DESIGN ISSUES IN DISTRIBUTED MANAGEMENT INFORMATION SYSTEMS

by

JACOB AKOKA

B.S., Faculte des Sciences-Paris (1970)

S.M., University of Paris VI (1971)

Doctorat 3ème Cycle, University of Paris VI (1975)

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 4, 1978

Signature of Author ..........................................................

Sloan School of Management, May 4, 1978

Certified by ..........................................................

Thesis Supervisor

Accepted by ..........................................................

Chairman, Department Committee
DESIGN ISSUES IN DISTRIBUTED MANAGEMENT INFORMATION SYSTEMS

by

Jacob Akoka

Submitted to the Department of Management
on May 4, 1978 in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

ABSTRACT

Due to the advances in computer network technology and the steadily decreasing cost of hardware, distributed information systems have become a potential alternative to centralized information systems. This thesis analyzes issues related to the design of distributed information systems. Most of the research done in the past can be characterized by a piecemeal approach since it tends to consider the computer network design issue and the distributed data base design issue separately. The critical survey presented in Chapter 2 points out the need to integrate both issues in an overall design approach. In an attempt to incorporate most of the factors that compose distributed information systems, we present a global model in which network topology, communication channels, capacity, size of computer hardware, pricing schemes, and routing disciplines are interrelated in an optimal design. In addition, we show how to derive from the global model a design model for distributed database systems. To solve the two distributed information system models (integer nonlinear programming models), we propose a mathematical programming algorithm that has the following characteristics: - it takes advantage of the special structure of the models; - it uses a specific binary
bounded branch and bound method which limits a priori the number of the nodes of the tree.

Comparing with previous work in this area, the algorithm is more efficient than traditional branch and bound techniques and can avoid the disadvantages of heuristic methods which were widely used in the optimization of computer networks and distributed database systems.

The global model and the solution procedure are used to design the distributed information system for a large bank in Europe. Our solution proved to be better in terms of optimal configuration than the bank's experimental network. In addition, our model for the design of distributed database systems performed better than past models.

An efficient computer code allowing to solve real-life distributed information system problems, with little programming effort for the users, was developed and tested.

An extension of our global model to the issue of centralization versus decentralization of information systems is presented. The information system is divided in three components (systems operations, systems development, and systems management), and a specific submodel is used to facilitate the decision for each component. Finally, further research in the area of design models for distributed information systems is discussed.

Peter P. S. Chen, Assistant Professor
I would like to express my gratitude to my thesis advisor, Professor P.P.S. Chen, for his careful guidance and his valuable comments; his patience and constructive criticism helped me to integrate many concepts within a coherent frame. His encouragement has been invaluable. Thank you so much Peter!

I would also like to thank Professors J.D.C. Little and S.E. Madnick for their insightful comments. Their suggestions contributed to broaden my view of the distributed management information system design problem. Thank you for your help and encouragement.

I am very grateful to Maitre Olivier, General Director of ESSEC (Ecole Superieure des Sciences Economiques et Commerciales) for his moral support and for facilitating my staying at the Sloan School of Management (M.I.T.). I would like to thank the following for the financial aid received in support of this doctoral program: l'Ecole Superieure des Sciences Economiques et Commerciales (ESSEC), La Fondation Nationale Pour l'Enseignement de la Gestion des Entreprises (FNEGE), and l'Institut de Recherche d'Informatique et d'Automatique (IRIA).

Thanks to my colleagues, D. Briolat, H. Dayan, R. Thietart, G. Viens and B. Yon for their help.

I wish to express my personal appreciation to Judith Mason for her excellent typing.

Last but not least, I would like to sincerely thank my wife,
Vivi, and my children Karen and Nathalie, for their encouragement and understanding during all phases of this work.
To my mother
TABLE OF CONTENTS

CHAPTER ONE -- INTRODUCTION
   I. Toward Distributed Information Systems ............... 12
   II. Statement of the Research ............................ 14
   III. Research Methodology .............................. 17
   IV. Outline of Dissertation ............................ 18

CHAPTER TWO -- DISTRIBUTED INFORMATION SYSTEMS: A STATE OF THE ARTS STUDY
   I. Introduction ........................................ 24
   II. Major Arguments Used for Distributed Information Systems 25
      A. Economy of Scale .................................. 25
      B. Sophistication of Applications .................. 29
      C. Quality of Systems Development ................ 30
   III. Computer Network: A Survey ........................ 32
      A. Network Topologies ................................ 32
      B. Communication Systems .......................... 39
      C. Computer Network Design Principles .......... 43
      D. Solutions for the Optimum Design ............ 46
   IV. Distributed Database Systems: An Overview ........... 52
      A. Introduction ...................................... 52
      B. Classification of Distributed Databases .......... 54
      C. Design Models for Distributed Database Systems . 59
   V. Conclusion ........................................... 66

CHAPTER THREE -- DESIGN AND OPTIMIZATION OF DISTRIBUTED INFORMATION SYSTEMS
   I. Introduction .......................................... 76
   II. The Global Model .................................... 81
      A. Assumptions and Modelling Aspects ............... 82
      B. The Objective Function ........................... 87
      C. The Constraints .................................. 95
      D. Final Formulation of the Problems ............. 100
      E. Model Type ....................................... 104
### Chapter Four

**A Mathematical Programming Algorithm for Distributed Information Systems**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Introduction</td>
<td>116</td>
</tr>
<tr>
<td>II. The Algorithm: A Brief Outline</td>
<td>122</td>
</tr>
<tr>
<td>III. Detailed Description of the Algorithm</td>
<td>124</td>
</tr>
<tr>
<td>IV. The Computer Implementation of the Algorithm</td>
<td>132</td>
</tr>
<tr>
<td>V. Conclusion</td>
<td>137</td>
</tr>
</tbody>
</table>

### Chapter Five

**Application of the Design Models for Distributed Information Systems**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Introduction</td>
<td>140</td>
</tr>
<tr>
<td>II. Application of the Global Model for the Design of a Distributed Information System for a Large Bank</td>
<td>141</td>
</tr>
<tr>
<td>A. The Settings</td>
<td>141</td>
</tr>
<tr>
<td>B. The Data Processing Group</td>
<td>142</td>
</tr>
<tr>
<td>C. The Decentralization</td>
<td>144</td>
</tr>
<tr>
<td>D. The Bank's Experimental Network</td>
<td>145</td>
</tr>
<tr>
<td>E. Application of our Model</td>
<td>151</td>
</tr>
<tr>
<td>F. Sensitivity Analysis</td>
<td>168</td>
</tr>
<tr>
<td>III. Optimal Distributed Database Systems: An Application of the Second Model</td>
<td>177</td>
</tr>
<tr>
<td>A. First Case - No Return Flow of Information and no Storage Cost of Database</td>
<td>184</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>B.</td>
<td>Second Case - Existence of Storage Cost for Database</td>
</tr>
<tr>
<td>C.</td>
<td>Third Case - Existence of the Return Flow of Information</td>
</tr>
<tr>
<td>IV.</td>
<td>Conclusion</td>
</tr>
<tr>
<td><strong>CHAPTER SIX</strong></td>
<td>APPLICATION OF THE GLOBAL MODEL TO THE ISSUE OF CENTRALIZATION VERSUS DECENTRALIZATION OF INFORMATION SYSTEMS</td>
</tr>
<tr>
<td>I.</td>
<td>Introduction</td>
</tr>
<tr>
<td>II.</td>
<td>A Critical Survey of the Methods used in the Past to Solve the Issue of Centralization Versus Decentralization of Information Systems</td>
</tr>
<tr>
<td>III.</td>
<td>The Decision Support Model</td>
</tr>
<tr>
<td>A.</td>
<td>Introduction</td>
</tr>
<tr>
<td>B.</td>
<td>System Operations: An Optimization Model</td>
</tr>
<tr>
<td>C.</td>
<td>System Development: A Decision Table</td>
</tr>
<tr>
<td>D.</td>
<td>System Management: Guidelines</td>
</tr>
<tr>
<td>IV.</td>
<td>Conclusion</td>
</tr>
<tr>
<td><strong>CHAPTER SEVEN</strong></td>
<td>SUMMARY, PERSISTING PROBLEMS AND DIRECTIONS FOR FURTHER RESEARCH</td>
</tr>
<tr>
<td>I.</td>
<td>Summary</td>
</tr>
<tr>
<td>A.</td>
<td>The Design and Optimization of Distributed Systems</td>
</tr>
<tr>
<td>B.</td>
<td>The Solution Procedure</td>
</tr>
<tr>
<td>C.</td>
<td>The Computer Code</td>
</tr>
<tr>
<td>D.</td>
<td>Real-Life Applications</td>
</tr>
<tr>
<td>E.</td>
<td>The Issue of Centralization Versus Decentralization of Information Systems</td>
</tr>
<tr>
<td>II.</td>
<td>Persisting Problems in Distributed Information Systems</td>
</tr>
<tr>
<td>III.</td>
<td>Directions for Further Research</td>
</tr>
<tr>
<td><strong>APPENDICES</strong></td>
<td></td>
</tr>
<tr>
<td>A.</td>
<td>Mathematical Description of Step 3 of our Algorithm</td>
</tr>
<tr>
<td>B.</td>
<td>Advantages and Disadvantages of Centralized and Decentralized Information Systems</td>
</tr>
<tr>
<td>C.</td>
<td>Detailed Version of the Computer and Communication Lines Constraints</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Total Annual Spending on Data Processing</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Grosch's Law</td>
<td>27</td>
</tr>
<tr>
<td>Figure 3</td>
<td>A Communication Computer Network</td>
<td>33</td>
</tr>
<tr>
<td>Figure 4</td>
<td>A Homogeneous Computer Network</td>
<td>35</td>
</tr>
<tr>
<td>Figure 5</td>
<td>The ARPA Network</td>
<td>36</td>
</tr>
<tr>
<td>Figure 6</td>
<td>A Star Network</td>
<td>38</td>
</tr>
<tr>
<td>Figure 7</td>
<td>A Set of Stars Network</td>
<td>40</td>
</tr>
<tr>
<td>Figure 8</td>
<td>A Distributed Network</td>
<td>41</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Some Examples of Real-Life Networks</td>
<td>44</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Cost of a Computer Network</td>
<td>50</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Cost Breakdown of Computer Network</td>
<td>51</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Computing Cost Estimation With or Without Computer Network</td>
<td>52</td>
</tr>
<tr>
<td>Figure 13</td>
<td>A Homogeneous Distributed Database System</td>
<td>55</td>
</tr>
<tr>
<td>Figure 14</td>
<td>A Heterogeneous Distributed Database System</td>
<td>56</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Areas of Research</td>
<td>77</td>
</tr>
<tr>
<td>Figure 16</td>
<td>A Distributed Database System and Computer Network</td>
<td>81</td>
</tr>
<tr>
<td>Figure 17</td>
<td>A Hypothetical Network</td>
<td>91</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Hierarchical Structure of the Computer Model</td>
<td>133</td>
</tr>
<tr>
<td>Figure 19</td>
<td>The Bank's Distributed Network</td>
<td>146</td>
</tr>
<tr>
<td>Figure 20</td>
<td>The Bank's Dual-Processor Message Switching System</td>
<td>147</td>
</tr>
<tr>
<td>Figure 21</td>
<td>The Message-Switching Network</td>
<td>148</td>
</tr>
<tr>
<td>Figure 22</td>
<td>A Multilevel Distributive Processing Network</td>
<td>149</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Our Solution</td>
<td>158</td>
</tr>
</tbody>
</table>
Figure 24 -- The Recommended Bank's Network ............... 160
Figure 25 -- Comparison of the Solutions .................. 161
Figure 26 -- Sensitivity Analysis Results ................... 171
Figure 27 -- Cost of Database Assignment: Case 1 ......... 184
Figure 28 -- Cost of Database Assignment: Case 2 .......... 187
Figure 29 -- Different Storage Cost at the Nodes ........... 190
Figure 30 -- Cost of Database Assignment: Case 3 .......... 193
CHAPTER ONE

INTRODUCTION

I. Toward Distributed Information Systems

For almost two decades people have argued the comparative merits of centralizing or decentralizing data processing activities. There are several good reasons why this has been a subject of perpetual concern. Certainly one of the primary reasons is the large and increasing investment and operation budget allocated to EDP, as shown in Figure 1. This has drawn management attention to the need to make full use of this resource. Also, the increasingly wide-spread use of data processing systems has caused substantial dependence of many organizational units on their information systems. Because of this dependence, many authorities perceive control of information systems to be synonymous with political power in the organization. While discussions of the advantages and disadvantages of both structures continue, the fact that a great many data processing operations are simply not performing up to expectations becomes alarmingly clear.

In the past, centralization has been the major trend in EDP systems architecture. However, the advent of cost efficient mini and micro computers as well as recent breakthroughs in network technology have added credence to the realities of distributed processing, which can be seen as a significant and realistic system alternative. Although a strong argument can still be made for serving distributed users with
### Distribution of Spending

<table>
<thead>
<tr>
<th>Item</th>
<th>$ Billions</th>
<th>% of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personnel</td>
<td>7.07</td>
<td>9.99</td>
</tr>
<tr>
<td>Computer Hardware and Maintenance</td>
<td>7.53</td>
<td>11.88</td>
</tr>
<tr>
<td>Purchased Computing and Other Services</td>
<td>2.46</td>
<td>4.35</td>
</tr>
<tr>
<td>Data Communications</td>
<td>1.22</td>
<td>2.55</td>
</tr>
<tr>
<td>Supplies</td>
<td>.82</td>
<td>1.23</td>
</tr>
<tr>
<td>Total Direct Costs</td>
<td>19.1</td>
<td>30.0</td>
</tr>
<tr>
<td>Indirect Costs</td>
<td>2.6</td>
<td>4.1</td>
</tr>
<tr>
<td>Total Data Processing Spending in the U.S.</td>
<td>21.7</td>
<td>34.1</td>
</tr>
<tr>
<td>Average Annual Increase</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. Total Annual Spending on Data Processing in the United States in Current 1973 and 1976 Dollars. Adapted from (1).
centralized systems, we now witness an increasing number of systems in which information processing and storage function are distributed among several computers. The aim of this thesis is to study the design aspects of such systems.

