1990

# ALLOCATION OF DATABASES IN A DISTRIBUTED DATABASE SYSTEM

Sudha Ram
*University of Arizona*

Sridhar Narasimhan
*Georgia Institute of Technology*

# ALLOCATION OF DATABASES IN A DISTRIBUTED DATABASE SYSTEM

**Sudha Ram**[1]
Department of Management Information Systems
University of Arizona

**Sridhar Narasimhan**
College of Management
Georgia Institute of Technology

## ABSTRACT

Our research focuses on developing a methodology for designing distributed database systems. This methodology is used to allocate databases across a number of computer sites connected by a communication network. It takes into account the pattern of usage of the databases, the communication costs in the network, delays due to queuing of requests for data, costs for maintaining consistency among the various copies of a database, and storage costs for the databases. The methodology is based on nonlinear integer programming modeling. A Lagrangian relaxation procedure using decomposition is developed to derive near optimal solutions for the problem. A tool has been built to operationalize this methodology, the model, and the solution procedure. The methodology developed in this research makes a significant contribution to the database field because it is one of the first to consider communication costs, costs of maintaining consistency, *and* queuing delays for the database allocation problem. It is applicable to a wide range of organizations which are in the process of moving from a centralized to a distributed computing environment.

## 1. INTRODUCTION

Databases are at the heart of most modern computerized systems. Typically an organization uses more than just a single database for its operation. In order to facilitate the use of multiple databases, most organizations are moving towards distributed database systems. A Distributed Database System (DDS) is a collection of data that are spread across different computers connected using a communications network. Each site of the network has autonomous processing capacity and can perform local applications. Each site also participates in the execution of one or more global applications (Ceri and Pelagatti 1984). These applications require accessing data from other remote sites using the communications network. The DDS is managed by a single Distributed Database Management System (DDBMS) that may offer complete transparency to the user. The most important feature of a DDS is the concept of cooperation among autonomous sites. DDS are used in applications requiring access to an integrated database from geographically dispersed locations. Such applications include banking, military command and control, and inventory control. In addition to the advantage of distributing a single database, advanced communication technology allows the integration of existing distinct, and perhaps radically different, databases for the purpose of sharing valuable information. Such heterogeneous systems are also called DDS. Heterogeneous distributed databases are found in applications such as computer integrated manufacturing and geographic information systems (Ram and Chastain 1989). The design of these distributed databases however is a very complex task. It requires understanding and solving a number of interrelated problems.

In this paper we focus on support for an important design problem: distribution of databases across a number of sites connected by means of a communication network. The outline of the paper follows. In Section 2, we briefly describe the various design issues to be considered in developing DDS. In the same section, we review past research in the areas of data allocation and concurrency control mechanisms. Section 3 proposes a model for database allocation and describes a procedure for solving this model using the Lagrangian Relaxation technique. Section 4 describes the analysis performed with the model for database allocation. Finally, Section 5 summarizes the contributions of this research and provides guidelines for future work.

## 2. ISSUES IN DISTRIBUTED DATABASE DESIGN

A database system serves as a foundation for the entire information system of an enterprise. It is very important, therefore, to design a database in such a way that it achieves the functions and performance needed to satisfy the information processing environment in the enterprise. The design of distributed database systems involves several

complicated issues. A detailed review of these design issues has been provided by several authors (Ceri, Pernici, and Wiederhold 1984; Goodman and Rothnie 1977). These issues are briefly summarized below.

1. *Fragmentation*: Typically, a large database is divided into several logical units. Each of these units is called a fragment. The fragments of the database need to be designed in a such a way that by combining them the original database can be reproduced without any loss of information. Several well known algorithms are available for database fragmentation (Elmasri and Navathe 1989).

2. *Data and Program Distribution*: Once the fragments are determined, they are allocated across the various sites in the DDS. Programs that operate on these fragments are also allocated. Some fragments may be replicated at two or more locations, while others may not be replicated at all.

3. *Global Conceptual Modeling*: A suitable data model must be used to depict the overall database such that it offers distribution transparency to the user. This means that the user need not be aware of the exact location of the data. A global directory keeps track of the locations of the data.

4. *Concurrency Control*: Since there may be more than one copy of a data item in the DDS, the Distributed Database Management System must maintain consistency among these copies (Bernstein and Goodman 1982). Several mechanisms are available to accomplish this function and they are described in more detail later in this section.

5. *Security, Integrity, and Recovery*: Access to the databases must be carefully controlled to restrict unauthorized entry. In order to provide a robust system to the user, the DDS must have procedures for recovering from network errors and other types of problems. It is possible that a computer site may go down due to errors in the network. Once it comes back up, it has to make its database(s) consistent with other copies of the same database(s) in the network. Sophisticated recovery techniques are required to accomplish this task.

6. *Design of communication network*: Network design is a complicated issue by itself. It involves deciding on the topology (i.e., which nodes are interconnected), choosing link technologies (fiber optics, coaxial cable, microwave, satellite, etc.), and routing strategies for transferring messages between network nodes. Typical criteria used in designing communication networks are minimizing costs of constructing and operating the network, minimizing delays experienced by users, and ensuring that reliability and availability requirements are met.

7. *Query Processing*: Since multiple copies of data items exist, it may be possible to process each query in different ways. Optimization of queries becomes more complicated and techniques to accomplish these have to be devised. Several techniques have been proposed in the past (Ceri and Pelagatti 1984).

In this paper we will be focusing on two of the design issues: data distribution and concurrency control. Each of the design issues mentioned above has been fairly well researched. However, most of the research has focused on only one aspect assuming that everything else was held constant. Initially this assumption was acceptable and probably even necessary in developing a basic understanding of each of these facets. A more comprehensive stage of research must now focus on the combined effects and interaction among these issues (Ram 1987).

### 2.1 Data Allocation

This paper uses the term *database* to mean a collection of data. Each collection of data can also be called a fragment. The issue that is being examined in this paper is the allocation of these databases. Several strategies have been suggested for data allocation (Hevner and Rao 1988; Apers 1988). In making a decision on database allocation, there is a tradeoff between response time and storage costs. If a copy of each database is stored at each node, answers to queries will be obtained very fast. However, one has to pay the price in terms of storage costs. With the recent decline in secondary storage costs, this does not seem to be a major consideration. What is important however is the cost of updating all copies to maintain consistency. If there are too few copies of a database, there will be a major delay in getting responses though it will be easier to maintain consistency among the different copies. In most cases it is desirable to allocate the databases locally so that most queries can be satisfied locally. However, allocating too many databases to a single node may cause a severe drop in performance of the system due to traffic bottlenecks. On the other hand, allocating too few databases locally could cause underutilization of resources. Thus the database allocation problem is complex and non trivial. Further, the size of the problem could easily become very large given a sizable number of nodes and databases.

