_{i,a}^{a,b} \nu_{i,b}^{a,b} + u_{i,b}^{a,b} \nu_{i,b}^{a,b}) l_b$$

$$U_i^b = (u_{i,b}^b \beta_{i,b}^b + u_{i,b}^{a,b} \beta_{i,b}^{a,b}) l_b$$

$$F_{i,b} = f_{i,b} l_b$$

subject to:

$$\sum_{i=1}^{n} Y_i^b \geq 1$$

$$\sum_{i=1}^{n} X_{i,j}^b = 1 \qquad i=1,\ldots,n$$

$$nY_j^b \geq \sum_{i=1}^{n} X_{i,j}^b \geq 0 \qquad j=1,\ldots,n$$

$$Y_i^b = 0,1 \qquad i=1,\ldots,n$$

Proof
We notice in optimization problem (1) that there are no cross product terms in the control variables of relations a and b. Therefore, the objective function of (1) can be written as a sum of objective functions of optimization problems (2) and (3), and similarly, the constraints can be partitioned into two independent sets. The solution to (2) will therefore be a constant in (1) which implies that (3) can be solved independently. Similarly, the solution to (3) will be a constant in (1) and this implies that (2) can be solved independently.

We conclude that the optimization problem for relations a and b can be carried out as two optimization sub-problems for relations a and b independently.

A further simplification of the integer programs (2) and (3) is to first solve for $X_{i,j}^r$, $r=a,b$, and substitute into the integer programs. It is shown in [ALC76] that,

$$X_{i,j}^r = \begin{cases} 1 & \text{if } S_{i,j}^r = \min\limits_{k,Y_k^r=1} S_{i,k} \\ 0 & \text{otherwise} \end{cases}$$

The detailed proof will not be shown here.

A generalization of Lemma 1 is to allow any number of relations in the DDB. This is shown in the following theorem.

THEOREM 1
The general problem of optimizing the placements of multiple relations on a DDB can be decomposed into multiple sub-problems of optimizing the placements of each relation independently.

The proof, which require some symbols to be defined and can be done by obvious generalization of the proof of Lemma 1, will not be shown here.

The importance of Theorem 1 is that the original optimization problem of placing multiple copies of m relations on the DDB, which has a complexity of the order of $O(2^{nm})$, is reduced to m simpler optimization sub-problems of placing multiple copies of each relation on the DDB, each of which has a complexity of the order of $O(2^n)$. We will not study the algorithm for deciding the placements of multiple copies of a relation on a DDB. There are many techniques available, e.g. [CAS72, LEV74, MOR77, WAH79]. Some of these techniques are exhaustive and give optimal solutions, e.g. [CAS72, LEV74, MOR77]; others give sub-optimal solutions and have a polynomial running time, e.g. [WAH79]. We describe in the next section a technique to reduce the cost of the objective function of (1) by the use of redundant information.

IV. *COST REDUCTION ON THE PLACEMENTS OF RELATIONS ON A DDB BY UTILIZING REDUNDANT INFORMATION*

In section II, the technique of query decomposition is briefly described. In query decomposition, optimization is performed on the processing of a single query which originates at a node. The objective is to decompose a multi-relation query into as many single relation sub-queries as possible so that data (relation) movements from one node to another can be minimized. However, there exists non-decomposable queries which require all the relations that they access to be present at a common location. A large number of relation transfers may be needed if these relations are geographically distributed. In order to avoid these extra relation transfers, a technique utilizing redundant information is proposed here. Instead of decomposing queries that access multiple relations, it may be sufficient to provide redundant information in each relation so that multiple relations do not need to reside at a single location before the query can be processed. For example, in processing the query:

GET (S.sname): (S.s#=SP.s# AND SP.p#="P_2")

on two geographically separated relations, S and SP (Figure 1), it may be necessary to transfer relation S to the node where SP resides and then process the query there or vice versa. However, if the information (S.s#=SP.s#) is compiled beforehand into the two relations (Figure 3), then the above query can be decomposed into two single relation sub-queries:

GET (S.s#, S.sname): (S.s#=SP.s#) and
GET (SP.s#): (S.s#=SP.s# AND SP.p#="P_2").

In this case, the processing can be done in parallel and the amount of information transfers is much smaller.

