AT&T Technical Journal
Vol. 64, No. 2, February 1985
Printed in U.S.A.

Models for Configuring Large-Scale Distributed Computing Systems

By B. GAVISH*

(Manuscript received October 13, 1983)

This paper presents a model for designing the architecture of a distributed computing system of the type used to support the management and control activities of a large corporation. The input to the design process consists of two major data components describing the inputs and outputs to the information system and the relationships between them. Based on this information and the structure of communication and processing costs, an optimization model is formulated. It aggregates transactions in distributed databases, selects the locations in which those databases will be placed, assigns data sources to those databases, and selects for each report the report generation location. The problem is formulated as a combinatorial optimization problem and procedures are developed for computing lower bounds on the value of the optimal solution and heuristics for generating good feasible solutions for the problem. The procedures were tested on several examples and have generated good initial designs. Computational examples are presented to design problems including organizations with a hierarchical structure.

I. INTRODUCTION

Providing data collection, storage, and retrieval capabilities to industrial service and governmental organizations is a primary responsibility of management information systems and of operations systems in those organizations. Traditionally, such services have been provided by centralized computing systems, using large computers with cen-

* University of Rochester, N.Y. Part of this research was done while Mr. Gavish was an employee at AT&T Bell Laboratories.

tralized databases and communication networks. Due to the increased dissatisfaction of users with the quality of services provided by centralized systems, distributed computing systems are evolving as superior substitutes to many of those large centralized systems, for providing data storage and retrieval services. Distributed computer systems offer many potential advantages to those organizations. Those advantages include:

- Reduction in data processing costs by trading communications for processing costs
- Easier and smoother introduction of new hardware and software into the system
- Shorter response time
- Direct user control over computing resources to allow for setting up their own scheduling priorities
- Establishment of site autonomy and direct user responsibility and control over their application programs, and the quality and timeliness of the data that they generate and collect.

Distributed computing systems present system designers with many challenges, some of which already have been encountered in centralized computing systems and were easier to solve in a centralized setting, while others are new and unique to a distributed environment. These include technical issues such as: system integrity; concurrency control;[1-4] query parsing and decomposition;[5] addressing, naming, and directory management; backup recovery;[6] and security.[7] Other issues involve managerial and design problems that are faced by the managers and designers of those systems. These include: decisions on the allocation of files and databases to computer locations;[8-12] placement of processors and allocation of databases to those processors;[13-15] query optimization;[16-18] directory assignment; and design and analysis of the communication subnetworks.[19]

The potential promised by distributed systems and the importance of the above questions have attracted the attention of many researchers. Several models and procedures have been developed for answering some of those design issues. All of the existing models for file, database, or processor allocation assume that the physical and logical designs of centralized files or databases are given, and that the primary responsibility of the system designer is to assign those files to computer locations.

The design issues are relatively well understood when the files and databases have been predetermined. Unfortunately, in reality, the situation faced by a distributed system architecture designer is inherently different. At the beginning of the design and analysis process, the files and databases are either not defined or, if they are given, they are expressed in terms that are recognizable and acceptable for the design of a centralized computing system. If they are left unchanged, however, they could severely restrict the performance of a distributed system. This paper is a first attempt to define, formulate, and develop design tools for cases in which the structure of the databases is not given in advance. The underlying theme of this paper is the view that it is the responsibility of the analyst and system designer to determine the structure of the system databases and to allocate them to computer locations.

The input to the design process consists, in part, of the set of locations (data sources) from which data are recorded, collected, and transferred to the system databases. The data stored in the databases are used in report generation processes to generate periodic and on-demand reports, or provide responses to queries. These reports are used by the companies that are served by the distributed computing system. The final destinations, volume, and generation frequency of those reports constitute part of the input to the system design process.

The data that are stored in the databases arrive there as transactions that have originated at the system data sources. The transactions report on events that have occurred in those locations.

Many reports are generated from data stored in the system databases and are distributed to end users located in different destinations. A report generation process might need as input data that are retrieved from several databases located in distinct locations. The retrieved data have to be sorted, aggregated, and edited in order to generate the requested report; this report will then be distributed to those who use it as part of their regular activities. Different reports might be generated in different locations. The decisions about which location should generate each one of the reports are based on which locations hold the databases that provide input to the report generation process and on the final destinations to which the edited reports have to be transmitted.

The problem that is being addressed in this paper is as follows. The designer is given detailed information on data sources, the volume of update transactions that originate in each one of those locations, the list of reports and queries that have to be generated, their frequency, the volume of output generated, and their distribution to final users. For each report, the designer is also given the set of update transactions needed to generate it, the location (data sources) from which those transactions originate, and their volume per production run. A clear distinction is made between the locations from which update transactions originate and the locations in which those transactions are stored as part of the systems database.

The system designer must decide:

- How many processors to have in the system.

- What constitutes a database, i.e., what aggregations of update transactions form each one of the system databases, and how to partition those databases to realize some of the advantages offered by a distributed system.
- Placement of computers and databases to locations, i.e., which locations will be selected for placement of processing capabilities.
- Assignment of data sources to database locations, i.e., to which database location will the users in each data source send their transactions.
- Report generation locations, the location in which each report will be generated, from which databases the data will be retrieved, and through what distribution channels the edited reports will be sent to the end users.

Many factors have to be taken into consideration when making those decisions. These include operational restrictions, such as response time, reliability, availability, and security constraints; and many cost factors, such as:

- Setup and operational costs for processors, storage devices, and access rights to communication channels.
- The costs involved in transferring transactions from their origination points (data sources) to their databases.
- The cost of updating and maintaining the databases.
- The cost of retrieving data needed for report generation from the databases.
- The cost of transferring the retrieved data from the data retrieval locations to the locations in which the reports are generated.
- The cost of generating a report in a location. Those costs might depend on the location in which the report is being processed.
- The cost of distributing the edited reports to the end users, which could take the form of physical distribution of the printed reports through a carrier or mail service, or through the communication network.

It is highly unrealistic to assume that in a single model we will be able to encompass all of those design issues. Instead we concentrate on the development of a simplified model that addresses the major cost factors influencing the design of a distributed computer system. The results obtained from such a model should be taken with caution and viewed as general guidelines for an initial design that can be used as input to the detailed design of such a system.

An integral part of every systems analysis and design process includes the identification and definition of the sets of inputs (transactions) and outputs (reports) from and to the different entities of the managed system. The major objective of the design process is to decide on a data collection and report generation strategy that will minimize the system costs. To the best of our knowledge in spite of its practical and fundamental importance, this problem has not been addressed in the open literature, neither for centralized systems nor for distributed systems.

In the next section we present the principal considerations, the setting under which the model has been developed, and the assumptions used for a mathematical formulation of the problem. These are followed by a detailed description of the cost factors that affect the design of a distributed computing system. The problem is formulated in Section IV as a quadratic integer programming problem followed by a linear integer programming formulation of the problem. Several relaxations of the problem are introduced in Section V. The problem addressed here belongs to the class of NP-complete problems. In Section VI, heuristics are suggested to obtain approximate solutions to the problem. The heuristics and the lower bounding procedures were tested on several numerical examples and have generated good initial designs for those cases. Some of the computational results and a numerical example are presented in Section VII. In Section VIII, we present possible extensions of this model. The last section contains conclusions and suggestions for further research.

The following terminology is used throughout the paper: transactions stand for update transactions, while reports stand for read-only transactions and include reports as well as query transactions.

II. UNDERLYING ASSUMPTIONS

The problem of configuring a general-purpose large-scale distributed computing system is a complex task. Therefore we restrict the formulation and analysis by using simplifying assumptions. The following assumptions are used to formulate the distributed system architecture problem. In subsequent sections we will show how some of those assumptions can be relaxed and incorporated in the model, thus extending the range of cases that can be handled by this and related models.

Assumption 1—(No splitting) All the transactions that originate in a data source are routed to the same set of computer locations. This assumption highly simplifies the directory for routing transactions from their origination location to the database in which they are stored. Under this scheme each data source has a unique set of addresses to which all its transactions are forwarded.

Assumption 2—(No duplication) A single computer location and its associated database handle all the transactions that originate from the same data source. However, one or more data source locations can be assigned to the same computer.

Assumptions 1 and 2 exclude the possibility of having multiple copies of the same file in the system. The fact that the model is restricted to single-copy databases eliminates the need to consider the nonnegligible overhead required for the synchronization and concurrency control for updating multiple copies of the same database. This highly simplifies the analysis and modeling of the problem. In the last section of the paper we discuss situations in which multiple copies of databases are possible.

Assumption 3—Databases can be assigned only to locations that contain computers with significant processing capacity. A distinction is made between I/O processors and processors whose main activity is arithmetic/logic operations. This implies that assigning a database to a location requires the assignment of a processor to the same location.

Assumption 4—The processing of a report is done in a single location. All the data that are needed for processing a report are sent on demand to that location. This assumption does not exclude the possibility that only a fraction of the transactions will be sent from the database to the report generation location. This fraction depends on the selectivity and compression/aggregation of the data retrieved at the database location. The portion of data that is actually transferred to the report generation location is not a collection of the original transactions. Typically it is only the compressed/aggregated content of the data selected from the databases.

Assumption 5—Reports can be generated only in locations that contain computers.

Assumption 6—Every report is generated in a separate report generation process. The input, intermediate, or final results of one report are not used as input for the generation process of another report (in the same location or some other location). In reality it is possible to save on processing or communication costs by combining a few report generation processes into a sequence of report generation processes. By doing it, it is possible to eliminate duplication in the retrieval, transfer, or editing steps of the report generation process and reduce the systems costs.

This assumption is clearly applicable to cases where reports can be initiated by users at any point in time. This eliminates the possibility of coordinating the report generation processes between the different users. However, it does not prohibit the possibility of combining the generation processes of reports that are always generated in the same run into a single report generation process. The model views the combination of such reports as a single report generation process with outputs being distributed to one or more final destinations. The only restriction that is imposed on such a combination is the requirement to specify beforehand the inputs and outputs from and to the combined process.

Using the above assumptions, four types of locations can be defined.

D_1—is the set of data sources. This is the set of locations from which data are collected and recorded to support the operations (decision making and control) of the organization that is being served by the distributed computer system. This is the set of locations in which people or automatic devices observe or detect changes in the state of the system; record them on forms or some other recording devices; and feed them through displays, data collection equipment, communication networks, optical scanners, or magnetic ink readers to the system databases.

Examples of transaction initiators are bank tellers in a bank branch, window clerks in a warehouse, sales personnel in a marketing group, cashiers in a supermarket, or surveying and testing devices located in central offices generating automatic alarms or responses to equipment and facility testing requests in the telephone system. It is unrealistic to assume that the model developed subsequently will be able to capture this level of detail. Therefore a data source location is defined as an aggregation of all the transactions that were initiated from the same geographical area. This might be the set of transactions originating from the same bank branch (which is an aggregation of the tellers in that branch), the set of banking transactions that originate from the same city/region (this is an aggregation of bank branches), the set of transactions that have originated in the same supermarket store, or all the transactions that originate in the same plant/laboratory or warehouse.

D_2—is the set of places in which end users are located. Those users request reports that are based on data and transactions that have originated from locations in D_1. Those are reports used by the headquarters, regional managers, or clerks. Depending on the organizational structure of the corporation, D_2 can include locations that are identical to locations in D_1, or locations that are not part of D_1.

D_3—is the set of locations in which databases are stored. We assume that the existence of a database in a location implies the presence of a processor in the same location.