Starting with Chu's (1969) pioneering effort, a number of mathematical models have been proposed to allocate databases in an optimal manner. The optimum can be in terms of either performance or operating costs. It has been proven that this problem is NP complete (Eswaran 1974; Garey and Johnson 1979) thus requiring special heuristic solutions. Some special heuristics have been suggested by Grapa and Belford (1977). A detailed review of several database allocation models can be found in the surveys by Dowdy and Foster (1982), Wah (1984), and others (Chandy 1977; Elam and Fisher 1979). In the last

five years, the problem of computer location and database allocation has been dealt with by several authors (Gavish 1985; Gavish and Pirkul 1986). Pirkul (1987) has also described the problem of configuring distributed computer systems in the presence of on-line backups. More recently models have been developed that incorporate different concurrency control mechanisms (CCM) (Ram 1989; Ram and Marsten 1989). The models use the central node locking mechanism or the Write locks all - Read locks one mechanism. However, these models do not directly account for delays in the system due to the CCM.

Based on a study of past research on database allocation it is evident that a need exists to integrate other design issues such as concurrency control and query processing into the database allocation problem. Very few models in the past have examined the consequences of incorporating a concurrency control mechanism. Further, the models need to take into account the queuing delays arising from the use of the CCM. Queuing delays impose costs on network users. These delays, which occur at the database sites, are dependent on the query and update transactions originating from users of the database and also on the CCM being used. It is this aspect that is being examined in our research. In the next section we briefly describe some commonly used distributed CCMs.

### 2.2 Concurrency Control Mechanisms (CCM)

As explained earlier, one of the main objectives of a DDBMS is to enable multiple users to access the data concurrently. This can be achieved without problems as long as the user is only retrieving data. However, if two or more users update the same data item at the same time, then all but the last of the updates may be lost, thus impairing the accuracy of the database. Similarly, when one user is updating a particular piece of data, no other user should be allowed to read the same piece of data in order to avoid an incorrect value from being read. In a centralized database, consistency has to be maintained among various parts of one database. In a distributed database, the problem is magnified because each copy of a database must reflect the same accurate information at the same point of time.

There are several methods available for concurrency control in distributed database systems. A careful study of these methods has revealed that they are variations of three basic techniques: Two-phase locking (2PL) (Stonebraker 1979), Timestamp Ordering (T/O) (Bernstein and Goodman 1980), and Optimistic methods (also called commit-time validation or certification) (Ceri and Owicki 1982; Kung and Robinson 1981). An exhaustive survey of all these methods is available in Kohler (1980). Bernstein and Goodman (1981) also survey a number of CCM and describe how new algorithms could be created by combining the three basic mechanisms. Given the large number of concurrency control algorithms, several studies have been devoted to evaluating the performance of these algorithms (Agrawal and DeWitt 1985; Peinel and Reuter 1983). From the results of these studies, it can be concluded that, in general, a concurrency control algorithm that tends to conserve a resource by blocking transactions that might otherwise have to be re-started is better than a restart-oriented algorithm in an environment where physical resources are limited. It was found that 2PL outperformed the immediate restart and optimistic algorithms for medium to high levels of resource utilizations (Agrawal, Carey, and Livney 1987). Hence, in this research we have chosen to incorporate concurrency control algorithms based on 2PL.

There are several distributed concurrency control algorithms based on the principle of locking. The Central Node Locking Mechanism is one such technique. In this technique one of the sites in the network is designated as the coordinating site. All requests for locks go through this site which is responsible for granting, delaying, or denying lock requests. It is a very easy mechanism to implement. However, there is potential for traffic bottlenecks to and from the central site. In addition, if the central site goes down, the entire system is affected unless a backup site can take over. In spite of this disadvantage, this is one of the most commonly used CCMs in distributed database systems (Ullman 1988). Other techniques such as "Write Locks All - Read Locks One," "Majority Consensus," and "Primary Copy" locking mechanisms are also widely used. For details of these techniques, the reader is referred to Ullman. In all of these techniques, the overhead due to communication is much higher than the central node locking mechanism.

## 3. A MODEL FOR DATABASE ALLOCATION

In this section we present the formulation of a mathematical model for database allocation. This model along with others will form a part of our methodology for DDS design. First the DDS environment is briefly described and the objectives and constraints in this environment are summarized. The model is described in detail and a solution methodology is presented.

The major decisions that will be made with the help of the methodology are:

1. Allocation of databases across various sites of the network.
2. Choice of the central node.
3. Identification of the site which will supply data to a requesting site that does not have the items.

In making these decisions, the following objectives and constraints are taken into account:

1. Minimize communication costs for different types of operations, i.e., read-only queries and updates.

2. Minimize storage costs.

3. Minimize delays due to queuing of data requests at each site.

4. Minimize the cost of maintaining consistency among the various copies of a database.

5. Maintain a minimum level of reliability in the system.

6. Satisfy the transactions processing capacity at the database sites.

The model described in this paper attempts to incorporate all of these features.

### 3.1 Assumptions about the Distributed Database Environment

Before describing the model, it is necessary to understand a few details about the DDS environment. There is assumed to be a set of nodes connected to each other by means of a communication network. Computer hardware (mainframe, minicomputer, intelligent terminal, etc.) is located at each node. The hardware need not be the same at each node. The nodes of the network can communicate at a certain cost per unit of data transmitted. Users of the system have access to databases that can be stored at any of the nodes. Each database is considered to be a collection of data items. Transactions in the databases are of two types: read-only queries (or retrievals) and updates. Each query may consist of a series of instructions to extract the data item values and present them in a suitable format to the person sending the request. Similarly, each update could consist of a sequence of steps designed to extract the data item values and write them back into the appropriate databases after updating them. Thus each query or update can be considered to be a program. For instance, in a banking application system, one program might answer a question such as "What is the balance in account number 354215?" Similarly, another program might transfer a sum of money from a money market account into a checking account.