This technique poses several problems. First, it is necessary to take one extra bit for each tuple in order to compile this piece of information. If the amount of information to be added is large, (e.g. when the number of different predicates defined on a common domain of two relations is large), the size of the extra storage space may be significant. Second, when the common domain of one relation is modified, it is necessary to "multiple update" the redundant information in all the common domains of the other relations in the DDB. Referring to Figure 3, if an extra tuple with s#="2", sname="Supplier D", city="Boston" and inventory="3000" is added to relation S, then it is necessary to find out what are the changes that have been made on the redundant information (s.s#=SP.s#) in both relations S and SP, and to update these changes in addition to the original update. In this case, the (S.s#=SP.s#) information has to be changed in relations S and SP because relation SP contains a tuple with s#=2. If updating activities are frequent, the "multiple update" cost is large. The net effect of this technique is therefore to reduce the total retrieval cost and increase the total update cost of the system. Further, the response time in reflecting an update on the DB may be longer in this case because of the need to update the redundant information. Third, this technique requires that the DB designer be able to estimate the amount of additional information to be compiled into the relations. A possible technique is to pre-analyze the type of predicates used in retrievals and updates and to determine what are the essential information to be compiled into the relations. A compromize should be made between introducing extra information with additional storage space and higher cost in multiple updates, and reducing the amount of relation transfers. It would be advantageous for the more frequently used predicates and less advantageous for the others.

(a) Relation S

S	s#	S.s#= SP.s#	sname	city	inventory
	1	1	Supplier A	New York	1500
	3	1	Supplier B	San Francisco	700
	5	1	Supplier C	Chicago	2500

(b) Relation SP

SP	s#	S.s#= SP.s#	p#
	1	1	A1
	2		A1
	3	1	A2
	4		A2
	5	1	P2

Figure 3 Relations S and SP with (S.s#=SP.s#) information compiled into the relations

In the remainder of this section, a model is developed for deciding how much redundant information is needed on the DDB in order for this technique to be cost effective. We first examine the strategies that have to be used for retrievals and updates.

The strategies on retrievals of a geographically distributed relation is the same as the strategy when no redundant information is used. The necessary information to be used in processing the query is first projected onto temporary files before they are sent to the originating node. On the other hand, the strategy on updates is different from the case of no redundant information because it is also necessary to check whether the redundant information is updated. There are two variations of the update strategy:

(1) The updates are first sent to the multiple copies of the file to be updated;
The necessary information on all the relations, which is needed to determine if the redundant information has to be updated, is sent to a common node;
The updates to be made on the redundant information are determined there;
The updates on the redundant information are sent out to all the affected relations.

(2) The necessary information on all the relations, which is needed to determine if the redundant information has to be updated, is sent to node i where the update originates, (actually, it can be sent to any other node, but the control overhead in doing this would usually be greater);
The update to be made on the redundant information are determined at this node;
The updates on the target relation as well as the updates on the redundant information, are sent out to all the relations.

The advantage of using strategy (1) is that the updates on the target relation will be reflected on the DB in a shorter time than strategy (2). But strategy (1) involves more control overhead and the response time in

reflecting the updates on the redundant information will be longer than strategy (2). In general, strategy (2) will have a shorter overall response time. We assume that strategy (2) is used in our model.

As before, the model for determining the use of redundant information is first developed for the special case of two relations and is generalized to the case of more than two relations later.

Consider two relations a and b, the retrieval and the update rates, using the notations defined earlier, are shown in Figure 4. There are two additional types of single relation retrievals. These are originally multi-relation retrievals. Due to the use of redundant information, part of the multi-relation retrievals are decomposed into single relation retrievals. In describing the model, the following symbols are defined:

$\gamma_{i,a,b}^{a,b}$ = fraction of non-decomposable multi-relation retrievals on a and b from node i that remain non-decomposable even with the use of redundant information;

$\frac{\sigma_{i,a}^{a,b}}{(\sigma_{i,a}^{a,b}+\sigma_{i,b}^{a,b})}$ $\left(\frac{\sigma_{i,b}^{a,b}}{(\sigma_{i,a}^{a,b}+\sigma_{i,b}^{a,b})}\right)$