D_4—is the set of locations at which reports are generated. This implies the availability of computing capacity in those locations. The intersection of D_4 and D_3 typically is not empty. A location that belongs to D_3 and D_4 is a location with a computer that maintains a database and is also used to generate reports for some users.

III. COST STRUCTURE OF THE PROBLEM

The selection of a final configuration of a distributed computing system depends to a large extent on many cost factors. Some of those cost components are dependent on the amount of activity in the system and could be nonlinear functions of the distance, number of transactions, or storage requirements. In this section we outline the major cost components that influence such a design. Distributed computer systems are expected to be operational over a long lifetime, with different expenses incurred at varying stages of the system life cycle. In order to bring those expenses into a common denominator, they are discounted to their present value, using appropriate discounting factors.

3.1 Setup and operational costs

These are the setup and operational costs of placing a computer and appropriate storage devices in a location. These costs are composed of purchasing/rental, installation, and maintenance of hardware and software; the physical facilities in which they are installed; and the staffing requirements for production runs, maintenance, and development activities. They include only cost factors that are independent of the amount of activity from and to that site, in terms of update transactions or retrieval requests.

3.2 Data collection transfer and update costs

Data collection transfer and update costs include:

- The costs for recording as transactions events that occur at data source locations
- Routing all the transactions that originate in a data source to their database location
- Editing, verifying, validating, and updating the database with correct transactions
- Sending a transaction update confirmation response to the transaction originator and handling the errors discovered during that process.

Those costs are composed of processing, communications, and storage costs, and depend on

- The number of transactions processed over the planning period
- The length (in bytes) of an average transaction
- The duration in time that individual transactions are kept as active detailed on line data in the database
- The amount of storage required to store the stable part of the database that contains data on the entities that belong to its data sources.

The cost expressions make a clear distinction between the traffic intensity (measured by the number of transactions) from a given source to its database, and the amount of storage needed to store the relevant data pertaining to that source. The on-line storage requirements for a single data source are composed of two components. They are tightly coupled to the amount of data held on the different entities in the data source, and are also a function of the number of transactions held as active transactions in the database.

Using a banking example, we can have bank branches that have relatively few accounts, but the average account holders in those branches are very active and therefore many transactions are generated from and to those accounts. On the other hand, we can have branches that have many accounts, with very little activity going on in those accounts. In this specific example, there is very little relationship between the number of accounts (which are the dominant factor in determining the on-line storage requirements) and the average activity in those accounts (which determines the communication, processing, and update costs).

The data collection, transfer, and update costs between points i and j are given by the following cost expressions:

$$C_{ij}^{(1)}(A_i,E_i) = f_i^{(1)}(A_i,E_i) + f_{ij}^{(1)}(A_i,E_i) + g_j^{(1)}(A_i,E_i),$$

where:

A_i—is the number of transactions per time period that originate from data source i, $i \in I$.

E_i—is the expected number of entities in data source i, $j \in I$ for which data are kept in the system databases.

$f_i^{(1)}(A_i,E_i)$—captures the costs for data recording, collection, and handling of correct and rejected transactions, in data source i. These costs are primarily dependent and proportional to A_i, in some cases it is also dependent on E_i.

$f_{ij}^{(1)}(A_i,E_i)$—captures the cost of transferring transactions from data source i to a database in location j; this also includes the cost of sending responses to the transaction originators on updated transactions, and the resubmission of transactions that have been rejected by the system. The expression $f_{ij}^{(1)}(\cdot)$ consists mainly of communication costs between points i and j, and is proportional to the number of messages transmitted and to the average message length.

$g_j^{(1)}(A_i,E_i)$—captures the processing costs for editing, verifying, validating, and updating in location j transactions that have originated from source i. The expression $g_j^{(1)}(\cdot)$ also includes the storage costs for storing in location j data that have originated or are relevant for processing transactions that have originated from data source i. The storage costs depend mainly on the amount of storage needed to store information on entities in E_i

and to keep readily accessible in location j a subset of the transactions that are submitted from i to j.

3.3 Data retrieval and transfer costs

These include the costs for retrieving from a database in location j data that are needed for generating report r, $r \in R$, and transferring the retrieved data to the location in which report r is generated. The data retrieval costs over the planning horizon depend on the frequency in which report r is generated and consist of:

- Data processing costs for retrieving data from the databases, and sorting and aggregating the retrieved data at the data retrieval location (in order to reduce the amount of data that has to be transferred to the report generation location)
- Data communication costs for transferring the retrieved data from the data retrieval location to the report generation location. The amount of data that has to be transferred depends on the selectivity of the retrieved data (the ratio between the amount of data collected in the database and the portion that is actually retrieved for a specific report). In addition, the retrieved data are also aggregated, which further reduces the volume of data that has to be transferred from the system databases to the report generation locations.

Using the assumption that those costs are separable over data sources, the following expressions give the cost of retrieving from a database in location j data that have originated in data source i, $j \in S_r$, are stored in location j, and are used as input to the generation process of report r, $r \in R$. Given that report r is generated in location k,

$$C^{(2)}_{ijkr}(A_i,E_i,S_{ir}) = f^{(2)}_{ijr}(A_i,E_i,S_{ir}) + f^{(2)}_{ijkr}(A_i,E_i,S_{ir})$$

S_{ir} = the selectivity factor between the amount of data entered into the database from data source i, and the amount of data that is retrieved from the same database for report r.

$f^{(2)}_{ijr}(A_i,E_i,S_{ir})$ = the data retrieval and processing costs (in location j) for retrieving from a database in location j, data that have originated from source i, $i \in S_r$, and are needed as input to the report generation process of report r.

$f^{(2)}_{ijkr}(A_i,E_i,S_{ir})$ = the communication costs for transferring the retrieved data from the data retrieval location j to the report generation location k. These costs are proportional to the amount of data that has to be transferred between the two locations (j and k), and the characteristics of the data source i and the requested report r.

3.4 Report processing editing and distribution costs

These are costs for generating report r in location k and for distributing the edited report to the end users of that report. They are incurred from the moment that all the data that are needed to generate report r are available at the report generation location (k). They are composed of processing costs in location k for combining the different data sets that were transferred to location k from database locations that contain data needed for report r. These data are sorted, aggregated, and edited to form report r. Communication costs are incurred when the edited reports are transferred to the end users. Since those cost components are not dependent on the locations from which the input data have been retrieved they are expressed as:

$$C_{kr}^{(3)}(\bar{A}_r,\bar{E}_r,N_r) = f_{kr}^{(3)}(\bar{A}_r,\bar{E}_r) + g_{kr}^{(3)}(N_r),$$

where:

$\bar{A}_r$ = the volume of data that is used as input to report r. It is based on transactions collected from all the data sources that provide input for report r.

$\bar{E}_r$ = the amount of data that is related to all the entities whose data are being used as input to report r.

N_r = the set of locations that contain end users of report r. This is the set of locations to which the report will be distributed.

$f_{kr}^{(3)}(\bar{A}_r,\bar{E}_r)$ = the generation costs (I/O, processing, and memory) of report r in location k. They are a function of $\bar{A}_r,\bar{E}_r$ and the frequency with which report r is being generated over the planning horizon.

$g_{kr}^{(3)}(N_r)$ = the distribution costs of report r, from location k in which it is generated, to the set of end users of this report.

Large amounts of data have to be collected and analyzed in order to provide input to a system configuration model. This is a time-consuming and costly effort that might preclude the use of such a design tool owing to excessive data processing and preparation costs. Fortunately, under appropriate assumptions, some of those cost components are identical for different system configurations and are irrelevant for cost comparisons, and therefore need not be collected. For example, if the cost of generating report r does not depend on the location in which the report is being generated, then there is no need to specify $f_{kr}^{(3)}(\bar{A}_r,\bar{E}_r)$ in the design model. Similarly, if the costs for recording, collection, and handling of data in data source i are independent of the location in which those data are finally stored, then there is no need to specify the values of $f_i^{(1)}(A_i, E_i)$ in $C_{ij}^{(1)}(A_i, E_i)$. By paying careful attention to which cost components are included in the model, the amount of effort needed for data collection and analysis can be reduced significantly.

IV. MATHEMATICAL FORMULATION OF THE PROBLEM

In this section we present two mathematical programming formulations of the distributed system architecture design problem. The first formulation is an integer programming problem with a linear constraint set and a quadratic objective function. This class of problems is extremely difficult to solve. Therefore, in the second formulation, the objective function is linearized by defining additional variables and constraints.

To formulate the problem, we define the following decision variables:

Z_j = a binary variable equal to one if a computer and database are placed in location j, and equal to zero otherwise.

X_{ij} = a binary variable equal to one if all the transactions that originate from a data source i are routed to a database in location j, X_{ij}, and equal to zero otherwise.

Y_{kr} = a binary variable equal to one if report r is assigned to be generated in location k, and equal to zero otherwise.

The distributed system architecture problem is formulated as:

Problem QIP:

Find binary variables Z_j, X_{ij}, Y_{kr} that satisfy:

$$Z_{\text{QIP}} = \text{Min} \left\{ \sum_{j \in J} Z_j P_j + \sum_{i \in I} \sum_{j \in J} X_{ij} V_{ij} + \sum_{r \in R} \sum_{k \in J} Y_{kr} U_{kr} + \sum_{r \in R} \sum_{i \in S_r} \sum_{j \in J} \sum_{k \in J} X_{ij} Y_{kr} W_{ijkr} \right\}$$

subject to:

$$\sum_{j \in J} X_{ij} = 1 \qquad i \in I \qquad (1)$$

$$X_{ij} \le Z_j \qquad i \in I, j \in J \qquad (2)$$

$$\sum_{k \in J} Y_{kr} = 1 \qquad r \in R \qquad (3)$$

$$Y_{kr} \le Z_k \qquad r \in R, k \in J \qquad (4)$$

$$X_{ij}, Y_{jr}, Z_j = 0 \text{ or } 1 \qquad i \in I, j \in J, r \in R. \qquad (5)$$

The constraints in (1) ensure that all the transactions that originate in data source i will be routed to exactly one of the locations that are candidates for database (and processor) placement; the constraints in (2) ensure that transactions will be sent from source i to location j only if a database has been placed in location j. Similarly, the constraints in (3) guarantee that each report will be assigned to some report generation location, while the constraints in (4) prevent the assignment of a report generation process to a location that does not have processing capabilities.

The objective function consists of the following cost components:

P_j = the cost of placing a computer and a database in location j. We assume here that placing a processor (or a database) in location j implies that the capabilities for handling a database (or report generation) have also been acquired. Those setup costs are incurred only when $Z_j = 1$.

V_{ij} = the marginal cost over the planning horizon for sending all the transactions that originate from data source i to update a database that is placed in location j. This includes the communication costs for transferring the transactions from location i to location j; the transaction processing costs in location j for verifying, validating those transactions, and updating them in the database; storage costs for storing those transactions in the database; and the costs of sending rejection or acceptance responses from location j to data source i on the status of those transactions. Those costs are incurred only if $X_{ij} = 1$.

U_{kr} = the marginal cost over the planning horizon for processing report r in location k. This includes the costs for editing the report in location k and for distributing its outputs to end users. U_{kr} is incurred only when $Y_{kr} = 1$.