Thus there are separate programs to handle each transaction type and these programs are also stored at different nodes. Read-only queries read information from a single copy of a database via a particular copy of a program if necessary. Updates write information into all copies of a database via any one copy of an update program. The model described in this paper allocates only databases across a network. It does not consider the allocation of programs. It is assumed that each program is allocated at all the nodes where it is required. Since programs are typically very small in size compared to databases, this is a reasonable assumption. We also assume that the database has already been divided into well-defined fragments. Distribution design and partitioning algorithms have been developed by other researchers (Ceri, Navathe, and Wiederhold 1983; Sacca and Wiederhold 1985; Gladney 1989).

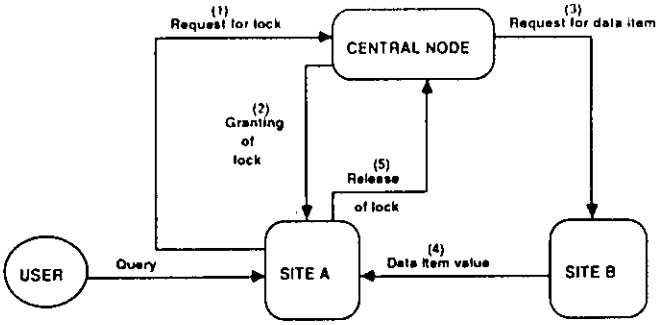
### 3.2 The Model

In our model we have incorporated the Central Node Locking mechanism for concurrency control. We differentiate between two kinds of locks: read-only locks (RLOCKS) and Write locks (WLOCKS). All lock requests and data requests are handled by a central node that controls access to each database. If a transaction wants to read and not update a data item, it requests an RLOCK on the data item. If a transaction wants to read and update a data item it requests a WLOCK on the data item. The rules for denying and granting locks are very simple; the central node grants an RLOCK as long as no other transaction has a WLOCK on the data item. A WLOCK is granted *only* if no other transaction has an RLOCK or WLOCK on the data item.

For read-only queries the following sequence of steps is executed (see Figure 1). The query is initiated at Site A which may or may not contain a database. A program at site A again decomposes the query into subqueries if required. Site A sends a message to the central site requesting a read lock on one or more data items. The central site checks its lock tables and grants the lock(s) if it can. The central node then identifies the site(s) that will supply the required data item(s) to Site A and sends a message instructing the site (Site B) to send the requested data item(s). Site B then sends the data item directly to Site A. Site A processes the query and provides the user with the answer to the query. After the query is executed, site A sends a message to the central node requesting that the lock(s) be released. In the case of updates, some additional communication will take place in the network (see Figure 2). As in the case of queries, locks are requested and granted. Data items are sent to the requesting site; Site A then updates the items and sends the updated value(s) to the central site which, in turn, broadcasts them to all sites that have a copy of the data item. A lock release message is also sent to the central site by Site A. Once all the updates are performed the central site releases the lock(s). We assume that acknowledgment messages are piggybacked on data or lock request messages.

All these communication flows are captured in the formulation of the mathematical model.

We define the following decision variables to formulate the problem:




Note: Sequence of messages is indicated by the number in parentheses.

**Figure 1. Communications Flow for Read-Only Queries**

$$x_{kd} = \begin{cases} 1 \text{ if a copy of database } d \text{ is stored at node } k \\ 0 \text{ otherwise} \end{cases}$$

$$z_m = \begin{cases} 1 \text{ if } m \text{ is the central node} \\ 0 \text{ otherwise} \end{cases}$$

$$y_{imkd} = \begin{cases} 1 & \text{if node } k \text{ is instructed by the central node } m \text{ to supply the processing node } i \text{ with data item values out of database } d \\ 0 & \text{otherwise} \end{cases}$$

In order to model the queuing delays in our model, we need to derive an expression for $T_k$, the message traffic which arrives at a node $k$. In processing queries, a database site receives one message per query, the message being the request for the data item. The central node receives two messages per query, the first is the request for the lock and the second is the release of the lock. The query originating site receives two messages per query, one is the granting of the lock and the other is the data item. Therefore, the number of query related messages that arrive at a site $k$ is equal to (refer to Appendix 1 for an explanation of the notation):

$$\sum_{d \in D} \sum_{i \in N} \sum_{m \in N} n_i * f_{id} * y_{imkd} + (2 * z_k * \sum_{i \in N} n_i) + (2 * n_k)$$

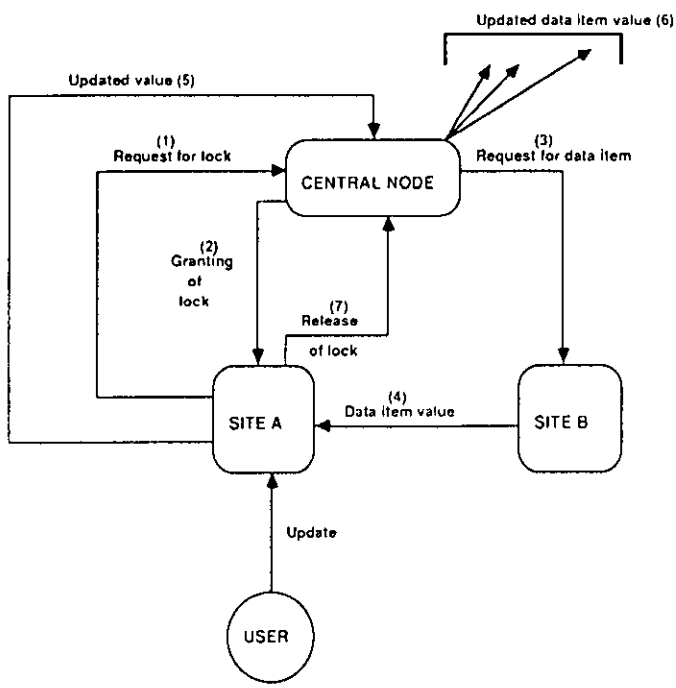
In processing updates, a database site furnishing the data item gets a request for the data item. The central node receives three messages per update. These messages are a request for a lock, the updated value, and the release of the lock. All database sites receive the updated value. The update originating site receives two messages per update, one is the granting of the lock and the other is the data item. The number of update related messages that arrive at node $k$ is equal to

$$\sum_{d \in D} \sum_{i \in N} \sum_{m \in N} m_i * g_{id} * y_{imkd} + (3 * z_k * \sum_{i \in N} m_i) + \sum_{d \in D} \sum_{m \in N} m_i * g_{id} * x_{kd} + (2 * m_k)$$

Therefore,

$$T_k = \sum_{d \in D} \sum_{i \in D} \sum_{m \in N} n_i * d_{id} * y_{imkd} + (2 * z_k * \sum_{i \in N} n_i) + (2 * n_k) + \sum_{d \in D} \sum_{i \in D} \sum_{m \in N} m_i * g_{id} * y_{imkd} + (3 * z_k * \sum_{i \in N} m_i) + \sum_{d \in D} \sum_{i \in N} m_i * g_{id} * x_{kd} + (2 * m_k)$$

In order to model the queuing delay in the distributed database system, we make the following assumptions:




Note: Sequence of messages is indicated by the number in parentheses.