II. Statement of the Research

The objective of this thesis is to propose optimization models and solution procedures for the design of distributed information systems. A distributed information system is defined as "a combination of information processing facilities, data communication facilities, and end point facilities. Together these support the movement and processing of files, programs, data, messages and transactions." (5)

Information processing facilities include a collection of processors, located at various stations. Data communication facilities include the transmission lines, coupling devices such as modems, and concentration devices such as multiplexors. The end point facilities include mostly terminal devices.

As we can see, two important components of a distributed information system are:

1. A computer network

2. A distributed database system

A computer network is composed of several computer systems connected by a communication network and used through terminals. A distributed database system is viewed as "a logical integration of several related databases localized in individual computing facilities." (6) In previous
work, most of the research focused on the following two separate problems:

- optimal distribution of files over a network of computer systems (7), (8), (9)
- design of a network of computer systems (10), (11), (12)

In the first group of problems, all the studies mentioned assumed that a network topology is given and that at each node of the network a computer of unlimited capacity is available. In the second group of problems, particular attention is devoted to the design of a computer network without taking into account the problems of files assignment. So far, network design and files allocation problems have been characterized by a piece-meal approach. This can lead to a suboptimal system. Our strategy is to unify all these approaches.

The objectives of the thesis are the following:

1. To propose a global model for the design of distributed information systems. This model will include design issues related to both computer networks and distributed database systems. Network topology, communication channels capacity and size of computer hardware, pricing schemes and routing disciplines are interrelated in an optimal design. Our assumption is, by encompassing all the aspects of distributed systems, we can attain an optimal configuration. This is fundamentally different from the previous approaches which dealt only with one aspect of the problem, therefore leading to a partial solution. Our intention is to incorporate in an overall design model, most of the factors that compose distributed information systems.
2. To derive from the global model for distributed information systems, an optimization model for the design of distributed database systems only. This is useful primarily for organizations having already access to a computer network without data sharing. The purpose of such a model is to find the optimal allocation of databases and programs over the network, and also to define the optimal routing disciplines for the messages flowing in the network.

3. To develop a solution procedure that can lead to optimal solutions of the models. In the past, most of the methods used to solve either computer network design models or distributed database system design models are heuristics. The main disadvantage of such methods is that they do not necessarily lead to the "true" optimal solutions. They usually provide only "acceptable solutions." Such heuristic methods, can lead to costly suboptimal solutions, when they are applied to real-life examples. Our objective is to develop a pure mathematical programming algorithm, allowing to solve exactly the design models for distributed information systems.

4. To provide an efficient computer code allowing to solve real-life distributed information systems problems, with a minimum programming efforts for the users. This can allow us to evaluate the efficiency of the mathematical programming algorithm proposed for the solution of the design models.

5. To apply our models and the solution procedure in real-life settings. The global model will be applied to design a distributed
information systems for a large bank. The design model for distributed
database systems will be applied using the same set of data as previous
authors (8), (9). This can allow us to compare their results with ours.

6. To investigate to what extent the global model for the design
of distributed information systems can be expanded to be used in the issue
of centralization versus decentralization of information systems: in
the past, the approach taken to solve this issue was mainly qualitative.
It is our intention to indicate how management science techniques in
general, and the global design model of this thesis in particular, can
be applied to help solve the issue of centralization versus decentrali-
zation of information systems. Since it is beyond the scope of this
thesis, we will not attempt to apply this model in real-life settings.

7. To provide a basis for further research in the area of design
models for distributed information systems.

In order to attain the objectives described above, we will use the
following research methodology.

III. Research Methodology

Since the main purpose of this thesis is to propose design
models for distributed information systems, we shall first develop
theoretical optimization models which take into account most of the
factors involved in such systems. This will be done by investigating
previous work and by relaxing most of the assumptions that have been
made in the past when the design problems of computer networks and
distributed database systems were studied separately. Then, we will bring into the models some aspects more pertinent to real-life settings such as security constraints and the return flow of information in distributed systems.*

The data needed to run the models will be gathered from a large bank designing its distributed information systems, and from an example used by Casey (8) and Levin-Morgan (9). A FORTRAN computer code of the algorithm solving the models will be developed and tested on several sets of data. Finally, to test the validity of the models, a sensitivity analysis will be performed.

IV. Outline of the dissertation

This dissertation will be organized as follows:

Chapter Two is a survey of the literature related to distributed information systems. It included a critical survey of the two components of distributed information systems described in paragraph I of this chapter, namely:

1. Computer networks and data communications
2. Distributed database systems

In this chapter, an attempt is made to identify major concepts, modelling issues, and assumptions, on the basis of which a critical literature survey is conducted. This is important since our design models and their solution complement other approaches and draw some concepts of them.

* These concepts will be defined in Chapter Three.
Chapter Three will address the issue of modelling, designing, and optimizing distributed information systems. We will first present a global model for the design of distributed systems which contains most of the factors relevant to computer networks and distributed database systems. The model will include computation power allocation, databases allocation, link capacities, message routing, and program sharing. A method to derive the design model of only distributed database systems (under the assumption of the existence of a computer network) will be described. This model will allow us to find the optimal allocation of databases and programs, given the network topology. A particular attention will be given to the issues of the return flow of information and storage cost of databases, and their consequences on the optimal allocation of databases. Some comparisons with previous work will also be made. Finally, some possible extensions of the models will be indicated.

The aim of Chapter Four is to develop a mathematical programming algorithm which can allow us to solve optimally distributed systems models. In this chapter, we argue that, given the important cost figures involved when designing distributed systems, the traditional heuristic methods can lead to costly suboptimal solutions. To avoid this disadvantage, we propose a pure mathematical algorithm leading to an optimal solution, regardless of the nature of the objective functions and the constraints (i.e., linear or non linear). Based on a revised version of the traditional branch and bound techniques, this algorithm has the following characteristics:
- It limits a priori the number of nodes of the graph, therefore allowing to solve large scale problems

- It converges rapidly, even though the objective function is nonlinear and non convex.

In this chapter, we will also describe the computer implementation of this algorithm.

The aim of Chapter Five is to apply the design models using the algorithm, in real-life settings. The global model will be applied using data gathered from a large bank designing an experimental distributed information system. The distributed database system model will be applied using data from Casey and Levin-Morgan's example. In this chapter, we will also focus on the differences existing between our solution and the bank's experimental distributed system. Some sensitivity analysis will be performed. Similarly, a comparison between Casey/Levin-Morgan's results and our solution will be made. Finally, some conclusions related to the optimal location of databases will be drawn.

The purpose of Chapter Six is to show some possible extensions of the first model of Chapter Three and its utilization in the issue of centralization versus decentralization of information systems. We will first give some background description of the issue and the concepts involved, then we will present the decision model for the issue of centralization-decentralization. No attempt will be made to apply the model, since it is beyond the scope of this thesis.

All the results obtained will be summarized and discussed in Chapter Seven. This chapter will provide also a good basis for further
research aimed at establishing a complete framework for distributed information systems. Some persistent and still unsolved problems related to distributed systems will be described.
REFERENCES

Chapter One

(1) EDP In-Depth Reports, November 1976.


May 1976.


CHAPTER TWO

DISTRIBUTED INFORMATION SYSTEMS

A STATE OF THE ART STUDY

I. Introduction

The literature dealing with distributed information systems is rapidly expanding. This literature may be classified in three categories:

- The management-oriented literature which is concerned with the issue of centralization versus decentralization of information systems and the management of distributed processing. [(1), (2)].

- The literature related to the issues of modelling, designing and optimizing distributed systems, including computer networks, distributed data communications and distributed database systems.

- The software-oriented literature which deals with issues such as protocol and currency control in distributed computer networks. [(3)].

This survey deals only with the second aspects of distributed information systems. The purpose of this survey is to establish the context of our approach and to set the terminology for our modelling issues. It is intended to identify major concepts used in distributed information systems, to describe modelling approaches used in the past,
and to present most of the assumptions on which previous works are based. This is important since our approach in modelling distributed systems and the solution procedure used, complement other approaches.

The plan of this survey will be as follows: in Section II, we present the major arguments used to justify the shift toward distributed information systems. This will involve an analysis of the previous centralized trend and the benefits expected from the new distributed trend. This background information is essential to understand the surge of interest in distributed information systems and their development as a potential alternative to centralized systems. In Section III, we describe relevant issues in computer networks, including network topologies, data communication systems, and networking design principles. In Section IV, we present an overview of distributed database systems which include the definition of major concepts and a presentation of past design models.

II. Major Arguments Used for Distributed Information Systems

Since the early 1950's, the trend in EDP installations has been toward centralization. There were several reasons for this trend, primarily economically motivated. Usually, three major arguments were used in favor of centralized systems: economy of scale, sophistication of applications, and quality of systems development.

A. Economy of Scale

This is one of the major arguments used to justify centralized
systems. It means that a large computer is more cost effective than a small computer. This conclusion was derived from "Grosch's Law" (4), which states that "the performance of a computer increases as the square of its cost," as shown in Figure 2. There were several reasons which justified the economy of scale. First of all, decentralized small computers may have unused capacity. Centralization on a large computer could eliminate such costs. In addition, individual small computers may be overloaded, generating pressure for upgrading equipment or purchasing expensive service bureau time. Centralization on a large computer could absorb this overload. While the arguments above can be valid for a particular centralized systems, many researchers and practitioners have questioned the validity of "Grosch's Law." They argue that "Grosch's Law" was valid during the 1950's and the 1960's when the CPU cluster was the dominant element in a computer. Today a dramatic change has occurred in the sense that the cost of the CPU is less than 40 percent of the total cost. Another argument used against "Grosch's Law" is the fact that the latter assumes that the power of a computer is proportional to its price. This is a very simplistic assumption, especially when one considers the structure of a machine, the variety of channels and their speeds, and the characteristics of secondary storages. Studies which invalidate "Grosch's Law" can be found in Littrel (6), Reynolds (7), and Lussato, et al. (8). These authors mainly argue that with the continuous reduction of hardware costs (20 percent yearly), the economy

*By performance we mean "cost effectiveness"
where:

\[ \frac{C}{E} = \frac{K}{C^2} \]

\[ C = \text{cost of the computer} \]
\[ E = \text{the effectiveness} \]
\[ K = \text{constant} = 1 \]

FIGURE 2. Grosch's Law - Adapted from (5)
of scale is becoming less and less important. The increasing amount of data being transferred between the central node and the dispersed users joined to the stability of communication costs are changing the shape of the economies of scale curve. Finally, the overhead associated with very large computers and the potential for under-utilizing the capacity of a large centralized computer combine to diminish the validity of "Grosch's Law."

An argument related to the concept of economies of scale and used in favor of centralized systems is the one stating that in terms of floor space, electricity, air conditioning and other facility costs, a single large installation is less costly than multiple smaller installations. Although this argument may still be valid, the changing technology of minicomputers and the intrusion of microcomputers can lead to a less important economy of scale. In fact, some minicomputers when used in distributed systems and most of the microcomputers do not need important facilities, and can use only very limited floor space, electricity and other facilities. An additional argument related to the economy of scale and used to justify centralization is the following: "The number of support personnel is lower for a large installation than for multiple small installations." This argument is true if one assumes that the complexity of minicomputers and microcomputers is as big as the complexity of large systems. In fact, mini and micro computers are fairly easy to operate (8). Besides, the development of new concepts like decision support systems (9), where systems are being tailored to the particular requirements of managers, favors the reduction of support and specialized personnel.
In summary, the advances in the computer technology and the development of new concepts reduce the importance of economies of scale, therefore clearing the way for distributed systems.

B. Sophistication of Applications

To justify the centralization of information systems, other arguments besides the economy of scale have been used. An important one is related to the sophistication of applications. In other words, there may exist certain applications which need high internal speed, great storage capacity and specialized personnel. These kinds of applications are not feasible in smaller installations. Some examples may include scientific computation, database management systems, and the access to hierarchically structured files for manufacturing systems. In such cases, the application would justify the larger computer, which would, in turn, justify the elimination of smaller computers in the organization in order to utilize the excess capacity of the large machine. The logic behind this type of argument is that decentralized systems are incapable of providing this service. A careful investigation of the capabilities of mini computers can show that: "like the mainframes, minis provide high performance processing, data and file handling capabilities, large amounts of on-line main storage, and high level languages like COBOL, RPG, and BASIC." (10)

The argument about database systems and hierarchical files is not valid anymore since the concepts of distributed files and distributed databases systems has shown to be viable and efficient. (11) In fact,
the degree of sophistication of applications does not depend entirely on the equipment used.

C. Quality of Systems Development

This is the third major argument used to justify centralized systems. The latter are said to establish and enforce systems documentation standards, to regulate standards for user documentation, to avoid redundant development of similar systems for different divisions of organization, and to allow an evaluation of projects from an overall organization perspective.

Although from a theoretical point of view, these advantages can be achieved, in reality, the huge centralized systems lead to an enormous complexity (12). This complexity is due to the necessity of handling large volumes of batch work. The resulting consequence can be a system failure or some difficulties to maintain a coherent system standard.

A related factor which has been argued in favor of centralization is the substantial difference between the abilities of large and small installations to attract and retain highly qualified technical personnel. It is argued that the smaller installation will frequently suffer a higher turnover rate as talented individuals outgrow the opportunities available. The retention of highly qualified personnel provided the centralized group with the capability to apply a higher level of expertise to the solution of problems. This personnel can then provide a greater range of alternative solutions to the problems for evaluation by management, resulting in a lower cost of development, operation, and
further maintenance of the systems. However, behavioral scientists such as Herzberg (13) and Maslow (14) argue that one of the real factors that can contribute to attract highly qualified personnel is motivation rather than the size of the companies. In decentralized systems, the possibility of being associated to the decision-making process can be a real factor to motivate skilled personnel.

Proponents of centralization have argued their case on additional grounds ranging from the benefits of company-wide consolidation of operating results to the ease of control by corporate executives. Many of these arguments, however, boil down to matters of management style. This adds additional confusion to the issue because styles change as the pendulum of management philosophy swings to and fro. All these and other such arguments have contributed to make a persuasive case for centralization. Indeed, centralization has been the major trend in EDP systems architecture for the last twenty years. Lately, however, a new and innovative approach to systems architecture has appeared. Distributed processing has bloomed into major prominence as a technique for increasing the efficiency of EDP operations. This is not meant to assert that distributed processing optimally satisfies every organization's EDP needs. As with most technologies, certain tradeoffs must be considered between efficiency and effectiveness.