W_{ijkr} = the marginal cost over the planning horizon of retrieving for report r, which is generated in location k, data that originated in data source i, $i \in S_r$, which is currently stored in a database in location j. This includes the cost of retrieving data from the database, selecting and aggregating it in location j and transferring the aggregated data from location j to location k, and preparing it as input to the process that generates report r (in location k). This cost is incurred only if the multiplication of X_{ij} by $Y_{kr} = 1$, i.e., the transactions collected from data source i have been assigned to a database in location j, and report r has been assigned to a processor in location k.

The mathematical programming formulation of *Problem QIP* is a quadratic integer programming problem. It has a linear constraint set and a quadratic objective function. The present state of the art in solving nonlinear integer programming problems does not hold much promise of solving those types of problems in reasonable amounts of computing times. The objective function can be linearized by defining a new set of binary variables, Ψ_{ijkr}, $r \in R$, $i \in S_r$, $j \in J$, $k \in J$. Ψ_{ijkr} is equal to one when report r is assigned to be generated at location k and it uses transactions that have originated from data source $i, i \in S_r$, and were stored in a database in location j. Ψ_{ijkr} is zero otherwise.

The distributed system architecture problem is reformulated as:

Problem ILP:

Find binary variables Z_j, X_{ij}, Y_{kr}, Ψ_{ijkr} that satisfy:

$$Z_{\text{ILP}} = \text{Min}\left\{\sum_{j\in J} Z_j P_j + \sum_{i\in I}\sum_{j\in J} X_{ij} V_{ij} + \sum_{r\in R}\sum_{k\in J} Y_{kr} U_{kr} + \sum_{r\in R}\sum_{j\in J}\sum_{k\in J}\sum_{i\in S_r} \Psi_{ijkr} W_{ijkr}\right\}$$

subject to:

$$\sum_{j\in J} X_{ij} = 1 \qquad i \in I \tag{6}$$

$$X_{ij} \le Z_j \qquad i \in I, j \in J \tag{7}$$

$$\sum_{k\in J} Y_{kr} = 1 \qquad r \in R \tag{8}$$

$$Y_{kr} \le Z_k \qquad r \in R, k \in J \tag{9}$$

$$\Psi_{ijkr} \le X_{ij} \qquad j \in J, k \in J, r \in R, i \in S_r \tag{10}$$

$$\Psi_{ijkr} \le Y_{kr} \qquad j \in J, k \in J, r \in R, i \in S_r \tag{11}$$

$$\sum_{j\in J}\sum_{k\in J} \Psi_{ijkr} = 1 \qquad r \in R, i \in S_r \tag{12}$$

$$Z_j, X_{ij}, Y_{kr}, \Psi_{ijkr} = 0 \text{ or } 1 \qquad i \in I, j \in J, k \in J, r \in R \tag{13}$$

The constraints in eqs. (6) through (9) have the same interpretation as constraints (1) through (4) in the quadratic programming formulation of the problem. The constraints in (10) ensure that reports that use data originating from data source i will be able to retrieve it from a database in location j, only if the transactions originating from location i have been assigned to update a database placed in location j. The constraints in (11) ensure that the data needed for report r are sent to location k only if report r is actually generated in location k. By definition, report r uses transactions that originate in location $i, i \in S_r$. These transactions must first be routed and update a database. When report r is requested, the data are retrieved from the database and sent to the location in which report r is generated. The constraints in (12) ensure that such a routing to and from a database actually takes place for each combination of $r \in R$ and $i \in S_r$.

V. GENERATING LOWER BOUNDS ON THE OPTIMAL SOLUTION

The problem of optimally configuring a distributed computing system is a combinatorial optimization problem. It belongs to the class of NP complete problems. This statement can be proved by setting $W_{ijkr} = 0 \forall i, j, k, r$. The quadratic terms in *Problem QIP* are eliminated and the problem is reduced to an uncapacitated plant location problem, which has been proven to be in the NP complete class.[20] This observation implies that it is unlikely that an algorithm will be developed that is capable of solving every instance of the problem to optimality. Moreover, using similar arguments it can be shown that unless $P = NP$, it is not possible to develop a polynomial time algorithm which produces approximate solutions that have a guaranteed absolute error bound. The existence of such an approximation scheme implies the ability to generate feasible solutions that have an objective function value within a user-predefined error tolerance from the optimal solution. Given this situation, the only alternative is for a system designer to resort to heuristics. Those heuristics can generate feasible solutions for the problem. However, they are not guaranteed to generate an optimal solution. Many heuristics have been developed for a variety of combinatorial optimization problems. The practical experience gained in using those heuristics is quite encouraging. Extensive computational experimentation with many of those heuristics shows that in many cases they generate solutions that are very close to the optimal solution.

The solutions generated by the heuristics are feasible to the problem of configuring distributed systems, and as such provide an upper bound on the value of the (unknown) optimal solution. Since the value of the optimal solution to the system configuration problem is unknown, to evaluate the quality of the solutions generated by the heuristics, we develop in this section methods for computing lower bounds on the value of the optimal solution. The difference between the value of the best upper bound generated by the heuristic and the tightest lower bound generated by the developed method is clearly an upper bound on the gap between the value of the heuristic solution and the true optimal solution. Keeping in mind that the data supplied to those models contain errors that in many cases exceed the errors introduced by the heuristic, these heuristics seem to provide a plausible option for a system designer.

We present two methods for computing lower bounds. The first method is based on a Lagrangian relaxation of *Problem ILP*, which, together with a multiplier updating method, can produce tight lower bounds on the optimal solution value. This procedure requires extensive computational resources, which limits its applicability to cases in which the system designer is willing to incur those expenses. As an alternative to this time-consuming procedure, a second method is developed for computing lower bounds. This procedure generates lower bounds that are not as tight as the Lagrangian-based procedure, but are easier to implement and require only a modest amount of computing resources. It is a preferable alternative for design problems in which a rough estimate on the quality of the solution suffices.

5.1 The Lagrangian relaxation procedure

The Lagrangian relaxation of *Problem ILP* is formed by multiplying the constraints in (12) by a vector $\{\lambda_{ir}\}$ of Lagrange multipliers, and the constraints in (11) with the vector $\{\beta_{ijkr}\}$, and adding them to the objective function, forming the following Lagrangian problem:

$$L(\lambda, \beta) = \min \left\{ \sum_{j \in J} Z_j P_j + \sum_{i \in I} \sum_{j \in J} X_{ij} V_{ij} + \sum_{r \in R} \sum_{k \in J} Y_{kr} U_{kr} \right.$$
$$+ \sum_{r \in R} \sum_{j \in J} \sum_{k \in J} \sum_{i \in S_r} \Psi_{ijkr} W_{ijkr} + \sum_{r \in R} \sum_{i \in S_r} \lambda_{ir} \left(1 - \sum_{j \in J} \sum_{k \in J} \Psi_{ijkr} \right)$$
$$\left. + \sum_{r \in R} \sum_{j \in J} \sum_{k \in J} \sum_{i \in S_r} \beta_{ijkr} (Y_{kr} - \Psi_{ijkr}) \right\}$$

subject to eqs. (6) through (10) and (13).

After rearrangement and collection of terms that correspond to identical variable types, the objective function is reduced to:

$$L(\lambda, \beta) = \min \left\{ \sum_{j \in J} Z_j P_j + \sum_{i \in I} \sum_{j \in J} X_{ij} V_{ij} \right.$$
$$+ \sum_{r \in R} \sum_{k \in J} Y_{kr} \left(U_{kr} + \sum_{i \in S_r} \sum_{j \in J} \beta_{ijkr} \right) + \sum_{r \in R} \sum_{i \in S_r} \lambda_{ir}$$
$$\left. + \sum_{r \in R} \sum_{j \in J} \sum_{k \in J} \sum_{i \in S_r} \Psi_{ijkr} (W_{ijkr} - \beta_{ijkr} - \lambda_{ir}) \right\},$$

leading to the following optimization problem.

Problem LR

Find binary variables X_{ij}, Y_{ij}, Z_j, Ψ_{ijkr} that satisfy:

$$L(\lambda, \beta) = \sum_{r\in R}\sum_{i\in S_r} \lambda_{ir} + \min \left\{ \sum_{j\in J} Z_j P_j + \sum_{i\in I}\sum_{j\in J} X_{ij} V_{ij} + \sum_{r\in R}\sum_{k\in J} Y_{kr}\hat{U}_{kr} + \sum_{r\in R}\sum_{j\in J}\sum_{k\in J}\sum_{i\in S_r} \Psi_{ijkr}\hat{W}_{ijkr} \right\}$$

subject to the constraints in eqs. (6) through (10) and (13),
where

$$\hat{U}_{kr} = U_{kr} + \sum_{j\in J}\sum_{i\in S_r} \beta_{ijkr},$$

and

$$\hat{W}_{ijkr} = W_{ijkr} - \beta_{ijkr} - \lambda_{ir}.$$

Problem LR is relatively easy to solve, if we note that for a fixed vector $\hat{X}_{ij}$ of X_{ij} variables, the Ψ_{ijkr} variables in the Lagrangian problem must satisfy:

$$\hat{\Psi}_{ijkr} = \begin{cases} 0 \text{ if } \hat{X}_{ij} = 0, \\ 0 \text{ if } \hat{X}_{ij} = 1 \quad \text{and} \quad \hat{W}_{ijkr} \geq 0, \\ 1 \text{ if } \hat{X}_{ij} = 1 \quad \text{and} \quad \hat{W}_{ijkr} < 0, \end{cases}$$

where $\hat{\Psi}_{ijkr}$ is the optimal solution to *Problem LR* for a fixed set of Lagrange multipliers. From the above relationship it is easy to verify that:

$$\hat{\Psi}_{ijkr}\hat{W}_{ijkr} = \hat{X}_{ij} \min\{0; \hat{W}_{ijkr}\}.$$

Using this relation, *Problem LR* can be rewritten as:

$$L(\lambda, \beta) = \left\{ \sum_{j\in J} Z_j P_j + \sum_{i\in I}\sum_{j\in J} X_{ij}\hat{V}_{ij} + \sum_{r\in R}\sum_{j\in J} Y_{jr}\hat{U}_{jr} \right\} + \min \sum_{r\in R}\sum_{i\in S_r} \lambda_{ir} \qquad (14)$$

subject to:

$$\sum_{j\in J} X_{ij} = 1 \qquad i \in I, \qquad (15)$$

$$\sum_{j\in J} Y_{jr} = 1 \qquad r \in R, \qquad (16)$$

$$X_{ij} \leq Z_j \qquad i \in I, j \in J, \qquad (17)$$

$$Y_{jr} \leq Z_j \qquad j \in J, r \in R, \qquad (18)$$

$$Z_j, X_{ij}, Y_{jr} = 0 \text{ or } 1 \qquad i \in I, j \in J, r \in R, \qquad (19)$$

where

$$\hat{V}_{ij} = V_{ij} + \sum_{k\in J} \sum_{r\in R} \min\{0; \hat{W}_{ijkr}\}.$$

The problem given by eqs. (14) through (19) is convertible to a simple uncapacitated plant location problem, in which the set of candidate plant locations is equal to J, and the set of customer locations is given by the union of the sets I and R. Uncapacitated plant location problems are optimization problems that belong to the class of NP complete problems. As such, it is unlikely that an algorithm that is capable of solving every instance of the problem to optimality in polynomial time will be developed. However, ample empirical evidence exists showing that several algorithms that have been developed by several researchers for solving uncapacitated plant location problems can solve without too much difficulty many occurrences of the problem. Erlenkotter has developed such a code.[21] It is a branch and bound dual-based procedure for solving plant location problems. The procedure was implemented and tested on a large set of cases and was able to solve without much difficulty many plant location problems to optimality.