**Figure 2. Communication Flows for Updates**

1. Queuing can occur at nodes that may or may not contain databases.
2. The number of transactions that can be processed per unit time is constrained by the capacity of each node.
3. The capacity of a node $k$ ($CAP_k$) can be expressed as the number of messages it can handle per unit time.
4. *Peak* and *non-peak* period phenomena are ignored.

5. The queuing (waiting) time cost per unit time is $C^w$(\$/unit time/message).

6. The arrival process of queries and updates follows a Poisson process.

7. Message lengths follow an exponential distribution.

8. There are no queuing and propagation delays in the network. Delays only occur at the database sites.

9. Each node has unlimited buffer to store messages.

10. The queuing delay of messages (transactions) is modeled as a network of M/M/1 queues (Kleinrock 1975, 1976) in which the database sites are treated as servers whose service rate is proportional to their message processing capacity.

Based on these assumptions, queuing time cost is

$$\sum_{k \in N} C^w \frac{T_k}{(CAP_k - T_k)}$$

This can be rewritten as

$$\sum_{k \in N} C^w \frac{f_k}{(1 - f_k)}$$

where

$$f_k = T_k / CAP_k$$

and

$$0 \leq f_k < 1.$$

The problem is formulated as shown below. The terms Q1 through Q5 and U1 through U7 refer to coefficients associated with the communication costs as explained in Appendix 2.

Problem-P

$$Z_p = Min. \sum_{i \in N} \sum_{m \in N} (Q1 + Q2 + Q5 + U1 + U2 + U5 + U7) z_m$$

$$+ \sum_{i \in N} \sum_{d \in D} \sum_{k \in N} \sum_{m \in N} (Q3 + Q4 + U3 + U4) y_{imkd}$$

$$+ \sum_{i \in N} \sum_{d \in D} \sum_{k \in N} \sum_{m \in K} U6 x_{kd} + \sum_{k \in N} C^w \frac{f_k}{(1 - f_k)}$$

$$+ \sum_{k \in N} \sum_{d \in D} S_{kd} * x_{kd} \qquad (1)$$

subject to:

$$\sum_{m \in N} z_m = 1 \qquad (2)$$

$$\sum_{k \in N} x_{kd} \geq 1 \qquad \forall d \in D \qquad (3)$$

$$y_{imkd} \leq x_{kd} \qquad \forall m \in N, \forall i \in N, \forall k \in N, \forall d \in D \qquad (4)$$

$$y_{imkd} \leq z_m \qquad \forall m \in N, \forall i \in N, \forall k \in N, \forall d \in D \qquad (5)$$

$$\sum_{m \in N} \sum_{k \in N} y_{imkd} = 1 \qquad \forall i \in N, \forall d \in D \qquad (6)$$

$$1/CAP_k \sum_{d \in D} \sum_{i \in N} \sum_{m \in N} n_i ( f_{id} * y_{imkd} + (2 * z_k * \sum_{i \in N} n_i)$$

$$+ (2 * n_k) + \sum_{d \in D} \sum_{i \in N} \sum_{m \in N} m_i * g_{id} * y_{imkd} + (3 * z_k * \sum_{i \in N} m_i)$$

$$+ \sum_{d \in D} \sum_{i \in N} m_i * g_{id} * x_{kd} + (2 * m_k)$$

$$= f_k \ \forall k \in N \qquad (7)$$

$$0 \leq fk < 1 \qquad \forall k \in N4 \qquad (8)$$

$$y_{imkd}, x_{kd}, z_m \in \{0,1\} \qquad \forall m \in N, \forall i \in N, \forall d \in D \qquad (9)$$

The first three terms of the objective function (1) capture the communication costs in the network. The fourth (nonlinear) term captures the queuing costs incurred by transactions (messages) waiting to be processed at the database sites. The fifth term in the objective function captures the cost of storage of the databases. Constraint set (2) states that there must be exactly one node selected to serve as the central site. Constraint set (3) states that there must be at least one copy of each database allocated in the network. For some or all of the databases, if needed, we can increase the right hand side of these constraints to ensure that more than one copy exists. This is one method for improving the reliability and availability of the distributed database system. Constraint set (4) states that a node $k$ can supply a node $i$ with data items from database $d$ only if a copy of database $d$ exists at node $k$. Constraint set (5) captures the relationship between the $y$ and $z$ variables, specifically it states that if node $m$ is not the central site, then it cannot instruct site $k$ to supply site $i$ with data items from database $d$. Constraint set (6) states that all the transactions which originate at node $i$ and request data items from database $d$, are served from one node $k$ which contains that database $d$. Constraint sets (7) and (8) capture the processing capacity constraints at the network nodes. These capture the fact that in order to prevent bottlenecks and the resulting inordinate queuing delays at each node, there is a limit on the total number of queries and updates that each node can process per unit time.

### 3.3 Solution Methodology

The model presented in the previous section constitutes a fairly large size zero-one mixed-integer programming problem. An instance of the model with ten nodes and three databases will have over 3,000 integer variables and 6,000 constraints. Furthermore, the problem is NP-complete (see Ram and Narasimhan [1990] for proof of NP-completeness). For problems in this class, the solution time for exact solution algorithms grows exponentially with problem size. Hence it is extremely unlikely that exact solution procedures can be developed to provide optimal solutions in reasonable amounts of computing time. Hence we have chosen to use the Lagrangian Relaxation (LR) technique to solve this problem.

In the past, LR techniques have been extremely successful in providing *good* quality solutions which are very close to the (unknown) optimal solutions. Further, these solutions are produced in reasonable amounts of computing time. The LR technique involves relaxing some of the constraints in the model by using dual multipliers and getting quick bounds on the optimal solution by solving "simpler" subproblems. Using the solution of the relaxation, a feasible solution to the problem is then constructed. The bounds furnish an estimate of the unknown optimal solution values and thereby provide the designer with a means of estimating the quality of the solutions. This is a significant advantage to this method since estimates of solution quality are unavailable using other heuristic techniques. The use of LR methods was first suggested by Everett (1963). Held's and Karp's (1970) successful application of this relaxation to the traveling salesman problem led to its use in a variety of other problems. For a survey of LR, the reader is referred to Fisher (1981; 1985). The technique has also been successfully used in various location problems (Geoffrion and McBride 1978), distributed computer system design (Gavish and Pirkul 1986), and communication network design (Narasimhan 1987; 1990).