But, the advent of cost efficient mini and micro computers as well as recent breakthroughs in network technology have clearly added credence to the realities of distributed processing. Let us precisely describe these technology advances and the underlying concepts.
III. **Computer Networks: A Survey**

A large body of literature has been published on the subject of computer networks. The computer systems section of the Institute of Computer Science and Technology, National Bureau of Standards (15) has published a fully annotated bibliography on the subject. The bibliography consists of references with critical annotations to the literature on computer networks. Instead of duplicating the effort made in the above Institute, we will concentrate on several aspects that were incompletely covered by the bibliography.

The aim of this section is to clarify usage of terminology and to provide an analysis of the characteristics of computer network models. The aspects of networking that will be surveyed are the following:

- network topologies
- communication systems
- networking design principles

A. **Network Topologies**

A computer network is composed of several computer systems (called "hosts" following ARPA (16) terminology) connected by a communication network, and used through terminals: terminals can be connected directly to one of the hosts or to the communication network (17), described in Figure 3.

One of the most important functions of a computer network is to give to each user the possibility of using computing resources in the
FIGURE 3. A Communication Computer Network
network regardless of their geographical sites (18). By resources we mean CPU's, peripheral devices, program libraries and databases.

There are two different types of computer networks: homogeneous and heterogeneous. Homogeneous computer networks are made up with the same brand of machines and operating systems. A typical example is described in Figure 4. It represents the SOC (Systems d'Ordinateurs Connectés) project which is a joint study carried out by IBM France and the computing centers of four French participants.

Except for internal corporate networks, homogeneous computer networks are the exception rather than the rule. Most computer networks are heterogeneous since organizations have to start out existing installations. A typical example of heterogeneous networks is described in Figure 5.

When one looks at the different possibilities of network sharing, there exist three functional forms of computer networks: remote-access networks, value-added networks, and mission-oriented networks (respectively RAN, VAN, MON). Remote-access networks are designed to support interaction between an end user and a given host computer. The service provided by remote-access networks can be divided into two categories: terminal access and remote batch. Examples of such networks are TYMSHARE (19), and CYBERNET (20). A value-added network does not support interaction between an end user and individual host computers. Instead, it supports communication directly between host computers. Therefore, it includes the possibility of file transfers, purging of remote databases, and geographically dispersed multiprocessing. An example of such added-value
FIGURE 4. A Homogeneous Computer Network
FIGURE 5. The ARPA Network. A logical map.

Adapted from (78).
network is the ARPA network (16). A mission-oriented network exists when there is a closer organizational coupling of the host computers and the subnet providing communication resources. Therefore, the difference between VAN and MON is organizational but not technological.

In fact, a MON is a value-added network in which host computers are under the control of a single administrative organization. According to Kimbleton et.al. (78), this organizational distinction "permits allocation and control of programs, data, and their interaction within the network thereby maximizing the efficiency of the organizational information processing function."

In summary, the distinction between a RAN and a NON is reflected in enhanced technological capabilities, while the difference between a VAN and a MON is primarily organizational.

A variety of network technologies is possible. These topologies can be classified according to the extent to which processing power is distributed among the host computers. There are mainly three basic network topologies (32):

- Centralized network
- Decentralized network
- Distributed network

In a centralized network (often called star network), all terminal stations are centrally connected to a central node, and the topology of the network represents a star as depicted in Figure 6. As can be seen, the network is composed of a single host computer which services all users.
FIGURE 6. A Star Network
Remote-access networks such as TYMSHARE (19) are typically implemented in this form.

In decentralized networks, there is a topological structure of sets of stars connected in a form of a larger star with an additional link forming a loop. The main characteristic of such a network is that the reliance upon a single point is not always required, as described in Figure 7. In distributed networks, a variety of host computers may be accessed by network users. Subscribers of the networks are served by many switching centers distributed throughout the network. The center in such networks may be either a multiplexer, concentrator or communication processors associated with the required computer. Figure 8 depicts a typical distributed network. It should be noted that the topologies presented are very general and often used. But, there exist other topologies which are a sub-variety of those described above.

B. Communication Systems

Various techniques may be considered when it comes to setting up some communication systems between computers. At present, there exist three distinct techniques:

1. circuit switching
2. message switching
3. packet switching

The main distinction between these techniques resides in the manner in which resources are allocated in support of communication.
FIGURE 7. A Set of Stars Network
FIGURE 8. A Distributed Network
1. **Circuit switching**

Circuit switching is the conventional method used in the telephone networks wherein a discrete path is established between a sender and receiver and the path is held open for the duration of the transmission. The nodes of the network perform only a switching operation between the input lines and the output lines according to the distribution of the messages. Effective utilization of circuit switching requires careful matching of circuit bandwidth against the transmission requirements to improve this efficiency multiplexing technique [21]. The design for such networks has been studied by Chu [22] and Martin [24].

2. **Message switching**

In message switching networks, the messages between two nodes are "stored" in a queue at any intermediate node and sent "forward" to the next node on the route only when the channel is free. So, at each node, there are different pieces, one for each output channel. A description of a major message switching network (AUTODIN) is given by Paoletti [24]. Although it is largely used in computer networks, the message-switching technique has not been proved to be more convenient than circuit-switching techniques [25]. Kleinrock (26) has provided a detailed study of message-switching networks.

3. **Packet switching**

In packet-switching networks, a message is subdivided into packets with a pre-established maximum size (typically on the order of 1000 bits) prefaced with suitable address information. Single packet messages can then be transmitted with a minimum of delay.
In a packet-switched system:
- Much of the use is interactive, exploiting the rapid response, so the loss of a packet is easily corrected by the users;
- Storage in the network is kept low to reduce transit delays;
- Terminal users are not involved with packet headings and will usually be unaware of the packet-switching aspect;
- Some of the users of the network are computers with exchange packets with many other terminals, so their connection to the network interweaves packets from several connections.

The packet-switching method was adopted by ARPA in 1968 and is presently used by the CYCLADES [27] network. Some networks, like SITA [28] use a combination of packet and message switching. Analysis of the tradeoffs which must be considered in evaluating circuit, message, and packet switched technologies was made by Miyahara et. al. [29] and Wood [30].

The packet-switching technology seems to make the best use of existing technologies and it can be supported by telephone, radio, or satellite transmission. Besides, it permits simultaneous support of multi-nodal (real-time, interactive and file transfer) traffic.

C. Computer network design principles

The requirement of minimizing the overall cost of a network and of better utilization of the available resources are becoming the prime factors of interest. The design of an optimal computer communication network covers many elements of which the following will affect the cost and performance of the system: end users, terminals, data sets, communication lines, multiplexers, concentrators, transmission control units, communication processors and computing facility. It is essential for
the network designer to obtain the following design data:

- Geographical location of each terminal or set of terminals,
- Amount of data transmitted and received from each geographical location,
- Required response time,
- Growth rates in terms of number of locations and traffic,
- Security restrictions in the selection of terminal types,
- Cost limit.

Most of the research considers that the topological allocation of the computers to be connection by the network and the traffic required between any pair of them are assigned. Therefore, the parameters that are not imposed by the problem are:

- **The network topology**: the structure of the network connecting the computers.
- **Capacity assignment**: the choice of the capacity of each link in the network.
- **Routing procedure**: the decision needs which routes a message from one node to another.
- **Queue discipline**: the priority rule which determines a message's relative position in the queue.
- **Message delay**: the total time that a message spends in the network.
- **Network cost**: the total cost of the system which is given by the sum of channels and terminals cost.
The expression of the problem is generally the following:

\[
\begin{align*}
\text{min} \quad & \text{(Total average message delay)} \\
\text{over} \quad & \text{topology, capacity assignment, routing procedure, priority discipline} \\
\text{subject to a given maximum total cost of the system}
\end{align*}
\]

A very similar formulation is the following:

\[
\begin{align*}
\text{min} \quad & \text{(Total cost)} \\
\text{over} \quad & \text{topology, capacity assignment, routing procedure, priority discipline} \\
\text{subject to:} \\
& \text{a given maximum total average message delay}
\end{align*}
\]

It has been shown by Fratta et. al. (31) that the two previous formulations are dual one to each other, that is the optimal solution for the first problem is also the optimum for the second problem. Most of the research done in this area takes the above approach (26) (33) often using different expressions for the objective function and the constraints. Various network designs have been studied in (34), (35), (36). Chang (37) developed a model for distributed computer system design that relaxes
the constraint related to the topology of the network. His model is formulated as a problem of deciding transaction allocations, routing, processor allocation, and line allocation to satisfy certain performance requirements. Although more complete than the preceding network design approaches, Chang's model does not fully take into account the issue of distributed database systems and their integration in the overall design approach. A tentative toward more complete approach was taken by Modiano (38). He studied a simultaneous distribution of computation power, file allocation, and link capacities. But, his paper ignores the issues of "data sharing" and the routing disciplines.

D. Solutions for the optimum design

In general, a computationally feasible algorithmic solution for determination of the optimum design is not available. This is due to the complexity of the formulation and the large size of the problem. Algorithmic solutions which can guarantee the obtainment of an optimal solution are very often computationally expensive. This may explain why most of the solutions proposed are heuristic. Most of the problems of network design are in fact a multicommodity flow problem. White (39) classified the solutions in two categories:

- decomposition techniques
- partitionning techniques

It seems that the decomposition approach leads to a more efficient solution.

Some design problems are unconstrained multicommodity flow problems with non linear costs. A good algorithm to solve this kind of problem
is provided by Cantor and Gerla (40). Their method, called the flow-deviation method, is very similar to the gradient method for functions of continuous variables. But, the concept of gradient is replaced by the concept of shortest route. The application of successive flow deviations can lead to a local minima.

A heuristic that can lead to an acceptable solution was presented by Chou and Frank (34). Other heuristics have been developed especially for the issue of distributed database systems and will be surveyed later on. Kleitman and Claus (41) and Chang (37) developed different heuristics to determine the network topology. Some solution approaches have been presented by Frank (42), (43), (44).

A comparison of network topology optimization algorithms is provided by Whitney (45). He compares the following five algorithms: minimal spanning tree (46), constrained minimal spanning tree (47), sectoring (47), reversed CMST (46), and steepest ascent hill climbing (48). The five procedures have been uniformly coded and applied to a variety of test configurations. The results show that the steepest ascent hill climbing procedure is better than the other algorithms. De Backer (49) compared the following four algorithms used in the design of centralized networks: Kruskal (50), Prim (51), Esaul William (52), and Vam (53). He found that the Vam's algorithm generates the best solution.

Most of the algorithms described in this section have a common characteristic: they are heuristic methods. The main advantage is that they generate "acceptable solutions" in a very small computing time. But, these procedures, when applied in real life situations (like, for a corporation, decentralizing their information systems), lead to very
costly suboptimal solutions.

Before ending this review on computer networks, let us give some examples of private and public networks. Figure 9 describes eight different networks and some of their characteristics. The reader should be aware that these networks represent only a few examples of existing networks.

Finally, there are only a few indications for the amount of expenses to realize a computer network. As far as networks were constructed and built up in scientific institutions, it is not possible to give an analysis of investment costs. The reasons are due to the fact that activities of scientific institutions are sponsored by government and industrial foundations. The people involved in such a project must not earn their salaries out of the returns of the network. However, we can evaluate an approximate cost of a computer network. Kreuzberger (54) summarized the monthly costs of the Infonet computer network. His evaluation is given in Figure 10. Hughes and Mann (55) give their breakdown of total cost in Figure 11. The benefit (in terms of hard dollars) that can be obtained through the use of a computer network is very difficult to evaluate.

For the ARPA network, several contractors were making substantial use of the network for a majority of their computing resources. Several of the computing centers on the network had grown to become substantial suppliers of computer service. At that time, an accounting was made of the total computer usage obtained through the network, and an estimate was made for each user of the cost of purchasing comparable time on outside computers on leasing the necessary in-house computer facilities to
<table>
<thead>
<tr>
<th>Country</th>
<th>Membership</th>
<th>Type of Package Service</th>
<th>Host Interface Type</th>
<th>Maximum Speed</th>
<th>User Input</th>
<th>Terminal Interface</th>
<th>User Input</th>
<th>OS Modification</th>
<th>User Capabilities:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Terminal-Computer</td>
</tr>
<tr>
<td>CIGALE/</td>
<td>US</td>
<td>France</td>
<td>Spain</td>
<td>Japan</td>
<td>Europe</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CYCLADES</td>
<td>Private</td>
<td>Public</td>
<td>Public</td>
<td>Public</td>
<td>Private</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>CTNE</td>
<td></td>
<td>Virtual Circuit</td>
<td>Virtual</td>
<td>Virtual Circuit</td>
<td>Datagram</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DDX-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Packet</td>
<td>Packet</td>
<td>Packet</td>
<td>Packet</td>
<td>No</td>
</tr>
<tr>
<td>EIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Packet</td>
<td>Packet</td>
<td>Packet</td>
<td>Packet</td>
<td>No</td>
</tr>
<tr>
<td>RCP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Packet</td>
<td></td>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Membership</th>
<th>Type of Package Service</th>
<th>Host Interface Type</th>
<th>Maximum Speed</th>
<th>User Input</th>
<th>Terminal Interface</th>
<th>User Input</th>
<th>OS Modification</th>
<th>User Capabilities:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Terminal-Computer</td>
</tr>
<tr>
<td>France</td>
<td>Private</td>
<td>Datagram</td>
<td>Serial</td>
<td>HDLC</td>
<td>Non</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spain</td>
<td>Public</td>
<td>Virtual Circuit</td>
<td>Serial</td>
<td>N/A</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td>Public</td>
<td>Virtual Circuit</td>
<td>HDLC</td>
<td>4.8Kb</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Europe</td>
<td>Private</td>
<td>Datagram</td>
<td>HDLC</td>
<td>130Kb</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UK</td>
<td>Public</td>
<td>Virtual Circuit</td>
<td>Serial</td>
<td>N/A</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UK</td>
<td>Private</td>
<td>Datagram</td>
<td>British Standard</td>
<td>130Kb</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>France</td>
<td>Public</td>
<td>Virtual Circuit</td>
<td>Serial</td>
<td>48Kb</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

N/A = Information Not Available

FIGURE 9. Characteristics of Eight Different Networks. (Adapted from (79)).
Hardware Costs

4 Processors of type 4004/151 incl. large peripherical resources 1,000

20 intelligent terminals installed in Service-Bureaus 20 x 20,000 0,400

Transmission Costs

modems 0,300

leased lines 0,300

Cost of Personnel

Operating: Computer Centers 40
Service Bureaus 40

Hardware Specialists 10

Research and Development

Systems Software 20
Application Packages 80

Sales 60

Sales Support 90

Administration 60

400 x DM 5000 2,000

Depreciation and other costs 1,000

Total Cost 5,000

*1 DM = 35.52¢ in 1974

FIGURE 10. Monthly Costs of a Computer Network
Adapted from (54)
FIGURE 11. Cost breakdown for computer network
Adapted from (55)
do the same job apparently being done through the network. Roberts (56) tabulated this information by user organization, identifying both the cost of computing with and without the network. His results are reported in Figure 12. Additional studies in economics of computer networks can be found in (57) and (58).