Given a fixed vector of Lagrange multipliers, it can be shown that $L(\lambda, \beta)$ is a lower bound on the value of the optimal solution to the distributed system configuration problem, i.e., $L(\lambda, \beta) \leq Z_{ILP}$. Different multiplier values will generate different values for $L(\lambda, \beta)$. We are interested in obtaining a set of multipliers $(\hat{\lambda}, \hat{\beta})$ that generate the tightest possible bounds on Z_{ILP}, i.e., they satisfy:

$$L(\hat{\lambda}, \hat{\beta}) = \max_{(\lambda,\beta)}\{L(\lambda, \beta)\}.$$

Obtaining the optimal multiplier values is not a simple task. $L(\lambda, \beta)$ is a nondifferentiable function, making a straightforward optimization of the multiplier values a difficult task. Several methods have been suggested in the literature for computing good approximations to $(\hat{\lambda}, \hat{\beta})$. Those procedures compute multiplier values that generate solutions that are very close to $L(\hat{\lambda}, \hat{\beta})$. Those procedures include methods such as column generation, dual ascent, multiplier adjustment, or subgradient optimization procedures.[22,23]

The subgradient optimization procedure is an iterative method that starts with an initial set of multiplier values and uses a multiplier updating scheme to improve the lower bound value. The procedure is initiated with an initial set of multiplier values and uses the following multiplier updating scheme when moving from one iteration to the next:

$$\lambda_{ir}^{p+1} = \lambda_{ir}^{p} - t_p \gamma_{ir}^{p},$$

$$\beta_{ijkr}^{p+1} = \beta_{ijkr}^{p} - t_p \gamma_{ijkr}^{p},$$

where λ_{ir}^{p} and β_{ijkr}^{p} are the multiplier values used in iteration p. The subgradient directions are given by:

$$\gamma_{ir}^{p} = 1 - \sum_{j\in J} \sum_{k\in J} \Psi_{ijkr}^{p},$$

$$\gamma_{ijkr}^{p} = Y_{kr}^{p} - \Psi_{ijkr}^{p}.$$

Y_{kr}^{p} and Ψ_{ijkr}^{p} are the optimal variable values for the Lagrangian problem at iteration p, and t_p is a step size. Poljack[24] proved that the sequence (λ^p, β^p) converges to $(\hat{\lambda}, \hat{\beta})$ if the step size t_p satisfies the conditions:

$$\lim_{p\to\infty} t_p = 0 \quad \text{and} \quad \sum_{p=1}^{\infty} t_p = \infty.$$

Unfortunately, such a sequence is not practical for a computer implementation. Therefore, a heuristic is used for computing the step size t_p. It is given by:

$$t_p = \delta_p \frac{\bar{Z} - L(\lambda^p, \beta^p)}{|\gamma^p|^2},$$

where $\bar{Z}$ is an overestimate on $L(\hat{\lambda}, \hat{\beta})$ and δ_p is a scalar in the range of zero to two. δ_p is initially set to two and is divided by two if, in a sequence of n iterations, there was no improvement in the Lagrangian value. The procedure is terminated when one of the following termination conditions is satisfied:

1. The number of iterations has exceeded an upper limit.
2. The difference between the upper and lower bound limits is below a given tolerance ϵ, where

$$\left| \frac{\bar{Z} - L(\lambda, \beta)}{L(\lambda, \beta)} \right| \leq \epsilon.$$

3. The step size t_p is small, $t_p \leq \epsilon_1$.
4. The scalar δ_p is small, $\delta_p \leq \epsilon_2$.

Extensive computational experience with the subgradient optimization procedure reveals that it is not sensitive to the overestimate $\bar{Z}$ value, or to the initial multiplier values. The procedure is sensitive to the value of the parameter n, which determines the rate of change in the δ_p value. Setting n to a high value implies that a large number of iterations will be needed before the procedure converges to a good solution. Setting n to a low value may lead to termination before the procedure had the time to converge. Appropriate n values can be determined only through computational experimentation.

The combination of Lagrangian relaxation and subgradient optimization procedures has been successfully applied to a variety of combinatorial optimization problems. Those include problems such as the traveling salesman problem,[23] multiple traveling salesman problems,[25] topological design of computer communication networks,[19,26] or routing in computer networks.[27] The application of the same procedure to the distributed system configuration problem is practical only for problems with a small number of reports and database locations. When applied to *Problem ILP*, the procedure requires the storage and update of $(|J|^2 \sum_{r\in R} |S_r|)$ multiplier values, leading to large storage requirements and to a relatively slow convergence.

5.2 A simple lower bounding method

The Lagrangian relaxation-based procedure for computing lower bounds to *Problem ILP* is a computationally expensive procedure, which might preclude its use for many system design problems. An alternative method for computing lower bounds that requires significantly less computing resources, but is expected to produce inferior bounds, is based on the quadratic programming formulation of the problem. The method is based on the following argument. If the quadratic terms in the objective function of *Problem QIP* are dropped, the optimization problem is reduced to a simple uncapacitated plant location problem. The cost of this plant location problem is clearly a lower bound on the value of the optimal solution to *Problem ILP.* This bound can be further strengthened by adding to the U_{kr} terms in the solution of the plant location problem an underestimate of the costs that have been neglected by eliminating the quadratic terms from the objective function. Those terms capture the costs for transferring from some unknown database locations to location k data that originated in source i that are needed for the generation of report r.

Letting W_{ikr} be the minimal cost of sending data that originate in data source i, $i \in S_r$ to some database location, retrieving it from that location, and transferring it to location k where report r is generated; that cost is given by $\min_{j\in J}\{W_{ijkr}\}$. By summing up those individual costs over all data sources that originate data for report r, a lower bound U_{kr} on the data retrieval and transfer costs for generating report r in location k is obtained:

$$\tilde{\tilde{U}}_{kr} = U_{kr} + \sum_{i\in S_r} \min_{j\in J}\{W_{ijkr}\} = U_{kr} + \sum_{i\in S_r} W_{ikr}\,.$$

Substituting $\tilde{\tilde{U}}_{kr}$ for U_{kr} in the relaxed problem leads to the following optimization problem.

Problem YLP

Find binary variables Z_j, X_{ij}, Y_{kr} that satisfy:

$$Z_{LB}^{Y} = \min \left\{ \sum_{j \in J} Z_j P_j + \sum_{i \in I} \sum_{j \in J} Z_{ij} V_{ij} + \sum_{r \in R} \sum_{j \in J} Y_{jr} \tilde{\bar{U}}_{jr} \right\}$$

subject to eqs. (15) through (19).

This optimization problem is a simple, uncapacitated plant location problem. Its solution provides a lower bound on the value of the optimal solution.

Theorem 1: $Z_{LB}^{Y} \leq Z_{\text{QIP}}$.

Proof: Let $(\bar{Z}^*, \bar{X}^*, \bar{Y}^*)$ be the optimal solution to *Problem QIP*, and $(\bar{Z}^Y, \bar{X}^Y, \bar{Y}^Y)$ the optimal solution to *Problem YLB*. $\square$

Letting j_i be the index of the database location to which data are sent and stored from data source i, in the optimal solution to *Problem QIP*. From the definition of W_{ikr}, it is clear that:

$$W_{ij_ikr} \geq W_{ikr} = \min_{j \in J} \{ W_{ijkr} \}.$$

The quadratic terms in the objective function of *Problem QIP* satisfy the following relation:

$$\sum_{r \in R} \sum_{j \in J} \sum_{k \in J} \sum_{r \in S_r} X_{ij}^* Y_{kr}^* W_{ijkr}$$

$$= \sum_{r \in R} \sum_{k \in J} Y_{kr}^* \sum_{i \in S_r} W_{ij_ikr} \geq \sum_{r \in R} \sum_{k \in J} Y_{kr}^* \sum_{i \in S_r} W_{ikr}.$$

From the above relations it follows that:

$$Z_{\text{QIP}} = \bar{Z}^* \bar{P} + \bar{X}^* \bar{V} + \bar{Y}^* \bar{U} + \bar{X}^* \bar{Y}^* \bar{W}$$

$$\geq \bar{Z}^* \bar{P} + \bar{X}^* \bar{V} + \bar{Y}^* \bar{U} + \bar{Y}^* \left\{ \overline{\sum_{i \in S_r} W_{ikr}} \right\}.$$

Since $(\bar{Z}^*, \bar{X}^*, \bar{Y}^*)$ is a feasible solution to *Problem YLB*, the following relation follows:

$$\bar{Z}^* \bar{P} + \bar{X}^* \bar{V} + \bar{Y}^* \bar{U} + \bar{Y}^* \left\{ \overline{\sum_{i \in S_r} W_{ikr}} \right\} \geq \bar{Z}^Y \bar{P}$$

$$+ \bar{X}^Y \bar{V} + \bar{Y}^Y \bar{U} + \bar{Y}^Y \left\{ \overline{\sum_{i \in S_r} W_{ikr}} \right\} = Z_{LB}^{Y}.$$

Thus $Z_{LB}^{Y} \leq Z_{\text{QIP}}$.

It is easy to see that this bounding method will provide a tight lower bound whenever

$$\sum_{i \in S_r} \max_{j \in J} \{ W_{ijkr} \} \ll U_{kr} \qquad r \in R,\ k \in J,$$

i.e., the marginal cost contribution of $\sum X_{ij}Y_{kr}W_{ijkr}$ is small when compared to the other cost factors of the problem.

By applying similar arguments to the X_{ij} variables in *Problem QIP*, a second bound is generated by the solution to the following problem.

Problem XLB

Find binary variables Z_j, X_{ij}, Y_{kr} that satisfy:

$$Z_{\mathrm{LB}}^{X} = \min \left\{ \sum_{j\in J} Z_jP_j + \sum_{i\in I}\sum_{j\in J} X_{ij}\tilde{\tilde{V}}_{ij} + \sum_{r\in R}\sum_{k\in J} Y_{kr}U_{kr} \right\},$$

where

$$\tilde{\tilde{V}}_{ij} = V_{ij} + \sum_{r\in T_i} \min_{k\in J}\{W_{ijkr}\},$$

and T_i is the index set of the reports that use data from data source i.

The lower bound value is given by the maximum of those two bounds:

$$Z_{\mathrm{LB}} = \max\{Z_{\mathrm{LB}}^{X};\ Z_{\mathrm{LB}}^{Y}\}.$$

The relationship between Z_{LB} and $L(\hat{\lambda}, \hat{\beta})$ is established in the following theorem.

Theorem 2: $Z_{\mathrm{LB}} \le L(\hat{\lambda}, \hat{\beta})$.