Using the LR technique Problem-P is first decomposed into four subproblems (L1, L2, L3 and L4) shown in Appendix 3. The subgradient method is used to generate the Lagrangian multipliers. The solution procedure for each subproblem and the overall Problem-P is described below.

- **Solution procedure for Problem-L1**

  Set $z_m$ to 1 for that $m$ with the minimum cost coefficient in the objective function of Problem-L1. Set all other $z_m$ variables to 0.

- **Solution procedure for the subproblems of Problem-L2**

  For every subproblem $d$, set $x_{kd}$ to 1 for those variables that have nonpositive coefficients in Problem-L2 ($d$). However, if all $x_{kd}$ variables have their coefficients in Problem-L2 ($d$) positive, then set $x_{kd}$ to 1 for site k with the minimum value of $U6 + S_{kd}$.

- **Solution procedure for the subproblems of Problem-L3**

  For every $(i,d)$ subproblem, set $y_{imkd}$ to 1 for that $m,k$ combination with the minimum cost coefficient in the objective function of Problem-L3(i,d). Set $y_{imkd}$ variables to 0 for all other $m,k$ combinations.

- **Solution procedure for the subproblems of Problem-L4**

  For every subproblem $k$, set the value of the $f_k$ variable as follows:

$$f_k = \begin{cases} 1 - (C^w/\lambda_k)^{1/2} \text{ if } C^w/\lambda < 1 \\ 0 \text{ otherwise} \end{cases}$$

**A Feasible Solution Procedure to Problem-P (the primal problem):**

**Step 1**: Use the solution of Problem-L1 to set the values of the $z_m$ variables. This will identify the central node.

**Step 2**: Set the value of the $x_{kd}$ variables using the solutions of Problems-L2 ($d$). This will identify for each database the set of nodes where its copies are to be located.

**Step 3**: Set the values of the $y_{imkd}$ variables using the solutions of Problems-L3 ($i,d$).

**Step 4**: Enforce constraint sets (4) and (5), thereby resetting some $y_{imkd}$ variables to 0. This may violate constraint set (6). Thus, for every node $i$, database $d$ pair for which constraint set (6) is violated, set $y_{imkd}$ to 1 for the minimum feasible cost combination.

**Step 5**: Check if the values of these $x$,$y$, and $z$ variables satisfy the capacity constraint sets (7) and (8). If so, then set the values of the $f_k$ variables using constraint set (7) and evaluate the value of the primal objective function.

This procedure is executed at every step of the subgradient optimization and the best primal feasible solution value is retained.

## 4. COMPUTATIONAL RESULTS

The solution procedure described in Section 3 has been implemented using FORTRAN and several experiments

were designed to test its performance. The upper limit on the number of subgradient iterations was established to be 200. An instance of the sample data used for these experiments is shown in Table 1. Table 2 summarizes the sizes of the problems that were tested along with the computation time required for each problem. All tests were performed on a VAX 11/780 computer running under version 5.3 of the VMS operating system. Six configurations of nodes and databases were used for the experiments. The configurations ranged from three nodes and three databases to ten nodes and ten databases. For each configuration a total of 50 problems were solved. The input parameter values for each of the 50 problems in each configuration were varied in a systematic manner. The test cases included the following patterns of transactions:[2]

- Only queries and no updates (e.g., 100,000 queries and 0 updates at each node).
- Only updates and no queries (e.g., 100,000 updates and 0 queries at each node).
- Increase the number of updates until they became equal to the number of queries.
- Decrease the number of queries while keeping the number of updates constant.
- Change the processing capacities of the nodes (e.g., a range of 200,000 to 2,000,000 messages at each node).

**Table 1. Sample Values for Non-Decision Parameters**

| PARAMETER | VALUE |
|---|---|
| $S_{kd}$ | \$1.00 per unit time |
| $C_{ik}$ | \$0.01 per packet |
| $U_{ik}$ | \$0.01 per packet |
| $\delta_i$ | 50 packets |
| $\delta_i'$ | 50 packets |
| $f_{id}$ | Evenly divided among $d$ databases |
| $g_{id}$ | Evenly divided among $d$ databases |
| $n_i$ | 100,000 per unit time |
| $m_i$ | 100,000 per unit time |
| $\alpha$, $\alpha'$ | 1.0 |
| $\beta$, $\beta'$ | 1.2 |
| $\gamma$ | 1.3 |
| $C^w$ | \$10 per message per unit time |
| $CAP_k$ | Ranged from 200,000 to 2,000,000 per unit time |

For each configuration, the most obvious patterns that emerged were:

- A fully replicated allocation of databases whenever there were no updates in the system.
- A single copy of each database allocated whenever there were only updates in the system.
- For configurations with queries and updates, the allocation was partially replicated, depending on the values specified for processing capacities and other parameters.

The solution procedure was very effective and efficient for a wide range of problems. As shown in Table 2, even for problems of considerably large size (such as ten nodes and ten databases), a good solution can be obtained in less than four minutes (average). In most cases, the computation time depends on the value that is used as a limit for the gap between the feasible solution and the lower bound. In many cases the subgradient algorithm did not reach the upper limit of 200 iterations but stopped earlier when the gap between the feasible solution and the lower bound was less than 0.01 percent. A more extensive computational study is reported elsewhere (Ram and Narasimhan 1990). We have implemented a decision support tool based on this model. The tool can be used by distributed database designers to study the impact of changes in various parameter values on the allocation of databases.

## 5. CONTRIBUTIONS AND FUTURE RESEARCH

This paper has presented a methodology for allocation of databases in a distributed system. The methodology is based on a mixed integer programming model. Since the model is fairly complex, a solution based on Lagrangian relaxation techniques was proposed to solve it. This research makes several significant contributions to the field of distributed database systems.