In summary, and as is apparent, the computer network design problem is a very rich one and has not been solved in all its generality. Especially, the closely related issue of distributed database systems has not been adequately taken into account in most of the models surveyed. This is due to the fact that the viability of distributed database systems has not been emerging until very recently. The purpose of the following section is precisely to describe the main characteristics of distributed database systems and past approaches taken to design such systems.

IV. Distributed Database Systems: An Overview

A. Introduction

One important and valuable feature of distributed processing is the distributed database application. There are two aspects of distributed database systems:

1. The distribution of database management (i.e., the control and manipulation of data)

2. The distribution of the content of the database (i.e., the data itself)
<table>
<thead>
<tr>
<th>User Organization</th>
<th>Activity</th>
<th>Remote Cost in thousands</th>
<th>Projected Cost for Local Replacement in thousands</th>
</tr>
</thead>
<tbody>
<tr>
<td>University of Illinois</td>
<td>Parallel processing research</td>
<td>$360</td>
<td>$1,100</td>
</tr>
<tr>
<td>NASA Ames Research Center</td>
<td>Air foil design and ILLIAC</td>
<td>328</td>
<td>570</td>
</tr>
<tr>
<td>Rand Corporation</td>
<td>Numerical climate modeling</td>
<td>210</td>
<td>650</td>
</tr>
<tr>
<td>Mass. Computer Associates</td>
<td>ILLIAC IV compiler development</td>
<td>151</td>
<td>470</td>
</tr>
<tr>
<td>Lawrence Livermore Laboratory</td>
<td>TENSOR code on ILLIAC</td>
<td>94</td>
<td>370</td>
</tr>
<tr>
<td>Stanford University</td>
<td>Artificial intelligence research</td>
<td>91</td>
<td>180</td>
</tr>
<tr>
<td>Rome Air Development Center</td>
<td>Text manipulation and resource evaluation</td>
<td>81</td>
<td>450</td>
</tr>
<tr>
<td>ARPA</td>
<td>On-line management</td>
<td>77</td>
<td>370</td>
</tr>
<tr>
<td>Seismic Array Analysis Center</td>
<td>Seismic data processing</td>
<td>76</td>
<td>300</td>
</tr>
<tr>
<td>Mitre Corporation</td>
<td>Distributed file network research</td>
<td>60</td>
<td>240</td>
</tr>
<tr>
<td>National Bureau of Standards</td>
<td>Network research</td>
<td>58</td>
<td>200</td>
</tr>
<tr>
<td>Bolt Beranek and Newman Inc.</td>
<td>TENEX system support</td>
<td>55</td>
<td>80</td>
</tr>
<tr>
<td>Xerox Palo Alto Research Center</td>
<td>Computer science research</td>
<td>47</td>
<td>100</td>
</tr>
<tr>
<td>University of Southern California</td>
<td>Picture processing research</td>
<td>35</td>
<td>70</td>
</tr>
<tr>
<td>Image Processing Lab</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of California, Los Angeles</td>
<td>Network measurement</td>
<td>28</td>
<td>90</td>
</tr>
<tr>
<td>Systems Control, Inc.</td>
<td>Signal processing research</td>
<td>23</td>
<td>70</td>
</tr>
<tr>
<td>University of California, Santa</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barbara</td>
<td>Network research</td>
<td>22</td>
<td>70</td>
</tr>
<tr>
<td>US Air Force Range Measurements Lab</td>
<td>ARPA NET management</td>
<td>17</td>
<td>60</td>
</tr>
<tr>
<td>Institute for the Future</td>
<td>Teleconferencing research</td>
<td>13</td>
<td>40</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Computer research</td>
<td>192</td>
<td>580</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>$2,018</strong></td>
<td><strong>$6,060</strong></td>
</tr>
</tbody>
</table>

*Annual remote computer usage cost is based on March 1973 data.

FIGURE 12. Cost of computing with and without the ARPA network. (Adapted from (56))
The need for data sharing shows up in many application areas, such as:

- on-line banking
- order entry
- retail sales
- reservation systems

The suggested need for data sharing between computers is found in large geographically distributed organizations. This is especially true when the data is processed locally but there also exists frequent exchange of information between sites.

Chandra (59) defines a distributed database as "a logical integration of several related databases localized in individual computing facilities." He made a distinction between homogeneous distributed databases and heterogeneous distributed databases. A homogeneous distributed database is composed of several databases having the same DBMS. For example, several IMS databases residing on a network form an homogeneous distributed database. (See Figure 13). An heterogeneous distributed database is composed of several different DBMS. An example is depicted in Figure 14.

Mullery (60) discusses in detail many of the problems that must be solved before such distributed databases can become viable in organizations.

B. Classification of distributed databases

Aschim (61) proposes the following classifications:

1. Classification according to geographical location of sites and directories.
IMS = Information Management System (IBM's DBMS)

FIGURE 13. A homogeneous distributed database system
FIGURE 14. A heterogeneous distributed database system
2. Classification according to distribution and types of Data Management Systems (DMS) used in the network.

In the first category, organizations can adopt one of the following possibilities:

- Centralized files, centralized directory: the entire database is effectively maintained at a single central location. The users situated in the nodes of the network access the distributed database via communication lines.

- Distributed files, distributed directory: this kind of organization is particularly suitable for the banking industry.

- Distributed files, centralized directory: this type of organization may be suitable when the files at the different sites contain large records of the same type.

- Centralized files, distributed directories: by having copies of the directory at the nodes of the network, much of the necessary processing may be done locally, thus reducing the system's response time.

- Distributed files, centralized and local directories: maintaining a local directory at each node in addition to a centralized directory may be a good alternative, especially when it is difficult to tell at what node the required information is stored.

Although from an organizational viewpoint, the first type of category can make sense, it is very useful to have another category using the DBMS as criteria. The following categories are the most common:

- Centralized system, one DBMS. This is the simplest form for
data sharing.
- Centralized system, several DBMS. It assumes that the users know all the DBMS he needs to use.
- Distributed system, same type of DBMS. This is a typical homogeneous distributed database system.
- Distributed system, several types of DBMS. This is the most flexible type of database network. But, some software problems are still unsolved in this type of organization.

Booth (62) classifies distributed database systems into two structures, partitioned databases and replicated databases. A partitioned database is one that has been decomposed into physically separate units distributed across the nodes of the network. A replicate database is one where all or part of it is being replicated at multiple nodes of the network.

A detailed list of the advantages of distributed database systems can be found in (61). The main advantages are:
- Network economies
- Availability
- Response time.

Alsberg's work (63) shows that a distributed database system can offer interesting economies by capitalizing on resource variety and by the proper exploitation of the topology of the network.

Belford (64) showed that the availability improves when a distributed database system is used. Belford (65) developed a mathematical model for response time in a distributed database environment. She showed that response time can be improved by load sharing.
C. Design models for distributed database systems

In previous works, most of the research focused on the problem of minimizing the operating cost of a distributed database. A great deal of attention has been devoted to the problem of optimal distribution of the files over a network of computer systems.

One of the earliest studies of the file allocation problem was done by Chu (66). He developed a linear-programming model allocating files so that the allocation yields minimum overall operating costs subject to the following constraints: (i) the expected time to access each file is less than a given bound, (ii) the amount of storage needed at each computer does not exceed the available storage capacity. His model includes storage costs, queuing delays, and communication costs. But he assumed that the number of copies of each file in the system is known. In a later paper, Chu (67) developed a procedure to determine in advance how many redundant copies of a file are required to achieve a desired level of reliability. Then, he inserted this number into the model, and the basic scheme remains unchanged.

Whitney (68) also formulated a similar model. He applied it to the design of a network topology and to the allocation of file copies. A communication network optimization procedure is developed. He showed that, for certain communication cost functions, the tree topology is less expensive than any non-tree topology. In addition, he showed that the system delay is minimized when there are as few independent channels as possible.

Casey (69) developed a procedure for finding a minimal cost solution.
Heuristic methods are used in this paper to find "good" solutions. The main difference between his paper and Chu's paper (67) is that the number of copies of files and their locations are treated as variables. He showed that the proportions of update traffic to query traffic generated by the users of a given file in the network could be used to determine an upper bound on the number of copies of the file present in the least cost network. He applied his algorithm to real data for the ARPA network and has thus shown the process feasible for networks of moderate size. He indicated that when update traffic equals query traffic, it is efficient to store all files at a central node.

Recently, Levin and Morgan (70) (71) developed models that allow dependencies between files and programs. In another paper (72) they developed a dynamic model for the multi-period case. In this model, the access time requests are assumed to be known for the next T periods. However, the assumption that the access request patterns are static over time was relaxed and a dynamic model which considers transition costs was suggested.

In a recent paper (73) Levin and Morgan provided a framework for research in optimizing distributed databases. They developed three models related to static file assignment with complete information, dynamic file assignment with complete information, and file assignment with incomplete information.

In a recent study, Chu (74) developed several models to study the performance of file directory systems for operating in the star network and distributed network topologies. He studied the cost-performance tradeoffs of three classes of directory systems. Assuming that the
transmission cost is much higher than the storage cost, he showed that for low directory update rates (less than 10 percent of the query rate), the distributed file directory yields a lower operating cost than the centralized directory system.

A particular attention should be devoted to Casey's paper (75) dealing with the design of tree networks for distributed data. He formulated a model locating information resources and choosing a topology for a network of distributed data files. In this model, he retains features such as discrete capacity assignment, economy of scale, and distinction between query and update transactions. He developed a heuristic method and formulated an algorithm solving the problem. The algorithm was tested for the special case of tree design.

In their paper, Mahmoud and Riordon (76) examine simultaneously the problems of file allocation and of link capacity allocation in order to achieve a minimum cost-design subject to constraints of file availability and network delay. The objective function contains two main costs: communication and file storage costs. The set of constraints is: the delay constraints and the file availability constraints.

The major contribution of this model is due to the fact that:

(i) it lifts up some restrictive assumption made by Casey (i.e., the reliability constraint and the time delay constraint)
(ii) it allows the allocation of link capacities.

An analysis of the papers surveyed above reveals some of the restrictive assumptions implicit in the models:
1. A fixed network topology is assumed.
2. A computer of unlimited capacity is assumed to be available at each node of the network.
3. The message routing disciplines are assumed to be known.
4. There are no dependencies between files and programs (except in the Levin-Morgan model)
5. There is no reliability constraint.
6. There is no link capacities assignment (except in the Mahmoud-Riordon paper)
7. There is no constraint on the time delay (except in the Levin-Morgan, and Mahmoud-Riordon papers)
8. The return flow of information is assumed to be null.

As it will be shown in Chapter Four, our approach lifts up most of these constraints.

Finally, there are a number of distributed database systems that have been implemented and are now operational. Among them are the following, described by Champine (77):

1. SITA (Societe Internationale de Communications Aeronautiques).
   It is a system of communication between airline database for the purpose of making passenger reservations on airplanes in Europe and Asia. This is a partitioned database system with the directory method of routing exception transactions.

2. CELANESE.
   It is a large United States textile manufacturer with a number of geographically distributed facilities with principal locations
at Charlotte, South Carolina, and Shelby, North Carolina. Each location has its own computer. An integrated database is maintained jointly at a number of nodes of the network. Celanese maintains duplicate files at both Charlotte and Shelby, with periodic updates from both locations. In addition, each location also has local data.


The Bank of America is a large financial institution headquartered in California, with over 1,000 branches and 11 million accounts. Data processing services are provided on a batch basis from centers in San Francisco and Los Angeles. An on-line teller information system was established. The database is partitioned between the two data centers, with exception transactions from one node forwarded to the other for processing. Each node is fully redundant so that processing can continue in spite of a hardware failure. The distributed data base approach is estimated to have a lower development cost by 40 percent than a centralized approach and is estimated to be some $4 million lower in yearly operating cost. The savings from this system are expected to exceed those operating costs by $3 million per year.

4. ARPANET

A software system entitled RSEXEC (Resource Sharing Executive System) has been developed for use on the ARPANET to provide distributed resource sharing. A major characteristic of RSEXEC is a distributed file capability which spans host computers
and supports uniform file access and automatic maintenance of replicated files. RSEXEC is currently operational on ARPANET.

5. LOWES COMPANIES, INC.

Lowes Companies, Inc. is a chain of 140 retail lumber/hardware stores located in the southeastern United States. In the early 1970's, the decision was made to move to an on-line inventory control and customer invoicing system. A centralized system was first considered, with the database located at the corporate headquarters and communication lines connecting it to terminals in all 140 stores. For many reasons, including heavy communications cost, a distributed system was examined.

The approach finally implemented was to install a minicomputer with disc and up to 16 terminals at each store. Each store is connected to the (functionally distributed) central system by data communication lines, and inventory and sales summary information is transmitted to the central system automatically each night. The system also receives information from the central site each night, such as new price information.

System installation is now basically complete. Store personnel have immediate on-line access to inventory, pricing, and customer account information which yields a 30 percent increase in sales person efficiency. It also yielded intangible but definite improvements in control over credit sales, accounts receivable, and price accuracy which results in additional cost savings.

The Lowes system is, then, a partitioned database with all files and transactions handled locally.
Aeroquip Corporation, a subsidiary of Libby-Owens-Ford, is a manufacturer of fluid power components, including hoses, fittings, and couplings in many sizes. The company had been using an on-line query system locally at the corporate headquarters for inventory control and order processing, and wished to extend the capability to a number of other manufacturing locations scattered from Georgia to Oregon. The objective was to provide on-line order processing, inventory control, credit checking, and shipping documents, to facilitate shipping most orders within 24 hours after receipt.

The classical centralized database approach was examined first and found to be too expensive from a data communications standpoint.