Proof: First we prove that $Z_{\mathrm{LB}}^{Y} \le L(\hat{\lambda}, \hat{\beta})$. The feasible sets for *Problem LR* and *Problem YLB* are identical. Therefore, it suffices to show that the objective function of *Problem YLB* is a special case for *Problem LR*. The objective function for *Problem LR* can be written as:

$$\Bigg[\sum_{j\in J} Z_jP_j + \sum_{i\in I}\sum_{j\in J} X_{ij}V_{ij} + \sum_{r\in R}\sum_{k\in J} Y_{hr}\Big(U_{hr} + \sum_{i\in S_r}\sum_{j\in J}\beta_{ijhr}\Big)$$
$$+ \sum_{r\in R}\sum_{i\in S_r}\lambda_{ir} + \sum_{i\in I}\sum_{j\in J} X_{ij}\left(\sum_{k\in J}\sum_{r\in R}\mathrm{Min}\{0;\ W_{ijkr} - \beta_{ijkr} - \lambda_{ir}\}\right)\Bigg].$$

When the multipliers are set to:

$$\lambda_{ir} = 0 \quad \forall r\in R,\ i\in S_r$$

and

$$\beta_{ijkr} = \begin{cases} 0 & \forall j \ne j_i,\ i\in S_R,\ r\in R,\ k\in J \\ W_{ikr} & \forall j = j_i,\ i\in S_r,\ r\in R,\ k\in J, \end{cases}$$

j_i is an index of j that satisfies $W_{ij_ikr} = W_{ikr}$. Under those conditions $\mathrm{Min}\{0;\ W_{ijkr} - \beta_{ijkr} - \lambda_{ir}\} = 0$ and the objective function of the Lagrangian is reduced to

$$\left[\sum_{j\in J} Z_j P_j + \sum_{i\in I}\sum_{i\in J} X_{ij} V_{ij} + \sum_{r\in R}\sum_{k\in J} Y_{kr}\, \tilde{\tilde{U}}_{kr}\right],$$

which is the objective function to *Problem YLB*. Thus

$$Z^Y_{\mathrm{LB}} = L(0, \bar{W}_{ikr}) \le L(\hat{\lambda}, \hat{\beta}).$$

Using similar arguments it can be shown that $Z^X_{\mathrm{LB}} \le L(\hat{\lambda}, \hat{\beta})$, leading to $Z_{\mathrm{LB}} \le L(\hat{\lambda}, \hat{\beta})$ □

VI. HEURISTICS FOR CONFIGURING DISTRIBUTED COMPUTER SYSTEMS

The problem of designing an optimal configuration of a distributed computing system belongs to the class of NP complete problems. As such it is unlikely that an algorithm will be developed that is capable of solving every instance of the problem to optimality. In lieu of this evidence, the only avenue that is open to a systems designer is to rely on heuristic procedures. Those are expected to generate good but not necessarily optimal solutions for the problem. In this section we develop such heuristics. Providing a theoretical bound on the expected quality of the solutions generated by those heuristics is itself a hard problem. Therefore, the performance of those heuristics is investigated in a set of computational experiments in which the heuristics are applied to a large number of problems and their performance is compared to the lower bound values that were generated by using the procedures developed in Section V. The sensitivity of those procedures to various design parameters is evaluated and can help in identifying the conditions under which those procedures can be expected to generate good solutions.

The heuristics developed in this section are of the greedy type. They start from an initial solution and attempt to improve it by reassigning data sources to database locations or reports to report generation locations. The solution improvement steps are based on the following observations.

Assuming that the decisions regarding the assignment of reports to report generation locations have been made, i.e., a vector $\{Y_{kr}\}$ of zero-one variables has been selected, the optimization problem, *Problem QIP*, is reduced to:

Problem YIP

$$Z_{YIP} = \min\left\{\sum_{j\in J_y} Z_j P_j + \sum_{i\in I}\sum_{j\in J} X_{ij}\, \ddot{V}_{ij}\right\}$$

subject to:

$$\sum_{j\in J} X_{ij} = 1 \qquad i \in I,$$

$$X_{ij} \le Z_j \qquad i \in I, j \in \ddot{J}_y,$$

$$X_{ij}, Z_j = 0 \text{ or } 1 \qquad i \in I, j \in J,$$

where

$$J_y = \{j \mid Y_{jr} = 1 \text{ and } r \in R, j \in J\}$$

$$\ddot{J}_y = J - J_y$$

$$T_i = \{r \mid i \in S_r \text{ and } r \in R\}$$

$$\ddot{V}_{ij} = V_{ij} + \sum_{r\in T_i} W_{ijk_rr},$$

and where k_r is the index of the location in which report r is generated (i.e., $Y_{k_rr} = 1$).

Thus, when the assignment of reports to report generation locations is fixed, the problem is reduced to a simple plant location problem. Using similar arguments, one can show that, when the assignment of data sources to potential database locations is given, i.e., an $\vec{X}_{ij}$ vector has been selected, *Problem QIP* is reduced to:

Problem XIP

$$Z_{XIP} = \min \left\{ \sum_{j\in J_x} Z_j P_j + \sum_{r\in R} \sum_{j\in J} Y_{jr} \ddot{U}_{jr} \right\}$$

subject to:

$$\sum_{j\in J} Y_{jr} = 1 \qquad r \in R,$$

$$Y_{jr} \le Z_j \qquad r \in R, j \in \ddot{J}_x,$$

$$Y_{jr}, Z_j = 0 \text{ or } 1 \qquad r \in R, j \in J,$$

where:

$$J_x = \{j \mid X_{ij} = 1 \text{ and } i \in I, j \in J\}$$

$$\ddot{J}_x = J - J_x$$

$$\ddot{U}_{jr} = U_{jr} + \sum_{i\in S_r} W_{ij_ikr},$$

and where j_i is the index of the location to which the data originating from source i has been assigned (i.e., $X_{ij_i} = 1$).

This problem is also a plant location problem.

Problems XIP and *YIP* are the basis for the development of the following class of heuristics to the distributed system configuration problem. The heuristic consists of the iterative application of those two subproblems in order to improve an initial feasible solution to the problem. A detailed description of the heuristic for configuring a distributed computing system follows:

Step 1—Using some selection criteria, select an initial assignment of reports to report generation locations, i.e., form a $\vec{Y}_{kr}$ vector.

Step 2—Improving the assignment of data sources to database locations:

a. Using the vector $\vec{Y}_{kr}$ as an initial assignment of reports to report generation locations, solve *Problem YIP*.
b. From the solution to *Problem YIP* form an $\vec{X}_{ij}$ vector.

Step 3—Improving the assignment of reports to report generation locations:

a. Using the vector $\vec{X}_{ij}$, which was generated in Step 2b of the algorithm, solve *Problem XIP*.
b. From the solution to *Problem XIP* form a $\vec{Y}_{kr}$ vector.

Step 4—If in a full pass through Steps 2 and 3 of the algorithm, there was no improvement in the objective function value, Stop. Otherwise, go to Step 2.

In each step of the algorithm an attempt is made to improve the assignment of data sources to database locations or the assignment of reports to report generation locations. The process terminates when, in a full pass through the algorithm, there was no further improvement in the solution. Various methods can be used in Step 1 of the algorithm for generating an initial assignment of reports to report generation locations. Different initial assignments could lead to different final solutions by the heuristic. Some of the methods that have been considered are listed:

1. A single location is selected for generating all the reports. This could be the location that minimizes the system costs and involves the evaluation of up to $|J|$ different assignments of data sources to database locations. Each one of those evaluations requires the solution of a plant location problem and might be computationally expensive. Another possibility is to select the starting location as the center of gravity in an approximate gravity model, or to pick the starting node at random.

2. For each report r, $r \in R$, the location that minimizes the processing and distribution costs for that report is identified. The union of those locations is selected as a starting solution.

3. The simple procedure that was developed for generating a lower bound to *Problem ILP* provides as a byproduct a feasible assignment of reports to locations, or of users to database locations. Those assignments can be used as input to Step 1 of the heuristic.

A second set of heuristics for the problem is obtained when the role of the $\tilde{Y}_{kr}$ and the $\tilde{X}_{ij}$ variables is reversed in the suggested heuristic. The correct combination of starting procedures and heuristics can be decided based on their empirical performance in a set of computational experiments.

VII. NUMERICAL EXAMPLES AND COMPUTATIONAL EXPERIMENTS

The methods that were developed in the previous sections provide a system designer with lower and upper bounds on the value of the optimal solution to the system configuration problem, and with a set of possible designs for such a system. To investigate the quality of the solutions that are generated by those procedures, they were programmed and tested on a set of simulated cases. In this section we present an example of using those procedures, and conduct a set of computational experiments in which the performance of those procedures is tested over a wide range of parameter values. These examples demonstrate the value of the design method and highlight the cases in which the procedure is expected to perform very well. First we present the data generation method for the base case. Then, by varying different parameters, we demonstrate how different cost factors influence the selection of an appropriate architecture for the system. By comparing the cost of the computed solution to the cost of a centralized solution, it is possible to identify the cases in which the distributed configuration clearly dominates the solution offered by a centralized system.

7.1 Data generation method for the base case

The base case attempts to simulate the flow of data and reports in a large organization that is distributed over a large geographic area. The organization consists of a set of points representing manufacturing, distribution, and customer services centers. Those points originate the transactions needed for the management and control activities. The points are grouped into regions; each region has a regional center responsible for managing the entities that belong to its region. The organization has a general headquarters that receives reports and data from the organizational entities and is responsible for managing the organization.

The data generation method consists of the following steps. A rectangle of dimensions 1000 by 2500 was divided into n by m regions of equal dimensions. In each region, the coordinates of p points were generated. Those points serve as data sources and can also be used for placement of databases as well as report generation processes. The geographical center of each one of those regions was set as the regional center location for that region. The geographical center of the rectangle

Table I—Test data for the computational example

Point i	Coordinates X_i	Y_i	Transactions A_i	Entities E_i
1	416	250	20543888	4430081
2	811	421	27995668	4661639
3	129	455	13852528	5601720
4	252	66	23314600	4835491
5	416	750	17652664	5938730
6	129	681	20787928	5319019
7	96	882	26454072	4510396
8	132	515	3892134	4835392
9	1249	250	22320524	5149213
10	1087	182	23819312	5394341
11	1579	276	27018756	5172949
12	1501	323	9757924	4795006
13	1249	750	3260475	5874032
14	1155	922	27545052	5244782
15	1364	847	15831062	5272542
16	1490	600	22183256	4646214
17	2082	250	2963550	4512458
18	2488	76	3077690	4119027
19	2417	100	24784832	4996060
20	1742	239	1123881	5207785
21	2082	750	2104029	5316656
22	2116	694	28796560	4985719
23	1891	989	10662968	5162839
24	1784	849	23130012	4246616
25	1250	500	4496444	4408979

was set as the location in which the general headquarters resides. In the numerical example that follows the parameters were set to $n = 2$, $m = 3$, and $p = 4$. Points 1, 5, 9, 13, 17, and 21 are regional centers, while point 25 represents the general headquarters. Regional center 1 controls and receives reports from points 1 to 4, 5 controls points 5 to 8, and so on.

For each point i a tuple (A_i, E_i) was generated, where: $A_i \sim U(10^6; 3 \times 10^7)$ and $E_i \sim U(4 \times 10^6; 6 \times 10^6)$. A_i represents the amount of monthly update activity that originates in data source i, while E_i represents the number of entities in data source i, on which data has to be stored in the system databases. Table I contains the coordinates and the (A_i, E_i) values for the different points.