= fraction of multi-relation-reduced-single-relation retrievals from node i on a(b) due to the use of redundant information;
$(1-\gamma_{i,a,b}^{a,b})q_{i,a,b}^{a,b}$ is the original rate of multi-relation retrievals that is decomposable with the use of redundant information;
$(1-\gamma_{i,a,b}^{a,b})q_{i,a,b}^{a,b}(\sigma_{i,a}^{a,b}+\sigma_{i,b}^{a,b})$ is the total rate of multi-relation-reduced-single-relation retrievals to relations a and b after the decomposition;
It is generally true that $\sigma_{i,a}^{a,b}+\sigma_{i,b}^{a,b}\geq 1$, that is, the total rate of additional single relation retrievals after the use of redundant information, is greater than the reduction in multi-relation retrieval rate;
The access rate of multi-relation-reduced-single-relation retrievals on relation r is $(1-\gamma_{i,a,b}^{a,b})q_{i,a,b}^{a,b}\sigma_{i,r}^{a,b}$ for r=a,b ;

$\varepsilon_{i,a}^{a,b}(\varepsilon_{i,b}^{a,b})$ = fraction of relation a(b) that is the result to a multi-relation-reduced-single-relation retrieval on a(b);

$\delta_{i,a}^{a,b}(\delta_{i,b}^{a,b})$ = fraction of non-decomposable multi-relation updates on a(b) from node i that remain non-decomposable even with the use of redundant information;

$\eta_{i,a,b}^{a}(\eta_{i,a,b}^{b})$ = fraction of updates on relation a(b) from node i that will update redundant information on relations a and b;

$\xi_{i,a}^{a}(\xi_{i,b}^{b})$ = fraction of relation a(b) in which the redundant information has to be updated due to updates originating from node i;

$l'_a(l'_b)$ = length of relation a(b) after the use of redundant information.

In our model, although the amount of storage is greater after redundant information is used, i.e. $l'_r>l_r$ (r=a,b), but the effect on communication is very small because the redundant information does not have to be transferred over the network in processing a query.

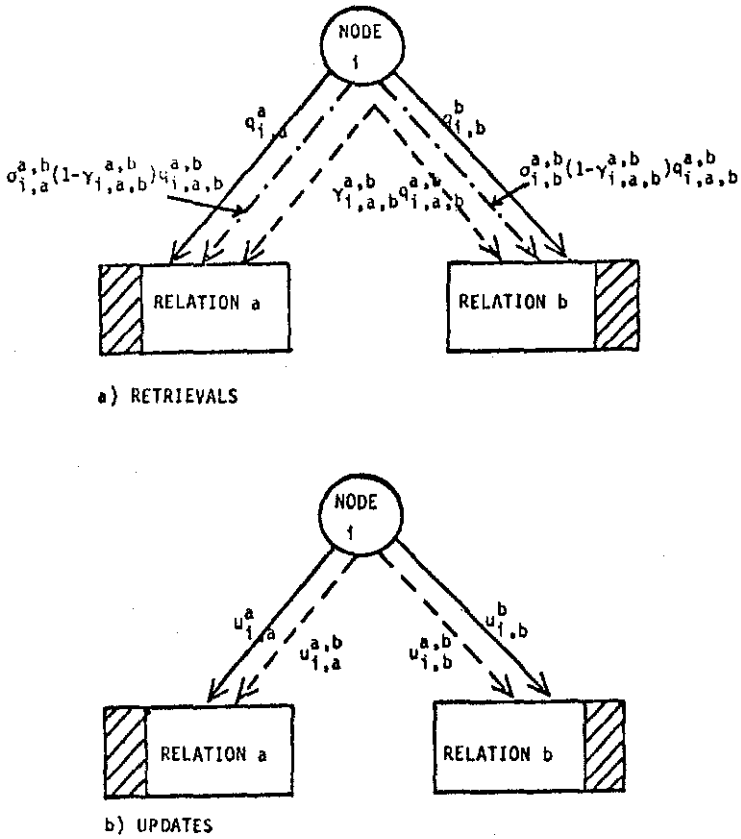


Figure 4 RETRIEVAL AND UPDATE RATE ON A 2-RELATION DATA BASE FROM NODE i USING ADDITIONAL REDUNDANT INFORMATION

The optimization problem of placing relations a and b on the DDB after the use of redundant information can be formulated in the following linear-program:

min (4)

$$\sum_{r=a,b}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{i,r}^{r}\alpha_{i,r}^{r}l'_r X_{ij}^{r}S_{ij}^{r} \quad (4a)$$

$$+\sum_{r=a,b}\sum_{i=1}^{n}\sum_{j=1}^{n}(1-\gamma_{i,a,b}^{a,b})q_{i,a,b}^{a,b}\sigma_{i,r}^{a,b}\varepsilon_{i,r}^{a,b}l'_r X_{ij}^{r}S_{ij}^{r} \quad (4b)$$

$$+\sum_{r=a,b}\sum_{i=1}^{n}\sum_{j=1}^{n}\gamma_{i,a,b}^{a,b}q_{i,a,b}^{a,b}\alpha_{i,r}^{a,b}l'_r X_{ij}^{r}S_{ij}^{r} \quad (4c)$$

$$+\sum_{r=a,b}\sum_{i=1}^{n}\sum_{j=1}^{n}u_{i,r}^{r}\beta_{i,r}^{r}l'_r M_{ij}^{r}Y_{j}^{r} \quad (4d)$$

$$+ \sum_{r=a,b} \sum_{i=1}^{n} \sum_{j=1}^{n} u_{i,r}^{a,b} \left[\delta_{i,r}^{a,b} \sum_{s=a,b} \nu_{i,s}^{a,b} l'_s X_{i,j}^{s} S_{i,j}^{s} + \beta_{i,r}^{a,b} l'_r M_{i,j} Y_j^r \right] \quad (4e)$$

$$+ \sum_{r=a,b} \sum_{i=1}^{n} \sum_{j=1}^{n} \eta_{i,a,b}^{r} (u_{i,r}^{r} + u_{i,r}^{a,b}) \left[\sum_{s \neq r} \alpha_{i,s}^{a,b} l'_s X_{i,j}^{s} S_{i,j}^{s} + \sum_{t=a,b} \xi_{i,t}^{r} l'_t M_{i,j} Y_j^t \right] \quad (4f)$$

$$+ \sum_{r=a,b} \sum_{i=1}^{n} f_{i,r} l'_r Y_i^r \quad (4g)$$

subject to:

$$\sum_{i=1}^{n} Y_i^r \geq 1 \qquad r=a,b$$

$$\sum_{j=1}^{n} X_{i,j}^{r} = 1 \qquad r=a,b \quad i=1,\ldots,n$$

$$nY_j^r \geq \sum_{i=1}^{n} X_{i,j}^{r} \geq 0 \qquad r=a,b \quad j=1,\ldots,n$$

$$Y_i^r = 0,1 \qquad r=a,b \quad i=1,\ldots,n$$

Most of the terms in Eq. 4 are the same as in Eq. 1, except in this case Eq. 4b represents the access cost for multi-relation-reduced-single-relation retrievals using the redundant information; and Eq. 4f represents the update cost for the redundant information. The term $\eta_{i,a,b}^{r}(u_{i,r}^{r}+u_{i,r}^{a,b})$ for r=a,b represents the access rate of updates that may have effects on the redundant information. In determining whether the redundant information will be updated, it is necessary to perform a multi-relation retrieval on the relations concerned. In this case, since we know the updates to be made on relation r, we can fetch a copy of all other relations $s \neq r$ and move the copy to node i. This cost is represented by the term $\sum_{s \neq r} \alpha_{i,s}^{a,b} l'_s X_{i,j}^{s} S_{i,j}^{s}$ in Eq. 4f. After the updates on the redundant information has been determined, the actual updates, together with the updates on the redundant information are sent to all the nodes which have a copy of the relation. This cost is represented by the term $\sum_{t=a,b} \xi_{i,t} l'_t M_{i,j} Y_j^t$ in Eq. 4f.

A similar lemma and theorem can be proved for this problem.