The system ultimately selected uses intelligent terminals at each node, supported by a local database on disc. These intelligent terminals are connected by low performance communication lines to the central database on a large scale mainframe, where a complete copy of the entire database is maintained. Each node maintains a local subset of the master database that is needed by that node.

In operation, the central site dials each node once every five minutes over a WATS line for transactions that cannot be handled locally. At night the central site dials each node in turn and obtains the accumulated transactions in compressed form and sends back updated records and output reports. The system became

This approach, then, is a partitioned database with exception transactions sent to the central node where they are handled by a complete copy of the file.

V. Conclusion

From this survey of distributed information systems, we can draw the following conclusions:

- Given the increasing network technology advancements, the trend toward distributed systems is made possible. Therefore, there is a potential alternative to the previous trend toward centralized systems.

- Given the decreasing hardware costs and the competitive mini computer market, distributed systems become economically feasible. As a consequence, organizations may feel reasonable to give each manager the responsibility and the resources of his own DP operations. This can lead to a greater user involvement which in return, can be beneficial to the organizations.

- As far as the design issue of distributed systems is concerned most of the research done in the past tends to consider the computer network design issue and the distributed database design issue separately. This is probably due to the fact that distributed database systems were developed as a con-
sequence of computer networks. This separate approach to both issues is one of the weaknesses of most of the models described in this chapter. The aim of the next chapter is to show how to combine both issues in an overall design approach.
REFERENCES

Chapter Two


(11) CHAMPINE, G.A., "Six Approaches to Distributed Databases," Data-
mation, May 1977

(12) GRUENBERGER, F., Report, 14th Annual One-Day Computing Symposium
    on Complexity and Communication, Bureau of Business Services
    and Research, California State University, 1972


(14) MASLOW, A., "Motivation and Personality," Harper and Row, New York,
    1954

(15) BLANC, R.P. et. al., "Annotated Bibliography on the Literature on
    Resource Sharing Computer Networks," NBS Special Publications,
    September 1973

(16) HEART, F.E., R.E. KAHN, S.M. ORNSTEIN, W.R. CROWTER, and D.C.
    WALDEN, "The Interface Message Processor for the ARPA Computer

(17) ORNSTEIN, S.M., F.E. HEART, W.R. CROWTER, H.K. RISING and S.B.
    RUSSELL, "The Terminal IMP for the ARPA Computer Network,
    Spring Joint Computer Conference, 1972, 243-254

(18) ROBERTS, L.G., B.D. WESSLER, "Computer Network Development to
    Achieve Resource Sharing," Spring Joint Computer Conference 1970,
    pp. 543-549

(19) TYMES, L., "TYMNET, A Terminal-Oriented Communication Network,
    in Proceedings AFIPS 1971, Spring Joint Computer Conference,
    Vol. 38, 1971, pp. 211-216

(20) LUTHER, W.J., "The Conceptual Basis of Cybernetics in Computer
    Networks," Randall Rustin, (ed), Prentice Hall, 1973, pp. 142-
    196


(25) PORT, E., F. CLOSS, "Comparison of Switched Data Networks on the Basis of Waiting Times," IBM Zurich, Report RZ405, January 1971


(39) WHITE, W.W., "Mathematical Programming, Multicommodity Flows and Communication Nets," MRI


(41) KLEINTMAN, D.J. and A. CLAUS, "A Large Scale Multicommodity Flow Problem: Telepak," MRI

(42) FRANK, H., R.E. KAHN and L. KLEINROCK, "Computer Communication
Network Design—Experiment with Theory and Practice," *Proceedings AFIPS*, 1972

(43) FRANK, H., "Research in Store and Forward Computer Networks," NTIS AD-757 090 Network Analysis Corporation, December 1972


(54) KREUZBERGER, H., "Profitability of Computer Networks," 1st European Workshop on Computer Networks, Arles, 1973


(59) CHANDRA, A.N., "Some Considerations in the Design of Homogeneous Distributed Databases," IBM Research, RC. 4125, 1972

(60) MULLERY, A.P., "The Distributed Control of Multiple Copies of Data," IBM Research, RC. 3642, December 1971


(64) BELFORD, G.G., P.M. SCHWARTZ, and S. SLVIZER, "The Effect of Backup Strategy on Database Availability," CCTD – WAD Doc. 6501, Center for Advanced Computation, University of Illinois at Urbana/Champaign, February 1976

(65) BELFORD, G.G. et al., "Initial Mathematical Model Report," CAC Doc. 69, University of Illinois at Urbana/Champaign, August 1975


(70) LEVIN, K.D., "Organizing Distributed Databases in Computer Networks," Ph.D. Dissertation, University of Pennsylvania, 1974


(77) CHAMPINE, G.A., "Six Approaches to Distributed Databases," Datamation, May 1977


CHAPTER THREE
DESIGN AND OPTIMIZATION OF
DISTRIBUTED INFORMATION SYSTEMS

I. Introduction

In the preceding chapter, we have shown that, due to the advances in computer network technology and the steadily decreasing cost of hardware, distributed information systems become a potential alternative to centralized systems. While many business firms still prefer the services of autonomous centralized systems, there is an increasing tendency to re-examine this concept (1). Various factors that may favor a shift towards distributed systems, were examined in Chapter Two.

In distributed information systems, research problem areas can be identified along the lines illustrated in Figure 15. In each of the problem areas, one can identify problems of optimal allocation of resources and problems of making the allocated resources available operationally to the distributed users. Examples of resource allocation problems are processing capacity assignments (2), (3) or optimal file assignments. Examples of operational problems are data transfer of software (4), (5), network control and deadlock problems (6). A large survey of this literature was presented in Chapter Two.

The focus of this research is on the distributed system configuration and the problem of optimal allocation of databases and programs over a network (illustrated by the shaded area in Figure 15). Although there is a great deal of research related to "distributed systems," most
FIGURE 15. Areas of Research (Adapted from (10))
of them addressed specific and separate issues such as network topology, file allocation, etc. Besides, most of the models developed so far are too simple, therefore hardly applied. Very often, no depth discussion of the methods for solving them is provided.

Our approach is unique in the sense that the model developed is more complete than most of the models used in the past. The detailed model developed includes computation power allocation, databases allocation, link capacities, message routing and program sharing. Network topology, communication channels capacity, size of computing hardware, and pricing schemes are interrelated in an optimal design. The pricing schemes affect the access request patterns, the distribution of the databases and the flow patterns of the communication lines. These factors determine the capacity requirements which, in turn, determine the cost of the network.

The purpose of this chapter is, therefore, to provide a general model for the optimization of distributed databases systems and computer networks. A general distributed system is regarded as:

1. a collection of computers of different capacities, located at various nodes of a network
2. a set of communication lines interconnecting the computers of the network
3. one or several databases and programs attached to the nodes of the network

Each of the computer systems possesses facilities for managing the data (store, retrieve, or process the databases). Every user of the network
is able to communicate with all other users. He may obtain information stored at any database of the network.

Most of the studies mentioned in Chapter Two assume that at each node of the network, a computer of unlimited capacity is available. Most of the models assumed also a fixed network topology. Streeter (12) relaxed the latest assumption. He provides a very simple model which determines the optimal number of computing facilities. He uses a cost function taking into account the economy of scale. The cost function is expressed in terms of the square root of the number of computing facilities. The main result obtained is that a company with user locations within about a 1000 mile radius, is optimally served by one or two computing facilities. The main disadvantage of Streeter's work is that he did not take into account the problem of file allocation. In addition, he assumed a fully connected network. His model does not determine the optimal location of computers. Finally, he ignores the case of heterogeneous networks.

Modiano (13) extended Casey's model (9). He presented a model of a computer network, which includes computation power allocation, link capacities and files allocation. The main disadvantage is that his model assumes independency between files and programs, thus not solving the case of heterogeneous computer networks. In addition, no consideration was given to the issue of optimal routing disciplines. Finally, no operational solution was provided.

Recently, Chang (3) developed a model for distributed computer systems design. It attempted to encompass both the hardware viewpoint
and the software viewpoint. But his model fails to take into account the issues of routing disciplines and diverse factors associated with distributed databases. Finally, his solution is a heuristic not necessarily leading to an optimal solution. To our knowledge, this model is the first to take into account most of the aspects of computer networks and distributed database systems. The assumption is, by encompassing all the aspects of distributed systems, we can attain a better configuration. This is fundamentally different from the previous approaches which dealt only with one aspect of the problem.

This chapter is organized as follows: In Section II, we present the global model for distributed information systems. It consists of a mathematical programming model with a nonlinear objective function and binary variables. Its purpose is to determine the optimal configuration without any assumption about the network topology.

In Section III, we will show how to derive from the model developed in Section II, a model for distributed database systems. To do so, we will explicitly assume the existence of the computer network, but try to determine the optimal distributed database systems. This is useful for organizations having access to a computer network but without program and data sharing. Finally, in Section IV, we will discuss some possible extensions of the models.
II. The Global Model

In this section we consider the optimization aspects of distributed information systems. A typical distributed database system and computer network is indicated in Figure 16.

![Diagram of a Distributed Database System and Computer Network]

FIGURE 16. A Distributed Database System and Computer Network

It should be noted that for a distributed system (like the one depicted in Figure 16):

(i) The computers at the nodes may be of different capacities.
(ii) The databases may be identical or different (each one handling information related to warehouses, plants, or personnel), and they may have different sizes.
(iii) The communication lines may be of different capacities.
(iv) The programs operating on the databases are specific to each computer system and may be of different lengths.

(v) No specific assumptions are made about optimal message routing.

A. Assumptions and Modelling Aspects

In this section, we consider a distributed information system. The set of nodes is denoted by $K$. At each node, a computer may be installed. The computer can be a mini computer or any other computer system. The set of computer types is denoted by $M$. It represents the type of computers that can be used in the network. For example, $M$ can be the set $[\text{IBM 370/168, IBM 370/158, CDC 6600, PDP11, PRIME 400, etc.}]$. The capacity of computer $m$ is denoted by $k_m$. The capacity of a computer is defined by its throughput. In other words, the capacity of a computer is defined as being equal to the number of transactions that can be processed for a given unit of time. (For example, an IBM 370/168 can process up to 20,000 transactions per hour.) The cost of a computer $m$ is denoted by $C_m$.

At each node of the network, one or several databases may be installed. We do consider traditional files as well as databases where all the data used by a specific node are included, which may be a manufacturing unit of a large geographically dispersed company. The set of databases is denoted by $N$. Two costs are associated with each database: (a) the set-up and operating cost, which is denoted by $D_d$ for database $d$. By set-up cost, we mean the fixed cost that occurs before storing and using the database (for example, coding and punching costs). (b) The storage cost, which is denoted by $C_{kd}$, the
storage cost per unit length of database \( d \) at node \( k \). The storage cost can be the same at each node of the computer network when the types of storage devices used to store the database are the same. On the contrary, when the types of storage devices are different, the storage costs can be different. For example, the storage cost of database using tapes, can be less expensive than the storage cost of the same database, using disks.

Since we consider a heterogeneous distributed system, we should consider a set of programs devoted to the use of the databases. The set of programs is denoted by \( P \). The length of program is \( L_p \), and the cost of storage per megabyte of program \( p \) at node \( j \) is denoted by \( S_{jp} \).

The computers of the network and their associated databases are interconnected by communication lines. The set of communication lines is denoted by \( C \). It represents the types of communication line that can be used in the network. The capacity of a communication line type \( c \) is denoted by \( Q_c \). For example, \( Q_c \) can be the following communication lines [50 bps, 600 bps, 1200 bps, half-duplex, full-duplex, etc., ]

Two different costs are associated with communication line installations: (a) \( B_c \) is the cost of installing a line of capacity \( c \). This is a fixed cost, independent of the distances between the nodes; (b) \( B_{lc} \) is the variable cost of installing a communication line of capacity \( c \), depending on the distances between nodes \( i \) and \( j \). These distances are represented by the parameter \( D_{ij} \). The unit used is kilometer.

There are two other communication costs that occur: (a) \( Q_{ij} \) represents the communication cost per query unit from \( i \) to \( j \); (b) \( U_{ij} \)
represents the communication cost per update unit from \( i \) to \( j \). It is obvious that \( Q_{ij} \) and \( U_{ij} \) are dependent upon the distances between nodes \( i \) and \( j \). \( Q_{ij} \) can be different from \( U_{ij} \) since the updates can be done during the night when the communication costs are lower.

Since we are studying a general heterogeneous system, we assume dependence between databases and programs. Therefore, in order to process a given transaction, both the relevant program and the relevant database must be accessed. A given transaction (query or update) to database \( d \) located at node \( k \), from a user located at node \( i \), should be first processed by the relevant program \( p \) that may reside at node \( j \). As a consequence, the transaction from node \( i \) should be first routed to node \( j \). During its processing by program \( p \), an access request to database \( d \) located at node \( k \), might be issued by the program. This access request to database \( d \) is originated at node \( j \). Therefore, different assignments of programs will yield different distributions of request rates to the databases. As a consequence, there will be two different communication costs for queries and updates: a communication cost for queries and updates from users to programs and a communication cost for queries and updates from programs to databases.

In our model, we explicitly take into account "the return flow of information." By this, we mean the flow of information that a user in the network gets as a response to its queries. There are two types of return flow of information:

\[
(a) \quad \gamma_{ij} = \frac{\text{estimated ratio of size of response}}{\text{size of request from a user situated at node } i}
\]
to a program located at node \( j \).

(b) \( \gamma_{jk} \) = the estimated ratio from a program located at node \( j \) to a database situated at node \( k \).

Finally, since we assume dependency between databases and programs, we have to define the concept of expansion factor. Let us call \( l_1 \) the length of the query issued from users to programs, and \( l_2 \) the length of the resultant query issued from programs to databases, then

\[ \alpha = \frac{l_2}{l_1} \]

is the expansion factor for the queries. Similarly, \( \beta \) will be the expansion factor for updates.

We do not define an expansion factor for responses back to users since, in general, the response given by the databases to programs is the same as the one transmitted by programs to users.

Our model will provide us with an optimal distributed information system if:

(i) an optimal allocation of computers over the network is specified. This includes the determination of the capacity of the computers allocated to the network.

(ii) an optimal allocation of the databases is obtained. We may have at a specific node one or more databases.

(iii) an optimal allocation of the programs is defined.

(iv) an optimal allocation of the communication lines between the nodes is specified, including the capacities of such communication lines.

(v) an optimal message routing discipline is obtained. This
includes the routing disciplines for messages flowing between users and programs and for messages flowing between programs and databases.