The set R is partitioned into R_1-daily, R_2-weekly, R_3-monthly, R_4-quarterly, and R_5-yearly subsets of report types. Those reports are generated in a frequency of 365, 52, 12, 4, and one time per year. For each point i a daily and a weekly report are generated (reports 1 to 50 in Table II). A copy of the weekly report is also distributed to the regional center that controls point i. For each regional center weekly, monthly, quarterly, and yearly reports are generated. Those are reports 51 to 74 in Table II. Monthly, quarterly, and yearly reports are generated for the general headquarters. Those are reports 75, 76, and

Table II—List of reports, their data sources, and end user locations

Report ID r	Generation Frequency	End User Locations	Set of Data Sources for Report r – (S_r)
01–25	Daily	r	r
26–50	Weekly	$r-25, RC_r$	$r-25$
51–56	Weekly	$(r-51)\cdot 4+1$	$(r-51)\cdot 4+1, \cdots, (r-51)\cdot 4+4$
57–62	Monthly	$(r-57)\cdot 4+1$	$(r-57)\cdot 4+1, \cdots, (r-57)\cdot 4+4$
63–68	Quarterly	$(r-63)\cdot 4+1$	$(r-63)\cdot 4+1, \cdots, (r-63)\cdot 4+4$
69–74	Yearly	$(r-69)\cdot 4+1$	$(r-69)\cdot 4+1, \cdots, (r-69)\cdot 4+4$
75	Monthly	25	1–25
76	Quarterly	25	1–25
77	Yearly	25	1–25

($RC_{26} = 1, RC_{30} = 5, RC_{34} = 9, RC_{38} = 13, RC_{42} = 17, RC_{46} = 21, RC_{50} = 25$)

77 in Table II. Table II lists for each report type the set of data sources that provide data for that report, and the list of end user locations to which the report has to be distributed.

The cost entries for the numerical example are based on the yearly operations of the system, and the objective is to minimize the yearly costs of operating it. The cost factors are setup and computer operation costs:

$$P_j = vw500{,}000 \quad j \in J,$$

data collection, transfer, update, and storage costs:

$$V_{ij} = (E_i Q_i C_i + A_i C_2 + A_i \ell_i d_{ij} C_3 V_i)\cdot v,$$

where:

$Q_i = 500$ bytes

$C_1 = 10^{-4}$ \$/ebyte

$C_2 = 10^{-3}$ \$/etransaction

$C_3 - 5 \times 10^{-5}$ \$/ebyte

$\ell_i = 120$ bytes

$V_i = 12$

$v = 0.0005$

and

$$d_{ij} = \max\{100; \text{10 times the distance between points } i \text{ and } j\}.$$

The report generation and distribution costs are:

$$U_{kr} = \left(n_r C_{4r} \lambda_r \log_2 \lambda_r + C_{5r} \lambda_r n_r L_r + C_{4r} e_r \log_2 e_r + \sum_{k\in D_r} d_{kr} \Phi_r n_r C_3\right)\cdot u,$$

where:

$$\lambda_r = \sum_{i \in S_r} A_i V_i / n_r,$$

$$e_r = 0.25 \sum_{i \in S_r} E_i,$$

$$L_r = 100,$$

D_r = the set of locations that receive the output of report r,

n_r = the number of times per year that report r is generated,

Φ_r = the amount of output generated by report r (in bytes).

Φ_r is given by:

$$\Phi_r = \lambda_r L_{ir} + 4 e_r L_{2r}$$

$$L_{1r} = \begin{cases} 60 & r \in R_1 \\ 50 & r \in R - R_1 \end{cases}$$

$$L_{2r} = \begin{cases} 200 & r \in R_1 \\ 250 & r \in R_2 \\ 300 & r \in R_3 \\ 500 & r \in R_4 \text{ or } r \in R_5 \end{cases}$$

$$C_{4r} = \begin{cases} 10^{-6} \text{ when } r \text{ is a report that goes to a regional} \\ \quad \text{center or the headquarters} \\ 2 \times 10^{-6} \text{ otherwise} \end{cases}$$

$$C_{5r} = 3 \times C_{4r}$$

$$\mu = 0.0003.$$

Data retrieval and transfer costs are given by:

$$W_{ijkr} = [C_{4r} g_{ir} (A_i V_i / n_r) \log_2 (A_i V_i / n_r) + C_{4r} h_{ir} E_i \log_2 E_i + (C_1 s_1 g_{ir} A_i V_i / n_r + C_1 s_2 h_{ir} E_i) d_{jk}] \cdot w$$

where:

$g_{ir} = 100$, $h_{ir} = 250$, $s_1 = 0.1$, $s_2 = 0.25$ and $w = 0.0015$.

s_1 and s_2 are the appropriate compression factors between the inputs and outputs from the databases.

The heuristic and the lower bounding procedures have been applied to the base case. The heuristic procedure was initiated with a solution in which all the databases were assigned to point 25 (the headquarters). After five iterations the heuristic stopped with a solution having a yearly cost of approximately vw9,090,000. Seven computers were assigned to the system; all of them are used for database assignment and

Table III—Assignment of data sources to database locations (for base case)

Database Locations	Data Sources Assigned to those Databases
1	1, 2, 3, 4
5	5, 6, 7, 8
9	9, 10, 11, 12
13	13, 14, 15
17	17, 18, 19, 20
21	21, 22, 23, 24
25	16, 25

for report generation processes. The 9 million dollars consist of 3.5 million dollars for computer setup and operational costs, *vw*1,655,200 for data collection transfer update and storage costs, *vw*3,509,350 for report generation and distribution costs, and *vw*426,340 for data retrieval and transfer costs. Tables III and IV present the assignment of data sources and report generation processes to computer locations.

The simple lower bounding procedure when applied to the base case has generated a lower bound value of *vw*8,662,100, assuring that the gap to optimality is at most *vw*422,000. The cost of a centralized solution in which only one computer is assigned to point 25 and all the databases and report generation processes are assigned to it is *vw*21,022,350, demonstrating that a distributed architecture is clearly preferable for the base case.

7.2 The impact of cost changes on system configuration

One of the important aspects of havińg a model for configuring distributed computing systems is the ability to perform sensitivity analysis in which the impact of changes in different parameters that influence such a design are investigated. Many cost factors and activity parameters influence the final design of a computing system. Many of those parameters have to be estimated long before the system is actually developed and implemented. Therefore, it is important to be able to investigate well in advance what the impact of changes will be in parameter values and the assumptions used to derive them.

In this section we demonstrate the use of the model to illustrate how the final configuration changes with changes in parameter values. The method used in those investigations was to multiply a parameter by a factor that was varied during the experiments. A factor value of 1 is identical to the base case that was described in the previous section.

In Table V we demonstrate the impact of changes in setup and operational costs on system configuration. The table contains:

Table IV—Assignment of report generation processes to computer locations (for base case)

Report r	Assigned to	Report r	Assigned to	Report r	Assigned to
1	1	27	1	53	9
2	1	28	1	54	13
3	1	29	1	55	17
4	1	30	5	56	21
5	5	31	5	57	1
6	5	32	5	58	5
7	5	33	5	59	9
8	5	34	9	60	13
9	9	35	9	61	17
10	9	36	9	62	21
11	9	37	9	63	1
12	9	38	13	64	5
13	13	39	13	65	9
14	13	40	13	66	13
15	13	41	13	67	17
16	25	42	17	68	21
17	17	43	17	69	1
18	17	44	17	70	5
19	17	45	17	71	9
20	17	46	21	72	13
21	21	47	21	73	17
22	21	48	21	74	21
23	21	49	21	75	25
24	21	50	25	76	25
25	25	51	1	77	25
26	1	52	5		

Table V—The impact of changes in setup costs (P) on system configuration

	System Costs (in Thousands)					Computer Locations		
Factor	$\Sigma P_j Z_j$	$\Sigma V_{ij} X_{ij}$	$\Sigma U_{kr} Y_{kr}$	$\Sigma W X_{ij} Y_{kr}$	Total	Data-bases	Report Gener-ation	Total
32	16000	5173.54	15346.17	2.64	36522	1	1	1
16	8000	5173.54	15346.17	2.64	28522	1	1	1
8	8000	3640.56	10847.64	246.36	22734	2	2	2
4	6000	2678.10	7264.51	409.21	16351	3	3	3
2	5000	2066.68	4886.43	358.52	12311	5	5	5
1	3500	1655.20	3509.35	426.34	9090	7	7	7
1/2	1750	1655.20	3509.35	426.34	7340	7	7	7
1/4	1125	1283.62	3509.35	483.64	6401	9	7	9
1/8	1000	331.98	3509.35	741.65	5582	16	7	16
1/16	561.6	147.83	3509.35	809.75	5028	18	7	18
1/32	343.2	22.4	3509.35	828.12	4703	22	7	22

- The total cost of the solution generated by the heuristic and the breakdown of the total cost into its major components
- The total number of computer locations used by that solution, how many of them are used for database placement, and how many for report generation processes.

Table VI—The impact of changes in setup costs (P) on the heuristic, lower bound, and centralized solution values

Factor	Lower Bound on Optimal Solution	Cost of Heuristic Solution	Cost of Lower Bound Policy	Cost of Best Heuristic	Cost of Centralized Solution
32	36517.0	36522.35	36522.35	36522.34	36522.35
16	28517.0	28522.35	28522.35	28522.35	28522.35
8	21939.8	22734.57	22351.82	22351.82	24522.35
4	15939.8	16351.82	16351.82	16351.82	22522.35
2	11950.5	12311.63	12311.64	12311.63	21522.35
1	8662.1	9090.90	9090.90	9090.90	21022.35
1/2	6912.1	7340.90	7340.90	7340.90	20772.35
1/4	5689.1	6401.62	6425.32	6401.62	20647.35
1/8	4776.2	5582.98	5597.27	5582.98	20584.85
1/16	4196.7	5028.54	5042.66	5028.54	20553.60
1/32	3868.9	4703.07	4708.95	4703.07	20537.95

Increases in setup costs lead to solutions that are closer to a centralized configuration. As the cost of computing is reduced we observe a decrease in system costs and a move towards a distributed solution. In Table VI we compare, for each one of the above cases, the values of two heuristics, one which is identical to the heuristic described in the previous section. The second one uses the solution generated by the simple lower bounding procedure, as a starting solution for the heuristic. The value generated by the best of those two heuristics is compared to the value of a centralized solution and to the lower bound value. The gap between a centralized solution and a distributed solution increases as computing costs decrease. It is also interesting to note that the gap between the feasible solution and the lower bound is small and is nonsignificant when compared to the gap to the centralized solution.

In Tables VII and VIII a similar analysis is made of the impact of changes in data collection, transfer, and update costs (V) on system configuration. We observe a trend towards a distributed architecture as those costs increase. The number of database locations increases with the increase in data collection costs. Setting the data collection and transfer costs to zero reduces the number of database locations opened in the system. However, this number is not reduced to one. The reason for this is the fact that report generation and distribution costs have not been reduced and require more than one database in the system in order not to increase further the report generation and distribution costs. In this case, too, the gap between the best heuristic and the lower bound is nonsignificant, assuring that the feasible solutions are very close to the optimal ones.