LEMMA 2

Optimization problem 4 can be partitioned into two independent optimization sub-problems for each relation:

$$(a) \min \sum_{i=1}^{n} Q_i^a \min_{j, Y_j^a=1} S_{i,j}^{a} + \sum_{i=1}^{n} \sum_{j=1}^{n} U_i^a M_{i,j}^{a} Y_j^a + \sum_{i=1}^{n} F_i^a Y_i^a \quad (5)$$

where:

$$Q_i^a = [q_{i,a}^{a} \alpha_{i,a}^{a} + (1-\gamma_{i,a,b}^{a,b}) q_{i,a,b}^{a,b} \sigma_{i,a}^{a,b} e_{i,a}^{a,b} + \gamma_{i,a,b}^{a,b} q_{i,a,b}^{a,b} \alpha_{i,a}^{a,b} + u_{i,a}^{a,b} \delta_{i,a}^{a,b} \nu_{i,a}^{a,b} + u_{i,b}^{a,b} \delta_{i,b}^{a,b} \nu_{i,a}^{a,b} + \eta_{i,a,b}^{a} (u_{i,b}^{a} + u_{i,b}^{a,b}) \alpha_{i,a}^{a,b}] l'_a$$

$$U_i^a = [u_{i,a}^{a} \beta_{i,a}^{a} + u_{i,a}^{a,b} \beta_{i,a}^{a,b} + \eta_{i,a,b}^{a} (u_{i,a}^{a} + u_{i,a}^{a,b}) \xi_{i,a}^{a} + \eta_{i,a,b}^{b} (u_{i,b}^{a} + u_{i,b}^{a,b}) \xi_{i,a}^{a}] l'_a$$

$$F_i^a = f_{i,a} l'_a$$

subject to:

$$\sum_{i=1}^{n} Y_i^a \geq 1$$

$$Y_i^a = 0,1 \qquad i=1,\ldots,n$$

$$(b) \min \sum_{i=1}^{n} Q_i^b \min_{j, Y_j^b=1} S_{i,j}^{b} + \sum_{i=1}^{n} \sum_{j=1}^{n} U_i^b M_{i,j}^{b} Y_j^b + \sum_{i=1}^{n} F_i^b Y_i^b \quad (6)$$

where:

$$Q_i^b = [q_{i,b}^{b} \alpha_{i,b}^{b} + (1-\gamma_{i,a,b}^{a,b}) q_{i,a,b}^{a,b} \sigma_{i,b}^{a,b} e_{i,b}^{a,b} + \gamma_{i,a,b}^{a,b} q_{i,a,b}^{a,b} \alpha_{i,b}^{a,b} + u_{i,a}^{a,b} \delta_{i,a}^{a,b} \nu_{i,b}^{a,b} + u_{i,b}^{a,b} \delta_{i,b}^{a,b} \nu_{i,b}^{a,b} + \eta_{i,a,b}^{a} (u_{i,a}^{a} + u_{i,a}^{a,b}) \alpha_{i,b}^{a,b}] l'_b$$

$$U_i^b = [u_{i,b}^{b} \beta_{i,b}^{b} + u_{i,b}^{a,b} \beta_{i,b}^{a,b} + \eta_{i,a,b}^{a} (u_{i,a}^{a} + u_{i,a}^{a,b}) \xi_{i,b}^{b} + \eta_{i,a,b}^{b} (u_{i,b}^{a} + u_{i,b}^{a,b}) \xi_{i,b}^{b}] l'_b$$

$$F_i^b = f_{i,b} l'_b$$

subject to:

$$\sum_{i=1}^{n} Y_i^b \geq 1$$

$$Y_i^b = 0,1 \qquad i=1,\ldots,n$$

THEOREM 2

The general problem of optimizing the placements of multiple relations on a DDB using additional redundant information can be decomposed into multiple sub-problems of optimizing the placements of each relation independently.

The proofs of Lemma 2 and Theorem 2 are very similar to that of Lemma 1 and Theorem 1 and will not be illustrated here.

We demonstrate the use of this technique in the next section with a simple example.

V. *A NUMERICAL EXAMPLE TO ILLUSTRATE THE USE OF REDUNDANT INFORMATION ON A DDB*

In this section, we show by the use of a numerical example, the cost improvement when redundant information is introduced on the DDB.