Our problem is to determine these optimal allocations at minimum cost. The costs considered in our model are:

- cost of computers (equipment cost)
- cost of databases (set-up and operating costs)
- cost of communication lines
- storage costs of databases
- storage costs of programs
- communication cost of queries and updates from users to programs
- communication costs of queries and updates from programs to databases.

All the costs that are considered in our model are the costs per unit of time. A reasonable unit of time can be a month. One can argue about the time period. But, taking a month as a time period may lead to a good estimation of the costs. The cost per month of a given equipment can be obtained by dividing the purchase cost of the equipment by its lifetime. Of course, other units of time periods are relevant and can be applied in this model.

We emphasize the facts that:

1. We do not assume the existence of a fixed network topology.
2. We do not assume the existence of a computer of unlimited
capacity at each node of the network.

3. We do not assume the existence of a fully connected network.

4. We do not assume independency between databases and programs.

The assumptions of this model are the existence of several geographically dispersed users with known query and update access requests to the database(s).

In the remainder of this thesis, subscript \( i \) indicates a user node, subscript \( j \) indicates a node where a program is located, and subscript \( k \) indicates a node where a database resides. Let us now exhibit the objective function and the set of constraints of our model.

### B. The Objective Function

(a) **Computer Costs**

Computer costs are assumed to be a function of their capacities. The capacity of a computer is expressed in terms of throughput.

Let us define:

\[
Y^m_k = \begin{cases} 
1 & \text{if a computer of capacity } k^m \text{ is allocated to node } k \\
0 & \text{otherwise} 
\end{cases}
\]

\[
C^m = \text{cost per unit of time of computer } m \text{ which capacity is } k^m
\]
The total cost of computer equipment is:

\[
\text{COST 1} = \sum_{m,k} C_m y_{k}^m \quad k \in K, \text{ and } m \in M
\]

(b) Database Costs

Database costs may be estimated by the set-up and operating costs. It is a function of the length of the databases. Let

\[
x_k^d = \begin{cases} 
1 & \text{if database } d \text{ is allocated to node } k \\ 
0 & \text{otherwise} 
\end{cases}
\]

and

\[
D_d = \text{cost of database } d \text{ which is considered equal to the set-up cost and the operating cost}
\]

The total cost of databases is:

\[
\text{COST 2} = \sum_{d,k} D_d x_k^d \quad k \in K, \text{ and } d \in N
\]

(c) Communication Lines Installation Cost

Communication lines connecting the nodes of the network may be of different capacities and different speeds. It seems reasonable to take the cost of communication lines as a function of the distance between the nodes and of the capacities to which we add set-up cost. Therefore, the communication lines cost is:
\[
\text{COST 3} = \sum_{i,j,c} \left[ B_{c} + (B_{c} D_{ij}) \right] L_{ij}^{c} \\
\text{where}
\]

\[
B_{c} = \text{installation cost of a communication line of capacity c (variable cost)}
\]

\[
B_{c} = \text{fixed cost of the communication line c}
\]

\[
D_{ij} = \text{distance between nodes i and j}
\]

\[
L_{ij}^{c} = \begin{cases} 
1 & \text{if a communication line of capacity } Q_{c} \text{ connects node i to node j} \\
0 & \text{otherwise}
\end{cases}
\]

(d) **Storage Costs**

We differentiate between the storage cost of databases and the storage cost of programs. A difference has to be made between programs and databases, in a heterogeneous distributed computer system.

(d.1) **Storage Cost of Databases**

The storage cost of databases is assumed to be a linear relationship. It is a function of the length of the database.

Let

\[
x_{k}^{d} = \begin{cases} 
1 & \text{if a copy of database } d \text{ exists at node } k \\
0 & \text{otherwise}
\end{cases}
\]

\[
C_{kd} = \text{storage cost per unit of time of database } d \text{ existing at node } k
\]
The total storage cost is:

\[ \text{COST 4} = \sum_{d,k} C_{kd} X_k^d \quad k \in K, \ d \in N \]

(d.2) **Storage Cost of Programs**

We make the same assumptions as above. Therefore, the total storage cost of programs is:

\[ \text{COST 5} = \sum_{j,p} S_{jp} Z_p^j \quad j \in J = K, \ p \in P \]

where:

- \( S_{jp} \) = storage cost per unit of time of program \( p \) at node \( j \)
- \( Z_p^j = \begin{cases} 1 & \text{if a copy of program } p \text{ is stored at node } j \\ 0 & \text{otherwise} \end{cases} \)

(e) **Communication Costs**

For the communication costs, we differentiate between the following costs:

(a) communication cost of queries from users to programs
(b) communication cost of updates from users to programs
(c) communication cost of queries from programs to databases
(d) communication cost of updates from programs to databases

The reason is that there is a possibility that transactions from
a given node, that are processed by the same program but require access to different databases, can be routed to different nodes at which copies of the program are stored. Levin (10) gives the following example in a hypothetical network:

![Diagram](image)

**FIGURE 17. A Hypothetical Network**

In the above example, part of the queries from node 1 require access to database 1, the other part requires access to database 2, both parts should be processed by program p. Database 1 is stored at node 5 and database 2 at node 4, compatible copies of program p are stored at node 2 and node 3. The variable $X_{ijp}$ permits us to route the queries to database 1 through node 3 and the queries to database 2 through node 2. When a return flow of information occurs, its communication cost will be included.

(e,1) **Communication Cost of Queries from Users to Programs**

This communication cost is a function of the query traffic between nodes. Let us define:
\[ Q_{ip}^d = \text{query traffic from node } i \text{ to database } d \text{ via program } p \]
\[ Q_{ij} = \text{communication cost per query unit from node } i \text{ to node } j. \]

Therefore, \( Q_{ij} \) represents the cost of a transaction originating at node \( i \) and sent to a program located at node \( j \). Although it is desirable to have \( Q_{ij} \) as a function of the communication line capacity, we will consider it only as a function of the distance between nodes \( i \) and \( j \) in order not to complicate the model.

\[ X_{ijp}^d = \begin{cases} 1 & \text{if transactions from node } i \text{ to database } d \text{ are routed to program } p \text{ located at node } j \\ 0 & \text{otherwise} \end{cases} \]

The total communication cost of queries from nodes to programs is:

\[
\text{COST 6} = \sum_{i,j,p,d} Q_{ip}^d Q_{ij} X_{ijp}^d (1 + \gamma_{ij}) \\
i \in I = K, j \in J = K, p \in P, d \in N
\]

where:

\[ \gamma_{ij} = \text{estimated ratio of size of response to size of query from a user located at node } i \text{ to a program located at node } j. \]

Notice that this ratio can take into account the expected ratio of new applications.

(e.2) Communication Cost of Updates from Users to Programs

We make the same assumptions as above. Therefore, the total cost of updates from nodes to programs is:

\[
\text{COST 7} = \sum_{i,j,p,d} U_{ip}^d U_{ij} X_{ijp}^d \\
i \in I = K, j \in J = K,
\]
where:

\[ U^d_{ijp} = \text{update traffic from node } i \text{ to database } d \text{ via program } p \]

\[ U_{ij} = \text{communication cost per update unit from node } i \text{ to } j \]

\[ X^d_{ijp} = \begin{cases} 
1 & \text{if transactions from node } i \text{ to database } d \text{ are} \\
0 & \text{routed to program } p \text{ located at node } j \\
\end{cases} \]

(e.3) Communication Cost of Queries from Programs to Databases

As in (e.1) and (e.2), we consider the cost of queries from programs to databases, as a function of the query traffic to databases, processed at different nodes. Let us define:

\[ \lambda_{jp} = \sum_i Q^d_{ip} X^d_{ijp} = \text{query traffic to database } d \text{ processed} \]
\[ \text{at node } j \text{ by program } p \]

\[ X^d_{jkp} = \begin{cases} 
1 & \text{if there are transactions from node } j \text{ to database} \\
0 & d \text{ which is located at node } k \text{ and which is processed} \\
& \text{by program } p \\
\end{cases} \]

The total communication cost is, therefore:

\[ \text{COST 8} = \sum_{j,k,p,d} \alpha \lambda_{jp} Q^d_{jk} X^d_{jkp} (1 + \gamma_{jk}) \]

where:

\[ \alpha = \text{expansion factor for query messages} \]
\( Q_{jk} = \text{communication cost per query unit from node } j \text{ to node } k \)

\( Y_{jk} = \text{estimated ratio of size of response to} \)
\( \text{size of query for requests from program located at node } j \)
\( \text{to database situated at node } k \)

\text{(e.4) Communication Cost of Updates from Programs to Databases}

We make the same assumption as above, and we define \( \mu_{jpd} \) as the update traffic to database \( d \) processed at node \( j \) by program \( p \):

\[
\mu_{jpd} = \sum_{i} U_{i}^{d} X_{ijp}^{d}
\]

The total communication cost of updates from programs to databases is:

\[
\text{COST} = \sum_{j,k,p,d} \beta_{jpd} U_{jk} X_{k}^{d}
\]

where:

\( \beta = \text{expansion factor for update} \)

\( U_{jk} = \text{communication cost per update unit from node } j \text{ to} \)
\( \text{node } k \)

\( X_{k}^{d} = \begin{cases} 
1 & \text{if a database } d \text{ is located at node } k \\
0 & \text{otherwise}
\end{cases} \)
To facilitate the interpretation of the expression of cost 9, let us decompose it.

\[ \sum_i u_{dp}^d \cdot x_{ijp} \]

represents the updating traffic to database \( d \) via program \( p \) flowing from node \( i \) to node \( j \). By taking the sum over \( k \), we include all the updating traffic to all the copies of the database. By also summing over \( j \), we include all the nodes updating the multiple copies. Therefore,

\[ \beta u_{jpd}^d \cdot u_{jk} \cdot x_k^d \]

represents the updating traffic to database \( d \) flowing from \( j \) to node \( k \). Finally,

\[ \sum_{j,k,p,d} \beta u_{jpd}^d \cdot u_{jk} \cdot x_k^d \]

represents the communication cost of updates from programs to all databases (i.e. all the copies of the database).

The overall objective function is, by summing all the costs, equal to:

\[ \text{COST} = \sum_{i=1}^{9} \text{COST}_i \]

C. The Constraints

A distributed system is feasible if the following constraints are satisfied:
(a) **Existence of Communication Lines**

In order to have a feasible solution, there must be at least one communication line between a user and the other nodes of the network. This condition is met if:

\[ \sum_{j} L_{ij}^c \geq 1, \quad \forall i \in I = K \text{, and } c \in C \]

(b) **Transactions with Defined Routes**

We have to assure that every transaction to every database via its related program and from every node will have a predefined route. Therefore:

\[ \sum_{j} X_{ijp}^d = 1, \quad \forall \in I = K \text{, and for all } p \in P \text{ and } d \in N \text{ which are related to each other.} \]

\[ \sum_{k} X_{jkpd}^e = z_{j}^p, \quad \forall j \in J = K \text{ which has } p \in P \text{, and for all } d \in N \text{ which are related to } p \in P \]

Notice that the first equation allows us to have exactly one route from a user to a program. The second equality permits us to exclude all the \( X_{jkpd} \) which do not have a program \( p \) stored at node \( j \).

(c) **Residency of Databases and Programs in Accordance with the Defined Routes**

We must assure residency of the appropriate databases and programs in accordance with the defined routes. In other words, if a
route is defined for transactions to a particular database or program, this database or program should reside in the corresponding node. Therefore:

\[ \sum_{i,j,p} x_{ijp}^d \leq \sigma \cdot x_j^p \quad \forall p \in P, \quad d \in N \]

\[ \sum_{j,k,p,d} x_{jkpd}^d \leq \sigma \cdot x_{k}^d \quad \forall k, \text{ and for all } p \in P \text{ related to } d \in N \]

where \( \sigma \) is equal to the number of nodes of the network.

(d) Computation Capacity

We shall assure that the total processing requirements of all transactions should not exceed the computer's capacity at the nodes of the network:

\[ \sum_{i,j,p,d} \alpha_{ip} q^d x_{ijp}^d x_{jkpd} + \sum_{i,p,d} \beta_{ip} x_{ip}^d + \sum_{i,p,d} (q_{ip}^d + u_{ip}^d) x_{ikp}^d \leq \sum_{k} v_{m}^m \]

\[ \forall i \in I = K, \quad \forall j \in J = K, \quad \forall p \in P, \quad \forall d \in N, \quad \text{and for all } k \in K \]

The first term in the left hand side of the inequality gives us the sum of all the queries to the databases located at node \( k \). The second term gives us the sum of all the updates to the databases located at node \( k \). The third term gives us the sum of all the queries to the programs located at node \( k \). The right hand side of the inequality represents the capacity of the computers allocated to node \( k \). The inequality states that the total processing requirements of all transactions should not exceed the capacity of the computers allocated to the node.
(e) **Existence of a Database with Respect to Computers**

We shall assure that databases may only exist at nodes where there are computers:

\[
X^d_k \leq \sum_m y^m_k \quad \text{for all } k \in K, \quad d \in N
\]

(f) **Existence of a Program with Respect to Computers**

We shall assure that programs may only exist at nodes where there are computers:

\[
Z^p_j \leq \sum_m y^m_j \quad \forall j \in J = K, \quad p \in P
\]

(g) **Communication Line Capacity Constraint**

We must assure that the total query and update traffic between node \(i\) and \(j\), does not exceed the link capacity:

\[
\sum_{r,p,d} Q^d_{rp}X^d_{rijp} + \sum_{r,p,d} Q^d_{rp}X^d_{jipd} + \sum_{r,p,d} U^d_{rp}X^d_{rijp}\]

\[
\sum_{r,p,d} U^d_{rp}X^d_{rijp} + \sum_{p,d} (Q^d_{ip}+U^d_{ip}) X^d_{ijp} + \sum_{p,d} (Q^d_{jp}+U^d_{jp}) X^d_{jip} \leq \sum_{c} L^c_{ij}
\]

for all \(i \in I = K\) and \(j \in J = K\) with \(j > i\).

The first term of the LHS represents the query traffic between nodes \(i\) and \(j\) when we have a database at node \(j\) and a program at node \(i\). The second term represents the query traffic between nodes \(i\) and \(j\) when we have the database at node \(i\) and the program at node \(j\). The third and fourth terms of the LHS are respectively the same as the first and the
second terms but for updates. The fifth term represents the query traffic between nodes \(i\) and \(j\) when we have both the database and the program at node \(j\). The sixth term represents the query traffic in the case that we have both the database and program stored at node \(i\). Finally, the RHS represents the capacity allocated to link \(i\) and \(j\). The overall inequality forces the total traffic between nodes \(i\) and \(j\) not to exceed the link capacity.