Tables IX and X summarize the results of changing the report generation and distribution costs (U) on system configuration. Here

Table VII—The impact of changes in data collection/transfer and update costs (V) on system configuration

	System Costs (in Thousands)					Computer Locations		
Factor	$\Sigma P_j Z_j$	$\Sigma V_{ij} X_{ij}$	$\Sigma U_{kr} Y_{kr}$	$\Sigma W X_{ij} Y_{kr}$	Total	Databases	Report Generation	Total
32	11000	604.80	3509.35	836.95	15951	22	7	22
16	10500	555.27	3509.35	842.35	15406	21	7	21
8	9000	1154.71	3509.35	818.58	14482	18	7	18
4	6000	2729.13	3509.35	733.68	12972	12	7	12
2	3500	3310.41	3509.35	426.34	10746	7	7	7
1	3500	1655.20	3509.35	426.34	9090	7	7	7
1/2	3500	827.60	3509.35	426.34	8263	7	7	7
1/4	3500	413.80	3509.35	426.34	7849	7	7	7
1/8	3500	580.09	3763.36	507.76	8351	4	7	7
1/16	3500	323.07	3800.57	534.65	8158	2	7	7
1/32	3500	161.53	3800.57	534.65	7996	2	7	7

Table VIII—The impact of changes in the data collection/transfer and update costs (V) on the heuristic, lower bound, and centralized solution values

Factor	Lower Bound on Optimal Solution	Cost of Heuristic Solution	Cost of Lower Bound Policy	Cost of Best Heuristic	Cost of Centralized Solution
32	15112.0	15951.10	15951.10	15951.1	181402.25
16	14562.4	15406.98	15406.98	15406.98	98625.52
8	13661.9	14482.65	14482.65	14482.65	57237.16
4	12236.4	12972.17	12972.18	12972.17	36542.99
2	10317.5	10746.11	10746.12	10746.11	26195.90
1	8662.1	9090.90	9090.90	9090.90	21022.35
1/2	7834.6	8263.30	8263.30	8263.30	18435.58
1/4	7420.8	7849.50	7849.50	7849.50	17142.20
1/8	7214.0	8351.22	7642.60	7642.60	16495.50
1/16	7110.5	8158.29	7539.15	7539.15	16172.15
1/32	7058.9	7996.75	7487.42	7487.42	16010.48

again we observe a reduction in the number of databases and report generation locations as the value of U is reduced. Reducing the report generation and distribution costs to a negligible level still requires two computer locations for an efficient solution. The reason for it is the fact that the data collection transfer and storage costs have not been changed. They are still significant in the total system costs and can be reduced only by having more than one database location in the system. The gap between a distributed and centralized solution increases as the value of U is increased. Here again we observe a nonsignificant gap between the heuristic and the simple lower bounding procedure.

Tables XI and XII summarize the impact of changes in interprocess communication costs W on system configuration. A decrease in W

Table IX—The impact of changes in report generation and distribution costs (U) on system configuration

	System Costs (in Thousands)					Computer Locations		
Factor	$\Sigma P_j Z_j$	$\Sigma V_{ij} X_{ij}$	$\Sigma U_{kr} Y_{kr}$	$\Sigma W X_{ij} Y_{kr}$	Total	Data-bases	Report Gener-ation	Total
32	3500	1655.20	112299.27	426.34	117880	7	7	7
16	3500	1655.20	56149.63	426.34	61731	7	7	7
8	3500	1655.20	28074.81	426.34	33656	7	7	7
4	3500	1655.20	14037.40	426.34	19618	7	7	7
2	3500	1655.20	7018.70	426.34	12600	7	7	7
1	3500	1655.20	3509.35	426.34	9090	7	7	7
1/2	2500	2066.68	2443.21	358.52	7368	5	5	5
1/4	1500	2678.10	1816.12	409.21	6403	3	3	3
1/8	1000	3640.56	1355.95	246.36	6242	2	2	2
1/16	1000	3640.56	677.97	246.36	5564	2	2	2
1/32	1000	3640.56	338.98	246.36	5225	2	2	2

Table X—The impact of changes in the report generation and distribution costs (U) on the heuristic, lower bound, and centralized solution values

Factor	Lower Bound On Optimal Solution	Cost of Heuristic Solution	Cost of Lower Bound Policy	Cost of Best Heuristic	Cost of Centralized Solution
32	117452.2	117880.8	117880.8	117880.8	496753.5
16	61302.9	61731.2	61731.2	61731.2	251214.9
8	33227.7	33656.3	33656.3	33656.3	128445.5
4	19190.3	19618.9	19618.9	19618.9	67060.8
2	12171.6	12600.2	12600.2	12600.2	36368.5
1	8662.1	9090.9	9090.9	9090.9	21022.3
1/2	6851.7	7368.4	7284.3	7284.3	13349.2
1/4	5766.7	6403.4	6199.8	6199.8	9512.7
1/8	5042.4	6242.8	5409.6	5409.6	7594.4
1/16	4596.7	5564.9	5340.0	5340.0	6635.3
1/32	4308.8	5225.9	4832.6	4832.6	6155.7

increases the incentive to move towards a distributed solution. As W increases the gap between the heuristic and the centralized solution is decreased. However, the number of computer locations is not reduced to one, mainly due to the nonnegligible data collection and report distribution costs. When interprocess communication costs are significant, we also observe a significant gap between the lower bound value and the heuristic solution value. This might be one of the cases in which it is worthwhile to apply the Lagrangian-based lower bounding procedure.

In the last set of experiments we investigate the impact of uniform changes in communication costs on system configuration. The method by which those changes were entered into the data was to multiply the distance measures $\{d_{ij}\}$, which were defined in the previous section, by

Table XI—The impact of data retrieval and transfer costs (W) on system configuration

	System Costs (in Thousands)					Computer Locations		
Factor	$\Sigma P_j Z_j$	$\Sigma V_{ij} X_{ij}$	$\Sigma U_{kr} Y_{kr}$	$\Sigma W X_{ij} Y_{kr}$	Total	Data-bases	Report Gener-ation	Total
32	2500	5173.54	9258.75	1908.21	18840	1	5	5
16	2500	5173.54	6222.17	3032.75	16298	1	5	5
8	3000	5173.54	5479.41	1694.65	15437	1	6	6
4	3000	5173.54	4848.00	1281.55	14303	1	6	6
2	3500	1655.20	3509.35	852.68	9517	7	7	7
1	3500	1655.20	3509.35	426.34	9090	7	7	7
1/2	3500	1655.20	3509.35	213.17	8877	7	7	7
1/4	3500	1655.20	3509.35	106.58	8771	7	7	7
1/8	3500	1655.20	3509.35	53.29	8717	7	7	7
1/16	3500	1655.20	3509.35	26.64	8691	7	7	7
1/32	3500	1655.20	3509.35	13.32	8677	7	7	7

Table XII—The impact of changes in data retrieval and transfer costs (W) on the heuristic, lower bound, and centralized solution values

Factor	Lower Bound on Optimal Solution	Cost of Heu-ristic Solu-tion	Cost of Lower Bound Policy	Cost of Best Heuristic	Cost of Cen-tralized Solu-tion
32	8744.7	18840.51	22307.59	18840.51	21104.26
16	8701.7	16928.46	15486.08	15486.08	21061.98
8	8680.7	15347.61	12075.32	12075.32	21040.85
4	8670.9	14303.10	10369.94	10369.94	21030.28
2	8664.9	9517.25	9517.25	9517.25	21024.99
1	8662.1	9090.90	9090.90	9090.90	21022.35
1/2	8660.8	8877.73	8877.73	8877.73	21021.03
1/4	8660.4	8771.14	8771.15	871.14	21020.37
1/8	8660.0	8717.85	8717.85	8717.85	21020.04
1/16	8659.9	8691.20	8691.21	8691.20	21019.87
1/32	8659.6	8677.88	8677.88	8677.88	21019.79

a factor that was changed during the experiments. A change in distances implies similar changes in

- Data collection transfer and storage costs
- Interprocess communication costs
- Report generation and distribution costs.

The results of those experiments are summarized in Tables XIII and XIV. As expected, a uniform reduction in communication costs leads to a centralized solution, while an increase in communication costs leads towards a distributed solution. The gap between a centralized and the distributed solution increases as communication costs increase. A similar trend is observed between the lower bound and the heuristic solution.

Table XIII—The impact of uniform changes in communication distances (D) on system configuration

	System Costs (in Thousands)					Computer Locations		
Factor	$\Sigma P_j Z_j$	$\Sigma V_{ij} X_{ij}$	$\Sigma U_{kr} Y_{kr}$	$\Sigma W X_{ij} Y_{kr}$	Total	Data-bases	Report Gener-ation	Total
32	11000	494.4	111923.3	26480.6	149898.9	22	7	22
16	9000	2276.1	55967.7	12944.1	80187.9	18	7	18
8	8000	2619.4	27989.9	5926.1	44535.5	16	7	16
4	5000	4364.7	14000.9	2156.9	25522.6	10	7	10
2	3500	3309.3	7006.5	850.7	14666.6	7	7	7
1	3500	1655.2	3509.3	426.3	9090.9	7	7	7
1/2	2500	1033.6	2447.5	180.3	6161.5	5	5	5
1/4	1500	535.8	1457.7	83.5	3577.1	3	3	3
1/8	500	517.1	1536.5	2.6	2556.3	1	1	1
1/16	500	7.3	24.8	2.6	534.7	1	1	1
1/32	500	7.3	24.8	2.6	534.7	1	1	1

Table XIV—The impact of uniform changes in communication distances (D) on the heuristic, lower bound, and centralized solution values

Factor	Lower Bound on Optimal Solution	Cost of Heuristic Solution	Cost of Lower Bound Policy	Cost of Best Heuristic	Cost of Centralized Solution
32	123303.1	149898.9	150070.3	149898.9	657069.9
16	66915.5	80187.9	80387.2	80187.9	328787.4
8	38099.9	44535.5	44646.2	44535.5	164646.1
4	22717.2	25522.6	25651.4	25522.6	82575.2
2	13813.0	14666.6	14666.6	14666.6	41539.9
1	8662.1	9090.9	9090.9	9090.9	21022.35
1/2	5978.4	6161.5	6161.5	6161.5	10763.4
1/4	3491.1	3577.1	3577.1	3577.1	4608.3
1/8	2450.2	2556.3	2479.4	2479.4	2556.3
1/16	529.8	534.7	534.7	534.7	534.7
1/32	529.8	534.7	534.7	534.7	534.7

VIII. POTENTIAL APPLICATIONS AND EXTENSIONS OF THE MODEL

The optimization models for configuring distributed computer systems, which were formulated in the previous sections, were based on simplifying assumptions. Those assumptions restrict the use of the model to a subset of the organizations in which we expect that distributed computing systems will be widely implemented. Some of those restrictive assumptions can be relaxed, and the model can be extended to a wider range of application areas. In this section we present a few of those extensions and provide examples to existing or planned systems that are prime candidates for using this model. Each of these systems has unique characteristics that are exploited in the models and the suggested solution methods.

8.1 Allowing for different transaction classes

The model developed in Section IV has assumed that all the transactions that originate in a data source are assigned to the same database location. This assignment is regardless of the set of entities and functions that are being served by the data contained in those transactions. This is rather a highly restrictive assumption. In many of today's organizations it is difficult to associate a single data source with a unique set of functions that are served by the transactions that originate in that data source. Situations in which the same data source, and sometimes the same individual, provides input data to many databases, and to a variety of sometimes conflicting functions, are quite common. For example, a salesperson in a company sales office can report orders for final products that were received from customers, which will update a database that supports the order handling subsystem. The same salesperson can also enter data on payments made by customers, fill out presence or absenteeism reports that will update the personnel and employee history database, or can enter customer reports on problems that they have encountered with equipment installed by the company (those transactions will typically update an engineering or technical support database). When larger entities are used as data sources (such as departments, branch offices, plants, or divisions), then the association between data sources and functions becomes less apparent. For example, a job shop in a large manufacturing company originates and feeds transactions that deal with different activities in the organization. Transactions can report on the progress of jobs in the production process, the consumption of inventory items in those jobs, transfer of final goods to inventory, machine and equipment failure, presence or absence of employees. Those transactions will be routed to different databases and will be used by completely different sets of end users in the organization. The events reported by those transactions occur at different rates, and they are of interest to completely different sets of end users that might be located in different regions of the country, where each one of those groups might have their own reporting requirements.