Consider a DCS of 3 nodes with two relations, S and SP, on the DDB. Let S has domains s#(1), sname(10), city(5), inventory(2) and SP has domains s#(1), p#(1)[1]. Assume that S has 500 tuples and SP has 10000 tuples. The following parameters are also assumed:

$$[S_{i,j}] = [M_{i,j}] = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1.5 \\ 2 & 1.5 & 0 \end{bmatrix} * 10^{-3}$$

$f_{i,S} = f_{i,SP} = 0$
$l_S = l'_S = 500*18 = 9000$ (words)[2]
$l_{SP} = l'_{SP} = 10000*2 = 20000$ (words)[2]

Node	Parameters						
i	$q_{i,S}^{S}$	$u_{i,S}^{S}$	$u_{i,S}^{S,SP}$	$q_{i,SP}^{SP}$	$u_{i,SP}^{SP}$	$u_{i,SP}^{S,SP}$	$q_{i,S,SP}^{S,SP}$
1	100	20	115	80	120	40	100
2	50	100	50	100	25	35	50
3	75	15	35	50	15	10	75

[1] The number in the parenthesis indicates the length in words in each domain.

[2] Note that $l_r = l'_r$ (r=S,SP) because in this case, we do not consider the cost of storage on the DDB ($f_{i,r}=0$, $r=S,SP$) and the redundant information usually does not have to be sent over the network in order to process a query.

and for all i∈{1,2,3},

$$\alpha_{i,S}^{S} = \alpha_{i,SP}^{SP} = \nu_{i,S}^{S,SP} = \nu_{i,SP}^{S,SP} = 0.1$$

$$\varepsilon_{i,S}^{S,SP} = \varepsilon_{i,SP}^{S,SP} = 0.05$$

$$\alpha_{i,S}^{S,SP} = \alpha_{i,SP}^{S,SP} = 0.3$$

$$\beta_{i,S}^{S} = \beta_{i,SP}^{SP} = \beta_{i,S}^{S,SP} = \beta_{i,SP}^{S,SP} = 0.25$$

$$\sigma_{i,S}^{S,SP} = \sigma_{i,SP}^{S,SP} = 0.6$$

$$\xi_{i,S}^{S} = \xi_{i,SP}^{SP} = 0.05$$

These parameters have all been chosen based on some estimated distribution of the data stored in the relations and the characteristics of the queries made on these two relations. They have been set independent of the nodes and the relations for easy understanding. The fixed cost of storage on the system have all been neglected because the storage cost is usually very small as compared to the communication cost. It is intended to show by this example, the amount of redundant information needed in order for this technique to be cost effective.

In Figure 5 and 6, two graphs are plotted to show the ratio of cost with redundancy and cost without redundancy against $\delta_{i,r}^{S,SP}$ [3]. In Figure 5, the graph is plotted for various values of $\gamma_{i,S,SP}^{S,SP}$ [3], with $\eta_{i,S,SP}^{r}$ [3] fixed at 0.5. Similarly, in Figure 6, the graph is plotted for various values of $\eta_{i,S,SP}^{r}$ [3], with $\gamma_{i,S,SP}^{S,SP}$ [3] fixed at 0.5. It is seen from these two graphs that whenever sufficient redundant information is added to the DDB so that over half of the non-decomposable queries or updates become decomposable, the resultant operational costs are less than the costs without the use of redundant information. Further, it is seen from Figure 6 that when the fraction of updates that will update the redundant information is less than 0.5, there is, in general, a cost improvement.

The results we have shown in the example are for illustration. More detailed evaluations are necessary before any definite conclusions can be drawn.

VI *CONCLUSION*

In this paper, we have studied the problem of optimal relation placements on a distributed relational data base. The objective of the problem is to minimize the total operational costs of the system. It is proven that the problem of placements of multiple relations on a DDB can be decomposed into multiple sub-problems of optimal placements of individual relations. This results in a significant reduction in the complexity of the optimization problem. The type of queries that can be made on a distributed relational data base are also classified. It is seen that non-decomposable queries cause a lot of communication overhead on the system. A technique is proposed in this paper which pre-analyzes the type of queries being made on the data base and the characteristics of the data, and introduces additional redundant information onto the data base so that non-decomposable queries can be made decomposable. The result is a decrease in the total retrieval cost and an increase in the total update cost. It is seen that this problem can similarly be decomposed into multiple sub-problems of optimizing the placements of individual relations. A simple example is used to illustrate some of the properties of the technique. It must be noted that a lot of generality is introduced in the development of the technique and a lot of parameters are defined. However, in most practical cases, most of these parameters will be identical, and therefore as illustrated in the example, the number of parameters to be estimated on the system is relatively small.