(h) **Existence of Database and Program**

To assure a feasible solution, there must be at least one copy of each database and program:

\[
\sum_{k} x_{k}^{d} \geq 1, \quad \forall d \in N
\]

\[
\sum_{j} z_{j}^{p} \geq 1, \quad \forall p \in P
\]

There is no need to exhibit explicitly the same constraint for computers, since it is forced by constraints (e) and (f).

(i) **Binary Constraints**

All the decision variables must be binary variables. Therefore:

\[
y_{k}^{m} = [1 \text{ or } 0] \quad \text{for all } k \in K \text{ and } m \in M
\]

\[
x_{k}^{d} = [1 \text{ or } 0] \quad \text{for all } k \in K \text{ and } d \in N
\]
D. Final Formulation of the Problem

Based on the costs discussed in Section (B) and the constraints discussed in Section (C), we can formulate the problem as follows:

$$\min \{ \sum_{m,k} c_{ym} y_{k}^m + \sum_{d,k} D_{d} x_{k}^d + \sum_{i,j,c} \left[ B_{ij} + (B_{ij}) \right] L_{ij}^c \}
$$

$$+ \sum_{d,k} C_{kd} x_{k}^d + \sum_{j,p} S_{jp} z_{j}^p + \sum_{i,j,p,d} Q_{ij}^d Q_{ij} x_{ijp}^d \left( 1 + \gamma_{ij} \right)$$

$$+ \sum_{i,j,p,d} U_{ijp} U_{ij} X_{ijp}^d + \sum_{j,k,p,d} \alpha_{jk} \lambda_{jd} Q_{jk} X_{jkp}^d \left( 1 + \gamma_{jk} \right)$$

$$+ \sum_{j,k,p,d} \beta_{jd} U_{jpd} U_{jk} X_{jkd}^d \}$$

subject to:

$$\sum_{j} L_{ij}^c \geq 1, \quad \forall i = 1, 2, \ldots, I = K, \text{ and } \forall c = 1, 2, \ldots, C$$

$$\sum_{j} x_{ijp}^d = 1, \quad \forall i = 1, 2, \ldots, I = K, \forall p = 1, 2, \ldots, P, \forall d = 1, 2, \ldots, N$$
\[
\sum_{j,k,p,d} x_{jkd} = z_j^p \quad \forall k = 1,2,\ldots,K, \quad \forall d = 1,2,\ldots,N \quad \forall p \in P
\]

\[
\sum_{j} x_{jdp} \leq \sigma \cdot z_j^p \quad \forall j = 1,2,\ldots,J=K, \quad \forall p = 1,2,\ldots,P, \quad \forall d = 1,2,\ldots,N
\]

\[
\sum_{j} x_{jkd} \leq \sigma \cdot x_{k}^d \quad \forall k = 1,2,\ldots,K, \quad \forall d = 1,2,\ldots,N \quad \forall p \in P
\]

\[
\begin{align*}
\sum_{i,j,p,d} & a_{ijp} x_{ijpd} + \sum_{i,p,d} \beta_{ip} x_{ip}^d + \sum_{i,p,d} (Q_{ip} + U_{ip}) x_{ip}^d - \sum_{k} \sum_{m} y_{km}^m \\
& \psi_{i} \in I = K, \quad \forall j \in J = K, \quad \forall p \in P, \quad \forall d \in N, \text{ and for all } k \in K
\end{align*}
\]

\[
x_k^d \leq \sum_{m} y_{km}^m, \quad \forall k = 1,2,\ldots,K, \quad \forall d = 1,2,\ldots,N
\]

\[
z_j^p \leq \sum_{m} y_{jm}^m, \quad \forall j = 1,2,\ldots,J=K, \quad \forall p = 1,2,\ldots,P
\]

\[
\begin{align*}
\sum_{r,p,d} & \alpha_{rp} (1+\gamma_{rij}) x_{rijp} + \sum_{r,p,d} \alpha_{rp} (1+\gamma_{rij}) x_{rijpd} + \sum_{r,p,d} \beta_{rp} x_{rp}^d x_{rd}^d \\
& + \sum_{r,p,d} \beta_{rp} x_{rp}^d x_{rd}^d + \sum_{r,p,d} \beta_{rijp} (1+\gamma_{rij}) + U_{ijp} x_{ijp}^d + \sum_{r,p,d} \beta_{rijp} (1+\gamma_{rij}) + U_{ijp} x_{ijp}^d - \sum_{c} \sum_{i,j} c_{ij}^c
\end{align*}
\]

for all \( i \in I = K \) and \( j \in J = K \) with \( j > i \)

\[
\sum_{k} x_k^d \geq 1, \quad \forall d \in N \quad \sum_{j} z_j^p \geq 1, \quad \forall p \in P
\]

where all the non-decision and the decision variables are described below:
Non-Decision Variables

\( K \) set of nodes

\( M \) set of computers

\( k_m \) capacity of computer type \( m \)

\( C_m \) cost of computer type \( m \)

\( N \) set of databases

\( D_d \) set-up cost of database \( d \)

\( C_{kd} \) storage cost of database \( d \) at node \( k \)

\( P \) set of programs

\( Q_c \) capacity of communication line type \( c \)

\( S_{jp} \) storage cost of program \( p \) at node \( j \)

\( C \) set of communication lines

\( B_{lc} \) fixed cost of communication line \( c \)

\( \gamma_{ij} \) estimated ratio of size of response to size of query requests from node \( i \) to a program located at node \( j \)

\( \gamma_{jk} \) estimated ratio of size of response to size of query from requests from program located at node \( j \) to database situated at node \( k \)

\( \alpha \) expansion factor for query message

\( \beta \) expansion factor for update message

\( \lambda_{jpd} \) query traffic to database \( d \) processed at node \( j \) by program \( p \)

\( \mu_{jpd} \) update traffic to database \( d \) processed at node \( j \) by program \( p \)
$B_c$ installation cost of a communication line of capacity $c$ (variable cost)

$Q_{ij}$ communication cost per query unit from $i$ to $j$

$U_{ij}$ communication cost per update unit from $i$ to $j$

$D_{ij}$ matrix of distances between $i$ and $j$

$q^d_{ip}$ query traffic from node $i$ to database $d$ via program $p$

$u^d_{ip}$ update traffic from node $i$ to database $d$ via program $p$

**Decision-Variables**

$$y^m_k = \begin{cases} 1 & \text{if a computer of capacity } k \text{ is allocated to node } k \\ 0 & \text{otherwise} \end{cases}$$

$$x^d_k = \begin{cases} 1 & \text{if a copy database } d \text{ is allocated to node } k \\ 0 & \text{otherwise} \end{cases}$$

$$L^c_{ij} = \begin{cases} 1 & \text{if a communication line of capacity } Q_c \text{ connects node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

$$z^p_j = \begin{cases} 1 & \text{if a copy of program } p \text{ is stored at node } j \\ 0 & \text{otherwise} \end{cases}$$

$$x^d_{ijp} = \begin{cases} 1 & \text{if transactions from node } i \text{ to database } d \text{ are routed to node } j \text{ via program } p \\ 0 & \text{otherwise} \end{cases}$$
\[
X_{jkpd} = \begin{cases} 
1 & \text{if transactions from node } j \text{ to database } d \text{ are routed} \\
& \text{to node } k \text{ and are processed by program } p \\
0 & \text{otherwise}
\end{cases}
\]

E. Model Type

Due to the fact that we have in the objective function and the constraints:

- non-linearities due to communication costs of queries and updates from programs to nodes
- integer variables \(y^m_k, x^d_k, L^c_{ij}, z^p_j, X^d_{ijp}, X^d_{jkpd}\)

therefore, the model is an integer non-linear programming problem (INLP).

Due to the type of nonlinearities that we have in the objective function, the problem is non convex. Finally, due to the fact that a typical problem may have between 20 to 1000 variables, the model is a large scale one.

F. Relevances of the Model

(i) Relevance to the Computer Network Design Problem

Besides its relevance to the design issue of distributed information systems, the model developed can be applied by a designer of a computer network only. It will allow the designer to define the network topology, and to assign communication lines capacities. The designer can use it as a technique to produce minimal cost designs. This can be done by ignoring all the variables directly related to the distributed database systems aspects (i.e., \(x^d_k, z^p_k, X^d_{ijp}, X^d_{jkpd}\)) and their corresponding costs components.
(ii) Relevance to the Centralization – Decentralization Issue

Until recently, the management-oriented literature focused only on the respective advantages and disadvantages of centralized – decentralized systems. The only comprehensive model which includes guidance for managers dealing with the centralization – decentralization issue, was developed by Rockart et. al. (14). One important aspect that this model omitted is how to determine the optimal configuration. Our model can be used as a quantitative tool to answer this question. Since centralized systems are subsystems of distributed systems, by solving our model we can determine a range of configurations from fully centralized to fully decentralized systems. This aspect is developed in Chapter Six. The model can also be used to evaluate configurations and to help managers to determine the configuration which minimized the tangible costs. It is a generalization of the model developed by Chen et al. (15).

(iii) Relevance to the Distributed Databases Design Problem

This model can be viewed (and used) as an extension of Chen's work (16) to the case of a computer network. Our distributed model can be interpreted as a combination of Chen's work with various network flow problems. Besides, the model can be used to minimize the operating cost of distributed databases, shared by a community of users interconnected through a computer network. In particular, it can be used to find the optimal database/program locations in a computer
network. Finally, it takes into account the structure of both heterogeneous and homogeneous computer network. This aspect is precisely the subject of Section III of the chapter.

III. The Optimal Distributed Database Systems Model

For some organizations, the topology of the distributed network exists already, but the issue of optimal distributed databases is still to be solved.

For example, in a network it can be the case that two or more copies of a database are cheaper than a single one. This can be due to the fact that by going to multiple copies, we can reduce query and update transmission costs. This savings can offset the additional costs incurred by the additional copy (such as storage cost). Therefore, an important question is where in the network to store the databases and the programs operating on them? This distinction between the optimal locations of databases and programs has been discussed in the preceding section, and proved to be important when one wants to take into account both homogeneous and heterogeneous computer networks.

To derive the formulation of an optimization model for distributed database systems, it is only necessary to make assumptions about the network topology, and the structure of communication lines. In other words, if we assume that a fully connected computer network exists, we will be investigating only the problem of optimal location of databases and programs. This can be the case of organizations sharing a common network or of existing networks without database sharing.
Using our preceding model, we can easily derive the new formulation. Since the network topology and the structure of the communication lines are assumed given, the variables $y^m_k$ and $L^c_{ij}$ related to computers allocation and communication lines assignment, are known. Therefore, the expressions of COST 1, and COST 3 can be omitted. To make possible a comparison with past models, let us assume that organizational and set-up costs of database are negligible. Therefore, the parameters $D_d$ are negligible, and so is the expression of COST 2.

The only questions to be answered by the model are:
- how many copies of databases are needed, and where to put them
- what is the optimal location of the programs
- which optimal message routing discipline to adopt

As for the general model, we assume dependency between files and programs. The objective function to be minimized is then:

$$ F = \sum_{d,k} C_{kd} X^d_{k} $$

storage cost of database

$$ + \sum_{j,p} S_{j,p} Z^P_{j} $$

storage cost of programs

$$ + \sum_{i,j,p,d} Q^d_{i,p} Q^d_{i,j} X^d_{i,j,p} (1 + \gamma_{ij}) $$

communication cost of queries from nodes to programs

$$ + \sum_{i,j,p,d} U^d_{i,p} U^d_{i,j} X^d_{i,j,p} $$

communication cost of update from nodes to programs

$$ + \sum_{j,k,p,d} \alpha^\lambda_{j,p,d} Q^d_{j,k} X^d_{j,k,p} (1 + \gamma_{jk}) $$

communication cost of queries from programs to databases
\[ + \sum_{j,k,p,d,jpd,jk} \beta_{j,k,p,d,jpd,jk} X_{j,k}^d \]

communication cost of updates from programs to databases

Subject to the following constraints:

- To assure the existence of a feasible solution, there must be at least one copy of each database and program, i.e.,

\[ \sum_{k} X_{j,k}^d \geq 1, \forall d \in N \]

\[ \sum_{j} Z_{j,p}^d \geq 1, \forall p \in P \]

- To assure that every transaction to every database via every program which is related to it and from every node, will have a defined route

\[ \sum_{j} X_{i,j,p}^d = 1, \forall i \in I=K, p \in P, d \in N \]

\[ \sum_{k} X_{j,k,p,d,jk} = Z_{j,p}^d, \forall j \in J=K, d \in N, p \in P \]

- To assure residency of the appropriate database and programs in accordance with the defined routes:

\[ \sum_{i} X_{i,j,p}^d \geq \sigma \cdot Z_{j,p}^d, \forall j \in J=K, p \in P, d \in N, \sigma = \text{number of nodes of the network} \]

\[ \sum_{j} X_{j,k,p,d,jk} \geq \sigma \cdot X_{j,k}^d, \forall k \in K, d \in N, p \in P \]

- To assure that program p can reside only in modes at which
it can be processed.

\[ Z^p_j = 0 \quad \forall j \notin I_p, \quad p \in P \text{ where } I_p \text{ is the set of nodes where program } p \text{ can be processed.} \]

- Binary constraints:

\[
X^d_k = [0 \text{ or } 1] \quad \text{for all } k \in K \text{ and } d \in N
\]

\[
Z^p_j = [0 \text{ or } 1] \quad \text{for all } j \in J=K \text{ and } p \in P
\]

\[
X^d_{ijp} = [0 \text{ or } 1] \quad \text{for all } i \in I=K, \quad j \in J=K, \quad p \in P, \quad d \in N
\]

\[
X^d_{jkpd} = [0 \text{ or } 1] \quad \text{for all } j \in J=K, \quad k \in K, \quad d \in N, \quad p \in P
\]

The final formulation for the optimal distributed database system is:

\[
\min F = \sum_{d,k} C_{kd} X^d_k + \sum_{j,p} S^p_j Z^p_j + \sum_{i,j,p,d} Q^d_{ip} Q^d_{ij} X^d_{ijp}
\]

\[
(1 + \gamma_{ij}) + \sum_{i,j,p,d} U^d_{ip} U^d_{ij} X^d_{ijp} + \sum_{j,k,p,d} \alpha^d_{jkpd}
\]

\[
Q_{jk} X^d_{jkpd} (1 + \gamma_{jk}) + \sum_{j,k,p,d} \beta_{jkpd} U^d_{jk} X^d_k
\]

subject to:

\[
\sum_k X^d_k > 1, \quad \forall d \in N
\]

\[
\sum_j Z^p_j > 1, \quad \forall p \in P
\]
\[ \sum_{j} x_{ijp}^d = 1, \quad \forall i \in I = K, \quad p \in P, \quad d \in N \]

\[ \sum_{k} x_{jkpd}^p = z_j^p, \quad \forall j \in J = K, \quad d \in N, \quad p \in P \]

\[ \sum_{i} x_{ijp}^d \leq \sigma \cdot z_j^p, \quad \forall j \in J, \quad p \in P, \quad d \in N \]

\[ \sum_{j} x_{jkpd}^d \leq \sigma \cdot x_k^d, \quad \forall k \in K, \quad d \in N, \quad p \in P \]

\[ z_j^p = 0, \quad \forall j \not\in I_p, \quad p \in P \]

\[ x_k^d, \quad z_j^p, \quad x_{ijp}^d, \quad x_{jkpd}^d \text{ binary variables} \]

All the variables that appear in this model were explained in Section II-D of this chapter.