- The model proposed here is one of the first to consider the cost of queuing delays. We have illustrated the use of the M/M/1 model for delays experienced by messages generated in the processing of queries and updates. While future models could incorporate more sophisticated models for queuing delays, we believe that the model used here is quite robust and will provide network designers with reasonable results. An interesting extension is the possibility of capturing different costs of delays for various users with suitable modifications in the model.
- The model has captured the communication flows for one concurrency control mechanism. The choice of a CCM affects the communication costs and also the mathematical structure of the model itself. We believe that with the experience gained in designing distributed databases in this research other CCM, such as the majority consensus and primary copy technique, can be explored in future research.

**Table 2. Performance of Solution Procedure**

| Number of | | | Solution time CPU Seconds | | | Percent Gap* | | |
|---|---|---|---|---|---|---|---|---|
| Nodes | Databases | Variables | Min. | Max. | Avg. | Min. | Max. | Avg. |
| 3 | 3 | 93 | 6.7 | 10.3 | 8.3 | .02 | .20 | .08 |
| 5 | 3 | 395 | 10.7 | 29.5 | 22.1 | .03 | 2.10 | 0.90 |
| 5 | 5 | 780 | 17.4 | 127.4 | 67.4 | .07 | 11.10 | 4.30 |
| 10 | 3 | 3040 | 50.5 | 345.6 | 201.1 | .13 | 10.50 | 4.20 |
| 10 | 5 | 5060 | 64.3 | 442.7 | 221.1 | .03 | 12.20 | 3.10 |
| 10 | 10 | 10110 | 78.2 | 357.7 | 234.5 | .008 | 11.40 | 3.20 |

*Gap between feasible solution value and the lower bound.

- Various query and update traffic scenarios can be studied by using the decision support tool being developed in this research. The tool can be used to analyze the impact of changes in communication costs, transaction patterns, and processing capacities.

- Availability and reliability requirements have been captured in the model to some extent. For example, if an organization deems it necessary that at least two copies of a particular database exist, this requirement can be incorporated into the models without much difficulty. The impact of such decisions on delays, storage and communication costs can be easily explored by the designer.

- The impact of new technologies and various communication alternatives can be studied via the proposed model. While the cost of installing the communications network has not been captured per se in our model, the designer can modify the per message communications cost between various nodes and can check if it is worthwhile to invest in the new communication system.

- A unique solution methodology is proposed to efficiently solve large and complicated models. As mentioned earlier, the model considered here is one of the first to consider various CCM and include queuing costs. These problems could conceivably involve thousands of variables and constraints. Off the shelf packages will not work in this problem domain due to the problems being nonlinear integer programming models. Our solution method has the benefit of providing us with a lower bound which is used to estimate the quality of the feasible solution. As more organizations continue the trend towards decentralization, the benefits of such models and their accompanying decision support tools will be invaluable.

In the model presented here, we do not attempt to capture the cost of dealing with deadlocks. In distributed systems, deadlock detection or prevention is very complicated and typically an optimistic strategy is followed. Our future research will attempt to model these costs explicitly as well as the locking granularity problem. Future research in this area will also attempt to examine the effect of fragmentation on database allocation. Later some simple query processing techniques will be incorporated into the model to examine a number of issues jointly. Another possible extension is the issue of developing multi-period models for distributed database design. Also to be considered in future research is the modeling of the integrated design of distributed databases and their underlying communications network.

## 6. REFERENCES


Agrawal, R.; Carey, M. J.; and Livney, M. "Concurrency Control Performance Modeling: Alternatives and Implications." *ACM Transactions on Database Systems*, Volume 12, Number 4, 1987, pp. 609-654.

Agrawal, R., and DeWitt, D. "Integrated Concurrency Control and Recovery Mechanisms: Design and Performance Evaluation." *ACM Transactions on Database Systems*, Volume 10, Number 4, 1985, pp. 529-564.

Apers, P. M. G. "Data Allocation in Distributed Database Systems." *ACM Transactions on Database Systems*, Volume 13, Number 3, 1988, pp. 263-304.

Bernstein, P., and Goodman, N. "Concurrency Control in Distributed Database Systems." *ACM Computing Surveys*, Volume 13, Number 2, 1981, pp. 185-222.

Bernstein, P., and Goodman, N. "Time Stamp Based Algorithms for Concurrency Control in Distributed Database Systems." *Proceedings of the Sixth International Conference on Very Large Databases*, Montreal, October 1980, pp. 285-300.

Bernstein, P., and Goodman, N. "A Sophisticates Introduction to Distributed Database Concurrency Control." *Proceedings of the Eighth International Conference on Very Large Databases*, Mexico City, October 1982, pp. 62-76.

Ceri, S.; Navathe, S.; and Wiederhold, G. "Distribution Design of Logical Database Schemas." *IEEE Transactions on Software Engineering*, Volume SE-9, Number 4, 1983, pp. 487-563.

Ceri, S., and Owicki, S. "On the Use of Optimistic Methods for Concurrency Control in Distributed Databases." *Proceedings of the Sixth Berkeley Workshop on Distributed Data Management and Computer Networks*, Berkeley, California, February 1982.

Ceri, S., and Pelagatti, G. *Distributed Databases: Principles and Systems*. New York: McGraw-Hill, 1984.

Ceri, S.; Pernici, B.; and Wiederhold, G. "An Overview of Research in the Design of Distributed Databases." *Database Engineering*, 1984, pp. 46-51.

Chandy, M. "Models of Distributed Systems." *Proceedings of the Third International Conference on Very Large Databases*, 1977, pp. 105-120.

Chu, W. W. "Multiple File Allocation in a Multiple Computer System." *IEEE Transactions on Computers*, Volume C-18, Number 10, 1969, pp. 885-889.

Dowdy, L. W., and Foster, D. V. "Comparative Models of the File Assignment Problem." *ACM Computing Surveys*, Volume 14, Number 2, 1982, pp. 287-313.

Elam, J., and Fisher, M. "The Use of Mathematical Models in Distributed Database Design." *Distributed Databases II, Infotech State-of-the-Art Report*, 1979, pp. 115-125.

Elmasri, R., and Navathe, S. B. *Fundamentals of Database Systems*. Menlo Park, California: Benjamin/ Cummings Publishing Company, 1989.

Eswaran, K. P. "Placement of Records in a File and File Allocation in a Computer Network." *Information Processing 74*, 1974, pp. 304-307.

Everett, H. "Generalized Lagrange Multipliers Method for Solving Problems of Optimum Allocation of Resources." *Operations Research*, Volume 11, 1963, pp. 399- 417.

Fisher, M. L. "An Applications Oriented Guide to Lagrangian Relaxation." *Interfaces*, Volume 15, 1985, pp. 10-21.