[3] It is assumed that r=S, SP; the variables $\delta_{i,r}^{S,SP}$, $\eta_{i,S,SP}^{r}$ are independent of i and r and the variables $\gamma_{i,S,SP}^{S,SP}$ are independent of r.

REFERENCES

[AND75] Anderson, G. A. and Jensen, E. D., "Computer Interconnection Structures: Taxonomy, Characteristics and Examples", Computing Surveys, Vol. 7, No. 4, December, 1975.

[ALC76] Alcouffe, A., and Muratet, G., "Optimum Location of Plants", Management Science, Vol. 23, No. 3, Nov. 1976, pp. 287-274.

[CAS72] Casey, R. G., "Allocation of Copies of a File in an Information Network", AFIPS, SJCC, 1972, pp. 617-625.

[COD70] Codd, E. F., "A Relational Model of Data for Large Shared Data Bases", CACM, Vol. 13, No. 6, June 1970.

[DAT77] Date, C. J., "An Introduction to Data Base Systems", 2nd Edition, Addison-Wesley, 1977.

[EPS78] Epstein, et. al., "Distributed Query Processing in a Relational Data Base System", Report No. UCB/ERL M78/18, Electronics Research Laboratory, University of California, Berkeley, 1978.

[GHO76] Ghosh, S. P., "Distributing A Data Base with Logical Associations on a Computer Network for Parallel Searching", IEEETSE, Vol. SE-2, No. 2, June 1976, pp. 106-113.

[HEV79] Hevner A. R., and Yao, S. B., "Query Processing in Distributed Data Bases", IEEE Trans. on Software Engineering, Vol. SE-5, No. 3, May 1979, pp. 177-187.

[LEV74] Levin, K. D., "Organizing Distributed Data Bases in Computer Networks", Ph.D. Dissertation, University of Pennsylvania, 1974.

[MOR77] Morgan, H. L., and Levin, K. D., "Optimal Program and Data Locations in Computer Networks", CACM, Vol. 20, No. 5, May 1977, pp. 315-322.

[RAM76] Ramamoorthy, C. V., and Krishnarao, T., "The Design Issues in Distributed Computer Systems", Infotech State of the Art Report on Distributed Systems, 1976, pp. 375-400.

[RAM79a] Ramamoorthy, C. V., and Wah, B. W., "Data Management in Distributed Data Bases", Proc. NCC, AFIPS Press, 1979, pp. 1011-1024.

[RAM79b] Ramamoorthy, C. V., and Wah, B. W., "The Isomorphism between File Allocation and Warehouse Location", submitted for publication.

[WAH79] Wah, B. W., "Data Management in Distributed Systems and Distributed Data Bases", Ph.D. Dissertation, University of California, Berkeley, 1979.

[WON76] Wong, E., and Youssefi, K., "Decomposition - A Strategy for Query Processing", ACM Trans. on Data Base, Vol. 1, No. 3, Sept. 1976, pp. 223-241.

[WON77] Wong, E., "Restructuring Dispersed Data from SDD-1: A System for Distributed Data Bases", Comp. Corp. of America, Tech. Rep. CCA-77-03, 1977.