The restriction that all the transactions that originate in a single location will be routed to the same database location, can, in many instances, increase the data processing costs to the organization that is being served by the data processing department and, at the same time, lengthen the response times to user requests for reports. Organizations that have such characteristics can benefit from relaxing this restriction. It might be possible to reduce the system's data processing costs and improve its response time by routing transactions that serve identical functions to database locations that are different from those that serve other functions even if the transactions originate from the same data source. Relaxing this assumption slightly complicates the routing of transactions in the network, but is justified by the potential reduction in the system costs.

Defining by F the index set of the various functions in the organization, it is possible to partition the set of transactions that originate in source i, $i \in I$, into up to $|F|$ subsets of transactions, where each subset corresponds to a specific function in the organization. Letting i_f denote the subset of transactions in source i that are used by function f, $f \in F$ in the organization, this problem can be formulated as follows.

Find binary variables Z_j, X_{mj}, Y_{kr} that satisfy

$$Z_{FIQP} = \min \left\{ \sum_{j \in J} Z_j P_j + \sum_{j \in J} \sum_{f \in F} \sum X_{mj} V_{mj} + \sum_{r \in R} \sum_{k \in J} Y_{kr} U_{kr} \right. \\ \left. + \sum_{r \in R} \sum_{k \in J} \sum_{j \in J} \sum_{f \in F} \sum_{m \in i_f} X_{mj} Y_{kr} W_{mjkr} \right\}$$

subject to eqs. (3) through (5) and

$$\sum_{j \in J} X_{mj} = 1 \quad m \in i_f,\ f \in F,$$

$$X_{mj} \leq Z_j \quad m \in i_f,\ f \in F,\ j \in J,$$

where V_{mj} and W_{mjkr} are the appropriate cost factors for those data sources and functions. This optimization problem has the same characteristics as the single-function/multiple-sources system configuration problem. The only difference between the two problems is in the increase in the number of data sources introduced by the splitting of single data sources into multiple data sources. The same methods that have been developed for solving *Problem ILP* are applicable to the above optimization problem.

8.2 Different setup costs for report generation than for data storage

The hardware and software requirements of a node in the distributed system could be dependent on the activities performed in the node. For example, different setup costs might be incurred for supporting a database versus supporting a report generation process, or for supporting both activities.

Letting P_{jt}, $t = \{1, 2, 3\}$ be the setup costs for placing in location j a computer that can support a: database $\{1\}$, report generation process $\{2\}$, or both activities $\{3\}$. We redefine the Z_j variables in *Problem QIP* and *Problem ILP* as the sum of three variables, Z_{jt}, $t = \{1, 2, 3\}$. Z_{jt} are binary variables that specify what type of capabilities will be provided in location j.

This problem has the same formulation as *Problem QIP* or *Problem*

ILP, with the Z_j variables replaced by the sum $\sum_{i=1}^{3} Z_{jt}$ and the addition of the constraints

$$\sum_{i=1}^{3} Z_{jt} \leq 1 \quad j \in J.$$

Techniques similar to the ones used for solving the previous models can be used to generate solutions for this problem.

8.3 Transactions that update multiple databases

A situation that arises in many systems is one where a transaction is used to update two or more functionally different databases that might be located in different locations. For example, a transaction that reports on the transfer of inventory items from one regional warehouse to another warehouse might have to update two fragments of the inventory database that are located in different locations. Transfer of employees from one region to another, transfer of items from inventory to the production process, transfer of final products from the production process to the distribution system, or third party charging of long distance phone calls are other examples in which there might be a need to simultaneously update more than one fragment of a database.

This problem can be handled in a similar fashion to the one used for modeling different transaction classes by splitting the flow of transactions according to the databases that they have to update and introducing a new set of variables and costs associated with the synchronization of simultaneous updates to multiple databases.

8.4 Allowing for multiple copies of databases

An underlying assumption of the model was that there is only a single copy of each fragment of the database in the system. In some cases it is possible to save on retrieval costs by allowing for a partial or full overlap between databases. An example in which such duplication can be useful is in a videotex system in which we can have full or partial overlap between databases. Models that select the fragments to be duplicated, decide which portion of each fragment will be duplicated, and assign each fragment to a computer location are not easy to formulate and will require further research.

IX. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

We have developed models and algorithms for designing an architecture of distributed computer systems. The algorithms have been tested on a few examples and have performed well by generating good initial designs for those systems. Research is now in progress on extending those methods by developing better procedures for computing lower bounds on the optimal solution and improved heuristics for generating feasible solutions for the problem. Despite the restrictive assumptions that have been used for developing the models, we believe that the developed procedures are valuable tools for helping the preliminary design of distributed computing systems.

We have already pointed out the possible extensions of the formulations and the models. Another extension that might be worthwhile to pursue includes multiperiod versions of the problem. The models developed in this paper are single-period versions of the problem, which are well suited for the design of stable systems. In reality some systems go through transitional expansion periods for which multiperiod versions of the problem have to be developed. The models presented here are also uncapacitated versions of the problem. Again, in reality we expect to have capacitated or capacity-dependent setup and operational cost versions of the problem.

X. ACKNOWLEDGMENT

I wish to express my appreciation to Dr. Liebert for raising many of the issues involved in the adoption of a distributed computing system to commercial organizations and for providing helpful comments on earlier versions of the paper; to Dr. Wakid for introducing me to TCAS and to testing, maintenance, and provisioning operations in the telephone system; Dr. Ormsby for handling the editing and typesetting of the paper, Dr. Friedel for many helpful discussions and comments on earlier versions of the paper; and to Mr. Altinkemer for programming the algorithms and performing the reported computational experiments.

REFERENCES

1. C. A. Ellis, "A Robust Algorithm for Updating Duplicate Data Bases," Berkeley Workshop on Distributed Data Management and Computer Networks, University of California at Berkeley (May 1977), pp. 146–58.
2. M. Hammer and D. Shipman, "An Overview of Reliability Mechanisms for a Distributed Data Base System," IEEE Tutorial: Centralized and Distributed Data Base Systems (1979), pp. 645–7.
3. L. Lamport, "Time Clocks and the Ordering of Events in a Distributed System," IEEE Tutorial: Centralized and Distributed Data Base Systems (1979), pp. 588–95.
4. R. J. Ramirez and N. Santoro, "Distributed Control of Updates in Multiple-Copy Data Bases: A Time Optimal Algorithm," Proc., 4th Berkeley Workshop on Distributed Data Management and Computer Networks, University of California at Berkeley, May 1979.
5. P. M. G. Apers, *Query Processing and Data Allocation in Distributed Data Base Systems*, Amsterdam: Free University, 1982.
6. R. M. Shapiro and R. E. Millstein, "Failure Recovery in a Distributed Data Base System," Proc. COMPCOM (Spring 1978), pp. 66–70.
7. R. L. Rivest, A. Shamir, and L. Adelman, "A Method of Obtaining Digital Signatures and Public Key Cryptosystems," Comm. ACM, *21*, No. 2 (February 1978), p. 120.

8. R. G. Casey, "Allocation of Copies of a File in an Information Network," AFIPS 1972 SJCC Conf. Proc. *40* (1972), pp. 617–25.
9. P. Chen and J. Akoka, "Optimal Design of Distributed Information Systems," IEEE Trans. Comput., *C-29,* No. 12 (December 1980), pp. 1068–80.
10. W. W. Chu, "Optimal File Allocation in Multiple Computer Systems," IEEE Trans. Comput., *C-18,* No. 10 (October 1969), pp. 885–9.
11. K. D. Levin and H. L. Morgan, "A Dynamic Optimization Model for Distributed Data Bases," Oper. Res. *26,* No. 5 (September 1978), pp. 824–35.
12. H. L. Morgan and K. D. Levin, "Optimal Program and Data Locations in Computer Networks," Comm. ACM, *20,* No. 5 (May 1977), pp. 315–21.
13. B. Gavish and H. Pirkul, "Allocation of Data Bases and Processors in a Distributed Computing System," *Management of Distributed Data Processing,* J. Akoka, ed., Amsterdam: North-Holland, 1982, pp. 215–31.
14. S. Mahmoud and J. S. Riordan, "Optimal Allocation of Resources in Distributed Information Networks," ACM Trans. Data Base Syst., *1,* No. 1 (March 1976), pp. 66–78.
15. H. Moortgat, "Processor and File Allocation in Distributed Data Base Systems," Ph.D. Dissertation, University of Washington, Seattle, 1979.
16. B. Gavish and A. Segev, "Query Optimization in Distributed Computer Systems," *Management of Distributed Data Processing,* J. Akoka, ed., Amsterdam: North-Holland, 1982, pp. 233–52.
17. B. Gavish and A. Segev, "Set Query Optimization in Horizontally Partitioned Distributed Data Base Systems," working paper, The Graduate School of Management, The University of Rochester, Rochester, N.Y., 1983.
18. A. R. Hevner and S. B. Yao, "Query Processing in Distributed Data Base Systems," IEEE Trans. Software Eng., *SE-5,* No. 3 (March 1979), pp. 177–87.
19. B. Gavish, "Topological Design of Centralized Computer Networks-Formulations and Algorithms," Networks, *12* (April 1982), pp. 355–77.
20. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* New York, NY: W. H. Freeman and Company, 1979.
21. D. Erlenkotter, "A Dual Based Procedure for Uncapacitated Facility Location," Oper. Res., *26,* No. 1 (January 1978), pp. 992–1009.
22. M. L. Fisher, "Lagrangean Relaxation Method for Solving Integer Programming Problems," Manage. Sci., *27* (January 1981), pp. 1–18.
23. M. Held and R. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," Math. Programming, *1* (January 1971), pp. 6–25.
24. B. T. Poljack, "Minimization of Unsmooth Functionals," U.S.S.R. Computational Math. Math. Phys., pp. 14–29. [Translation of "Zurnal Vycislitel 'noi Matematiki i Matematiceskoi Fiziki," *9* (1969), pp. 509–21.]
25. B. Gavish and S. K. Srikanth, "Optimal Solution Methods for Large Scale Multiple Salesman Traveling Salesman Problem," working paper, The Graduate School of Management, The University of Rochester, Rochester, N.Y. (1980).
26. B. Gavish, "Formulations and Algorithms for the Capacitated Minimal Directed Tree Problem," J. ACM, *30,* No. 1 (January 1983), pp. 118–32.

AUTHOR

Bezalel Gavish, B.Sc. (Industrial Management and Engineering), M.Sc. and Ph.D. (Operations Research), Technion-Israel Institute of Technology, Haifa, in 1967, 1970, and 1975, respectively. From 1968 to 1973 he was Department Head of Systems Analysis and Scientific programming in charge of developing large-scale on-line logistics and transportation management computerized systems. From 1973 to 1976 he was Project Head of developing and testing a system for computer-assisted medical diagnosis at the IBM Scientific Center. He joined the Graduate School of Management, University of Rochester, Rochester, NY, in 1976, where he is an Associate Professor in Computers and Information Systems. During that time he was a visiting faculty member at the IBM T. J. Watson Research Center, Bell Laboratories, and the Technion. His research interests span the design and analysis of computer communication networks, design and analysis of distributed computing systems, systems analysis and design, combinatorial optimization, and scheduling and routing in logistic systems. Member, Association for Computing Machinery, the Operations Research Society of America, The Institute of Management Sciences, IEEE Computer Society. Recipient, 1972 IPA Award, 1982 AIIE Transactions Development and Applications Award.