Fisher, M. L. "The Lagrangian Relaxation Method for Solving Integer Programming Problems." *Management Science*, Volume 27, Number 1, January 1981, pp. 1-18.

Garey, M. R., and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman and Company, 1979.

Gavish, B. "Models for Configuring Large-Scale Distributed Computing Systems." *AT&T Technical Journal*, Volume 64, Number 2, 1985, pp. 491-532.

Gavish, B., and Pirkul, H. "Computer and Database Location in Distributed Computer Systems." *IEEE Transactions on Computers*, Volume C-35, Number 7, 1986, pp. 583-590.

Geoffrion, A. M., and McBride, R. "Lagrangian Relaxation Applied to Capacitated Facility Location Problems." *AIIE Transactions*, Volume 10, 1978, pp. 40-47.

Gladney, H. M. "Data Replicas in Distributed Information Services." *ACM Transactions on Database Systems*, Volume 14, Number 1, 1989, pp. 75-97.

Goodman, N., and Rothnie, J. B. "A Survey of Research and Development in Distributed Database Management." *Proceedings of the Third International Conference on Very Large Databases*, 1977, pp. 10-27.

Grapa, E., and Belford, G. G. "Some Theorems to Aid in Solving the File Allocation Problem." *Communications of the ACM*, 1977, Volume 20, pp. 878-882.

Held, M., and Karp, R. M. "The Traveling Salesman Problem and Minimum Spanning Trees." *Operations Research*, Volume 18, 1970, pp. 1138-1162.

Hevner A. R., and Rao, A. "Distributed Data Allocation Strategies." In M. C. Yovits (ed.), *Advances in Computers*, Volume 27. Orlando, Florida: Academic Press, Inc., 1988, pp. 121-155.

Kleinrock, L. *Queueing Systems*, Volume 1. New York: Wiley-Interscience, 1975.

Kleinrock, L. *Queueing Systems*, Volume 2. New York: Wiley-Interscience, 1976.

Kohler, W. H. "Overview of Synchronization and Recovery Problems in Distributed Databases." *Proceedings COMPCON80*, Fall 1980, pp. 433-441.

Kung, H., and Robinson, J. "On Optimistic Methods for Concurrency Control." *ACM Transactions on Database Systems*, Volume 6, Number 2, 1981, pp. 213-226.

Narasimhan, S. *Topological Design of Networks for Data Communication Systems*. Unpublished Ph.D. Dissertation, The Ohio State University, Columbus, Ohio, 1987.

Narasimhan, S. "The Concentrator Location Problem with Varying Coverage." *Computer Networks and ISDN Systems*, 1990, forthcoming.

Peinel, P., and Reuter, A. "Empirical Comparison of Database Concurrency Control Schemes." *Proceedings of the Ninth International Conference on Very Large Databases*, Florence, 1983, pp. 97-108.

Pirkul, H. "Configuring Distributed Computer Systems with Online Database Backups." *Decision Support Systems*, Volume 3, 1987, pp. 37-46.

Ram, S. "The File Allocation Problem: An Expanded Perspective." *Proceedings of the Twentieth Hawaii International Conference on Systems Sciences*, January 1987, pp. 394-405.

Ram, S. "A Model for Designing Distributed Database Systems." *Information and Management*, Volume 17, 1989, pp. 169-180.

Ram, S., and Chastain, C. "Distributed Database Management Systems: An Architectural Survey." *Journal of Systems and Software*, Volume 10, Number 2, September 1989, pp. 77-95.

Ram, S., and Marsten, R. "A Model for Database Allocation Incorporating a Concurrency Control Mechanism." Working Paper, Department of MIS, University of Arizona, Tucson, 1989.

Ram, S., and Narasimhan, S. "Models for Database Allocation." Working Paper, Department of MIS, University of Arizona, Tucson, 1990.

Sacca, D., and Wiederhold, G. "Database Partitioning in a Cluster of Processors." *ACM Transactions on Database Systems*, Volume 10, Number 1, 1985.

Stonebraker, M. "Concurrency Control and Consistency of Multiple Copies of Data in Distributed Ingres." *IEEE Transactions on Software Engineering*, Volume SE-5, Number 3, 1979.

Ullman, J. D. *Principles of Database Systems*. Rockville, Maryland: Computer Science Press, 1988.

Wah, B. W. "File Placement on Distributed Computer Systems." *IEEE Computer*, 1984, pp. 23-32.

## 7. ENDNOTES

1. The work of this author is supported by a grant from the Small Grants Award Program, University of Arizona, Tucson.

2. These are only sample scenarios. In each case the values of input parameters were verified to observe trends in allocation.

## APPENDIX 1

## NOTATIONS USED IN THE PAPER

| | | |
|---|---|---|
| $N$ | = | Set of nodes |
| $D$ | = | Set of databases |
| $i, k, m$ | = | Denotes a node, $i, k, \in N$ |
| $d$ | = | Denotes a database, $d \in D$ |
| $n_i$ | = | Number of queries originating at node $i$ per unit time |
| $m_i$ | = | Number of updates originating at node $i$ per unit time |
| $f_{id}$ | = | Fraction of queries originating at node $i$ requiring data item(s) from database $d$ |
| $g_{id}$ | = | Fraction of updates originating at node $i$ requiring data item(s) from database $d$ |
| $\delta_i$ | = | Size of a query originating at node $i$ (number of packets) |
| $\delta_i'$ | = | Size of an update originating at node $i$ (number of packets) |
| $C_{ik}$ | = | Communication cost per packet from node $i$ to node $k$ (for queries) |
| $U_{ik}$ | = | Communications cost per packet from node $i$ to node $k$ (for updates) |
| $\alpha, \beta$ | = | Expansion factors for queries |
| $\alpha', \beta', \gamma$ | = | Expansion factors for updates |
| $CAP_k$ | = | Processing capacity at node $k$ per unit time |
| $S_{kd}$ | = | Storage cost of database $d$ at node $k$ per unit time |
| $C^w$ | = | Cost of waiting per message per unit time |

## APPENDIX 2

## OBJECTIVE FUNCTION COEFFICIENTS

### COMMUNICATION COST FOR READ-ONLY QUERIES

1. Request for lock $Q1 = C_{im} * n_i * f_{id} * \delta_i$
2. Granting of lock $Q2 = C_{im} * n_i * f_{id} * \delta_i$
3. Request for data item from site B $Q3 = C_{mk} * n_i * f_{id} * \alpha * \delta_i$
4. Data item sent from site B to site A $Q4 = C_{ki} * n_i * f_{id} * \beta * \delta_i$
5. Release of lock $Q5 = C_{im} * n_i * f_{id} * \delta_i$