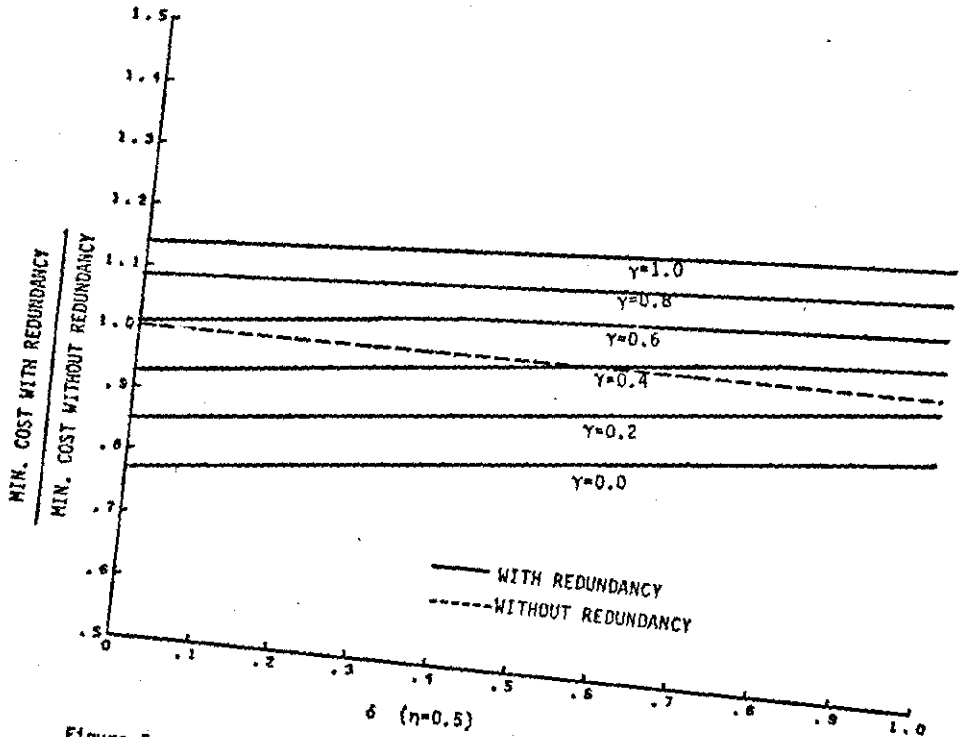


Figure 5 A PLOT OF COST RATIO w.r.t. γ FOR VARIOUS VALUES OF δ
(It is assumed that γ,δ,η are independent of r=S,SP and 1)

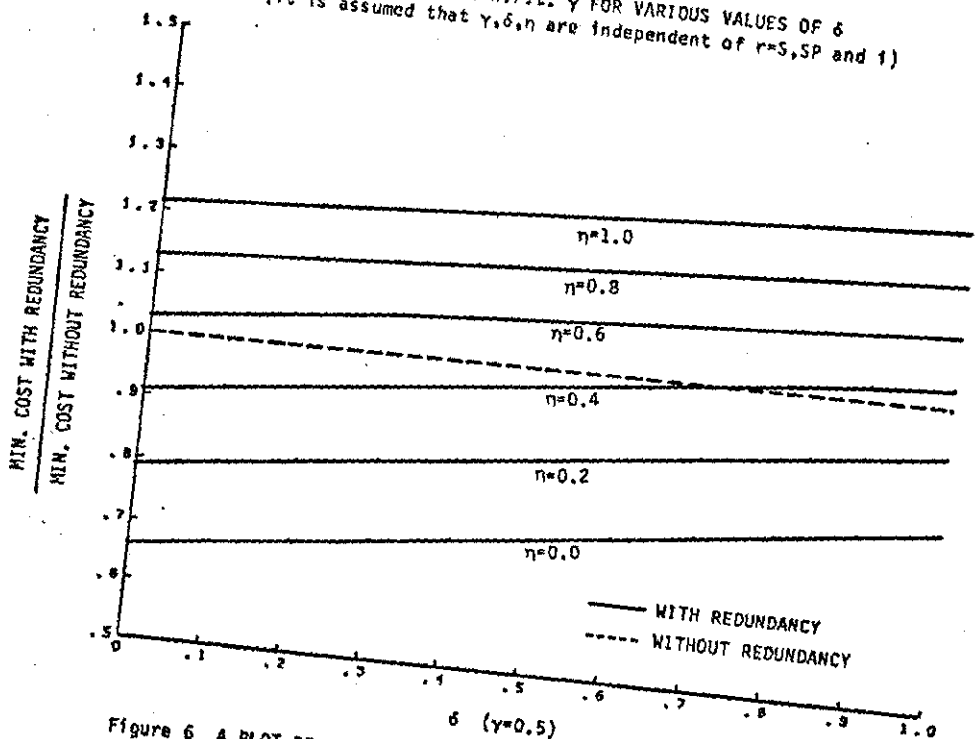


Figure 6 A PLOT OF COST RATIO w.r.t. γ FOR VARIOUS VALUES OF δ
(It is assumed that γ,δ,η are independent of r=S,SP and 1)

ACKNOWLEDGEMENT
We would like to thank Mr. C. R. Vick and Mr. J. E. Scalf for many helpful discussions related to this work.