### COMMUNICATION COST FOR UPDATES

1. Request for lock $U1 = U_{im} * m_i * g_{id} * \delta_i'$
2. Granting of lock $U2 = U_{im} * m_i * g_{id} * \delta_i'$
3. Request for data item from site B $U3 = U_{mk} * m_i * g_{id} * \alpha' * \delta_i'$
4. Data item value sent from site B to site A $U4 = U_{ki} * m_i * g_{id} * \beta' * \delta_i'$
5. Updated value sent from site A to central node $U5 = U_{im} * m_i * g_{id} * \gamma * \delta_i'$
6. Updated value from central node to all relevant sites $U6 = U_{ml} * m_i * g_{id} * \gamma * \delta_i'$
7. Release of lock $U7 = U_{im} * m_i * g_{id} * \delta_i'$

## APPENDIX 3

## A LAGRANGIAN RELAXATION

A Lagrangian Relaxation of the problem can be formed by relaxing constraint sets (4), (5), and (7). We then get the following Lagrangian function:

*Problem-L*

$$Z_L = Min. \sum_{i\in N} \sum_{m\in N} (Q1 + Q2 + Q5 + U1 + U2 + U5 + U7)\, z_m$$

$$+ \sum_{i\in N} \sum_{d\in D} \sum_{k\in N} \sum_{m\in N} (Q3 + Q4 + U3 + U4)\, y_{imkd} + \sum_{i\in N} \sum_{d\in D} \sum_{k\in N} \sum_{m\in N} U6 x_{kd}$$

$$+ \sum_{k\in N} C^w \frac{f_k}{(1-f_k)} + \sum_{k\in N} \sum_{d\in D} S_{k,d}^{*}\, x_{k,d}$$

$$+ \sum_{i\in N} \sum_{m\in N} \sum_{k\in N} \sum_{d\in D} \psi_{imkd}\, (y_{imkd} - x_{kd}) + \sum_{i\in N} \sum_{m\in N} \sum_{k\in N} \sum_{d\in D} \theta_{imkd}\, (y_{imkd} - z_m)$$

$$+ \sum_{k\in N} \lambda_k \Big\{ (1/CAP_k) \Big\{ \sum_{d\in D} \sum_{i\in N} \sum_{m\in N} n_i * f_{id} * y_{imkd} + (2 * z_k * \sum_{i\in N} n_i) + (2 * n_k)$$

$$+ \sum_{d\in D} \sum_{i\in N} \sum_{m\in N} m_i * g_{id} * y_{imkd} + (3 * z_k * \sum_{i\in N} m_i) + \sum_{d\in D} \sum_{i\in N} m_i * g_{id} * x_{kd}$$

$$+ (2 * m_k) \Big\} - f_k \Big\} \qquad (10)$$

*subject to:*

$$\sum_{m\in N} z_m = 1 \qquad (11)$$

$$\sum_{k\in N} x_{kd} \geq 1 \qquad \forall d\in D \qquad (12)$$

$$\sum_{m\in N} \sum_{k\in N} y_{imkd} = 1 \qquad \forall i\in N,\ \forall d\in D \qquad (13)$$

$$0 \leq f_k < 1 \qquad \forall k\in N \qquad (14)$$

$$y_{imkd},\ x_{kd},\ z_m \in \{0,1\} \qquad \forall m\in N,\ \forall i\in N,\ \forall k\in N,\ \forall d\in D \qquad (15)$$

**Problem-L can be decomposed into the following four sets of subproblems:**

*Problem-L1*

$$Z_{L1} = Min. \sum_{i \in N} \sum_{m \in N} (Q1 + Q2 + Q5 + U1 + U2 + U5 + U7)\, z_m$$

$$- \sum_{i \in N} \sum_{m \in N} \sum_{k \in N} \sum_{d \in D} \theta_{imkd}\, z_m + \sum_{m \in N} (\lambda_m / CAP_m) * (2 * \sum_{i \in N} n_i + 3 * \sum_{i \in N} m_i)\, z_m \quad (16)$$

*subject to:*

$$\sum_{m \in N} z_m = 1 \quad (17)$$

$$z_m \in \{0,1\} \qquad \forall m \in N \quad (18)$$

For $d = 1,.../D/,$

*Problem-L2(d)*

$$Z_{L2}(d) = Min. \sum_{i \in N} \sum_{k \in N} \sum_{m \in N} U6 x_{kd} + \sum_{k \in N} S_{kd} x_{kd} - \sum_{i \in N} \sum_{k \in N} \sum_{m \in N} \psi_{imkd} x_{kd}$$

$$+ \sum_{i \in N} \sum_{k \in N} (\lambda_k / CAP_k) m_i * g_{i,d} * x_{kd} \quad (19)$$

*subject to:*

$$\sum_{k \in N} x_{kd} \geq 1 \quad (20)$$

$$x_{kd} \in \{0,1\} \qquad \forall k \in N \quad (21)$$

For $i = 1,...,|N|$, $d = 1,...,|D|$,

*Problem-L3(i,d)*

$$Z_{L3}(i,d) = Min. \sum_{m\in N}\sum_{k\in N}(Q3 + Q4 + U3 + U4)\, y_{imkd} + \sum_{m\in N}\sum_{k\in N}\psi_{imkd}y_{imkd}$$

$$+ \sum_{m\in N}\sum_{k\in N}\theta_{imkd}y_{imkd} + \sum_{k\in N}\sum_{m\in N}(\lambda_k/CAP_k)*n_i*f_{id}*y_{imkd}$$

$$+ \sum_{k\in N}\sum_{m\in N}(\lambda_k/CAP_k)*m_i*g_{id}*y_{imkd} \qquad (22)$$

*subject to:*

$$\sum_{m\in N}\sum_{k\in N} y_{imkd} = 1 \qquad (23)$$

$$y_{imkd} \in \{0,1\} \qquad \forall m\in N, \forall k\in N \qquad (24)$$

For $k = 1,...,|N|$,

*Problem-L4(k)*

$$Z_{L4}(k) = Min.\ C^w\frac{f_k}{(1-f_k)} - \lambda_k f_k \qquad (25)$$

*subject to:*

$$0 \le f_k < 1 \qquad (26)$$