The model is similar to the Levin-Morgan Model (11), in the sense that like our model, Levin-Morgan's model determines the optimal locations of databases and programs in a computer network. As in our model, it takes into account the dependencies between databases and programs. Finally, as in our model, it determines the message routing disciplines.

However, there are some important differences between Levin-Morgan's approach and our approach. These differences will be described in Chapter Five.
IV. Possible Extensions of the Models

So far, we have dealt with very detailed models for the design of distributed systems. Although the two models presented in this chapter are reasonably complete, there are some possible extensions. They are related to the issues of access time and reliability.

(i) Access time constraints

To assure a minimum level of service, one has to introduce some access time constraints. This access time can be one of the factors taken into account when a particular communication link is chosen. For example, in the ARPA network circuits of 50 kb/seconds are used in order to achieve a delay of 0.2 seconds on communication links. In general, access time is given as an a priori design constraint. In that case, the access time constraint is introduced under the assumption that the maximum expected delay on a given communication link is given as one of the network specifications, with the possibility of taking into account the queuing of messages flowing in the network. A possible way to introduce the access time constraint is the following:

Let $T_d$ be the maximum allowable delay permitted by the management of the network to access database d. And let $t_{ij}$ be the maximum expected delay on the communication line from node i to node j. A feasible route from node i to database d at node k, via program p at node j, is a route for which $t_{ij} + t_{jk} \leq T_d$. 

In the bank example described in Chapter Five, the access time constraint will be introduced as a factor taken into account in the choice of the communication line type.

(ii) Reliability Constraints

If a particular node of the network (corresponding to the location of a user) is perceived as unreliable by the management of the network, it is usually preferable to exclude it as a potential node to locate the computer or to store the database or the program.

In the bank's example that will be studied in Chapter Five, the management of the distributed information systems explicitly excludes some possible nodes of the network, although there are potential users. The main reason invoked is the risk of having improper telephone service, and unreliable maintenance possibilities. Therefore, in order to avoid prolonged breakdown of the distributed system, the management prefers to exclude those unreliable nodes.

In fact, it is very simple to impose these constraints in our model. It is enough to exclude the concerned node from the network. Therefore, the permissible set of nodes is smaller than the one defined without security constraints, thus reducing the size of the problem.
V. Conclusions

It this chapter, we have presented two different models for the design of distributed information systems. Our global model incorporates many characteristics that have only been considered separately in the models developed in the past. As a consequence, the global model is more realistic than most of the previous models.

The distributed database systems model was derived from our global model, by assuming the existence of a network topology. Although similar to Levin-Morgan's models, it includes the return traffic from queried databases and programs. This return traffic was not included in past models for distributed database systems. This return traffic gives to our second model a realistic view of the issue. This realistic view can be evaluated only if we develop adequate techniques to solve such models. This is precisely the aim of the next chapter.
References

Chapter Three

(1) MOORE, W.M., "Study Highlights Four Areas of Distributed Processing," Computerworld, December 20-27, 1976


I. Introduction

The aim of this chapter is to present a mathematical programming algorithm in order to solve the two distributed models presented in Chapter Three. In the past, the approaches used to solve these types of models can be classified in one of the following three categories:

1. Linearization techniques [(1), (2),]
2. Branch and bound techniques [(3), (4)]
3. Heuristics techniques [(5), (6)]

Linearization techniques are used when one has to deal with nonlinearities. The main disadvantage of such techniques is the fact that they are not general enough to be applied to all varieties of nonlinearities. Besides, the techniques add more variables to the original problem, making large-scale problems very difficult (if not impossible) to solve.

Traditional branch and bound techniques were mainly used for linear programming problems. The main difficulty with branch and bound techniques is that they lead to an increasing number of nodes of the graph, thus requiring a very large size of the computer core memory. In other words, there is no a priori limitation of the number of nodes of the arborescence to be searched. This arborescence can have as
many as $2^N$ nodes (where $N$ is equal to the number of variables of the problem to be solved), thus making it impossible to solve large-scale problems.

Finally, heuristic techniques can solve certain types of problems, in a very few CPU time (sometimes less than one second CPU time), but they have the following disadvantages:

(i) They are not general enough to be used as a solution for every type of model. In general, the heuristic is very specific to the problem studied. Therefore, it cannot be used for other types of problems (unlike branch and bound techniques which are independent from the structures of the problems to be solved.)

(ii) They do not necessarily lead to the "true" optimal solutions of the problems. Usually, they lead to "acceptable solutions," which can be very different from the true optimal solutions. As a consequence, the solutions provided by these techniques can be very costly (since they are not optimal), especially if they are applied to real-life problems.

(iii) Very seldomly, the convergence of such methods can be proved.

The aim of this chapter is to present a mathematical programming algorithm that can allow us to avoid most of the disadvantages cited above. In this chapter it is mainly argued that in order to avoid the
disadvantages inherent to heuristic methods, companies prefer to obtain the true optimal configuration, even if it takes minutes of CPU time to provide it. The cost to run the computer code is very marginal, but the economies obtained can be very important in real-life examples, as for distributed information systems. To solve the two distributed information systems models described in the preceding chapter, we propose a mathematical programming algorithm that has the following characteristics:

- The number of nodes of the arborescence that have to be stored in the computer is at most equal to \( n \), where \( n \) is the number of variables of the problem.

- It can solve integer problems where the objective function is nonlinear without any assumption about the convexity of the problems. Therefore, we can solve real-life problems where there are included set-up costs.

- It has the property of heuristic methods in the sense that a rapid solution can be obtained in a very reasonable CPU time.

Our method incorporates much of the ideas developed by Little (7), Land and Doig (8) and Tomlin (9). An important aspect of our algorithm is the "binary bounded branch and bound" method described in Step 3 of our algorithm (Page 126). This binary method is based on the bounded branch and bound (BBB) method, first proposed by Abadie (10) and developed by Akoka (11). However, there are important differences
between the two methods:

1. The original BBB method was intended to solve general integer nonlinear programming problems. The method used in this thesis is very much specialized to the binary nonlinear case. This has a number of consequences on:

- **The state of the nodes in the arborescence**
  In the binary BBB method, we will only have the states describing the nodes that result only from the binary characteristics of the variables (see paragraph C of Appendix A)

- **The rules used in the management of the arborescence**
  This is due to the fact that the number of states of the arborescence is reduced when all the variables are binary (see paragraph D of Appendix A)

- **The conditions of refusal of certain nodes of the arborescence**
  In the binary case, and since the objective function in non-convex, two specific refusal cases are described (see paragraph D of Appendix A, rule 3)

2. Another difference is the one related to the convergence of the methods. In the general BBB method, the convergence may not be attained if the objective function or the constraints are non-convex. In our binary method, even though the objective function is nonlinear and non convex, the convergence is always obtained. This is due to the
binary characteristics of the variables which lead to a limited number of possible refusal of the nodes of the arborescence.

3. At each node of the arborescence and after solving the continuous problem, some variables may fortuitously become integers. In order to eliminate many nodes from the arborescence, we have explicitly in this thesis designed a criterion which enables us to schedule the fortuitous integer variables (see paragraph G on Appendix A). This was not the case in the original method. This criterion improves the computing efficiency of the method, and limits the number of nodes of the graph.

4. In the original method, the criterion used to choose the "separation variable" (see paragraph H of Appendix A) was the one that chose the variable which was the closest to its integer value. After several tests, it was found that this criterion was not suitable to the binary case. Therefore, we used in this thesis a pseudo-cost criterion which proved to be better than the original one and avoided searching for the optimal solution in non-desirable descending directions.

5. From a computer implementation viewpoint, we took advantage of the differences explained above to redesign a new computer code. The purpose was to limit the core memory to be used, (this is crucial if one wants to solve large-scale problems) and to accelerate the rate of convergence (this is also important from the viewpoint of CPU execution time).
Besides, we built a computer interface to the algorithm which automatically computes the objective function, the constraints, the gradient of the objective function, and the Jacobian of the constraints.

Finally, let us emphasize the fact that the algorithm used in this thesis consists of several steps which take into account the specific structure of our models (STEP 1, STEP 2, and STEP 4). One can argue about the possibility of eliminating STEP 4 and incorporating the lemmas used as constraints. Although it is possible to do so from a strict theoretical point of view, it seems to us that it has the disadvantage of increasing the number of constraints (which is already very high), thus increasing the core memory and the CPU time needed to solve the problem. Besides, there is no easy analytical way to incorporate lemma 3 as a constraint.

In summary, the algorithm used in this chapter is original in the sense that:

1. It takes advantage of the special structure of our models

2. It uses a specific binary BBB method.

This chapter is organized as follows: in Section II we present a brief outline of the algorithm. Each step of the algorithm will be described in more detail in Section III. However, in order to facilitate the reading of Step 3, much of the mathematics involved will be presented in Appendix A. Section IV will be devoted to the computer implementation of the algorithm. We will emphasize the aspects that are important from a user point of view.
In order to simplify the notations used, let us call our objective function \( \phi(x) \), the constraints \( h(x) \). Therefore, the simplified formulations of our models described in Chapter Three are:

\[
\begin{align*}
\text{min } & \phi(X) \\
\text{subject to:} & \\
(P) & h(x) \leq 0 \quad (c-1) \\
& 0 \leq X \leq 1 \quad (c-2) \\
& X = 0, 1 \quad (c-3)
\end{align*}
\]

where the components of vector \( X \) are \([ y_k^m, x_k^d, z^p_j, L^c_{ij}, X^d_{ijp}, X^p_{jkd} ]\).

We assume that:

(i) \( \phi(x) \) is a nonlinear function, continuously differentiable

(ii) constraints \((c-1)\) define a domain which is convex

(iii) constraints \((c-2)\) define a parallelotope \((\pi)\). \((\pi)\) is assumed to be bounded.

Let us now describe our algorithm.

II. The Algorithm: A Brief Outline

Step 1 wrapping the last constraint \((x = 0,1)\), solve the following continuous nonlinear programming problem:
\[
\begin{aligned}
&\min \phi(x) \\
\text{subject to:} \\
P(S) & \begin{cases}
& h(x) < 0 \\
& 0 < x < 1 \\
& x \text{ continuous}
\end{cases}
\end{aligned}
\]

**Step 2**

If all the variables \( L_{ij}^c \), \( X_{ijp}^d \) and \( x_{jkpd} \) are integer, go to Step 4. Otherwise go to Step 3.

**Step 3**

Use the Binary Branch and Bound method (described in the following pages) to find the optimal integer values for the variables \( L_{ij}^c \), \( X_{ijp}^d \) and \( x_{jkpd} \).

**Step 4**

Use the following lemmas to obtain the optimal values of the remaining variables \( y_k^m \), \( x_k^d \) and \( z_j^p \).

**Lemma 1**

If \( x_{ijp}^d = 1 \), then \( z_j^p = 1 \) for all \( i,j,p,d \).

**Lemma 2**

If \( x_{jkpd} = 1 \), then \( x_k^d = 1 \) for all \( j,k,d \).

**Lemma 3**

Let \( (x_k^d)^* \) be the optimal value of \( x_k^d \), then
\( y^m_k = (x_k^d)^* \) for all \( m \) such that:

\[
Z^p_j \leq \sum_m y^m_j, \quad \forall j \in J = K, \quad \text{and} \quad p \in P
\]

\[
\sum_{i,j,p,d} \alpha_{ip} x^{d}_{i,ip} x_{jkpd} + \sum_{i,p,d} \beta_{ip} x^{d}_{ip} + \sum_{i,p,d} (Q_{ip} + U_{ip}) x^{d}_{ip} \leq \sum_m \sum_k y^m\]

\( \forall i \in I = K, \quad \forall j \in J = K, \quad \forall p \in P, \quad \forall d \in N, \quad \text{and for all} \quad k \in K \)

\[
X^{d}_k \leq \sum_m y^m_k, \quad \forall k \in K, \quad d \in N
\]

Let us now describe in detail each step of the algorithm and prove the different lemmas:

III. Detailed Description of the Algorithm

Step 1 Solve the following problem:

\[
P(S) \quad \left\{ \begin{align*}
\min (x) \\
h_k(x) & \leq 0 \quad \text{(c-1)} \\
0 & \leq x \leq 1 \quad \text{(c-2)} \\
x \text{ continuous}
\end{align*} \right. 
\]

In this step we are solving the continuous nonlinear programming problem. In order to do so, we use the GRG (12) algorithm since Colville (13) found it faster than other nonlinear codes.

The GRG algorithm proceeds as follows:

(1) Vector \( x \) is partitioned into basic variables and non-basic
variables.

(ii) It computes the direction of move of the non-basic variables by the following substeps.
   - computes the reduced gradient of the objective function
   - computes the projected reduced gradient
   - computes the modified projected reduced gradient

(iii) Computes the direction of move of the basic variables

(iv) Compute a feasible point by solving a system of $m$ equations in $m$ unknowns.

(v) Improve the feasible point obtained until the criterion of convergence is satisfied.

A detailed mathematical version of this algorithm can be found in (14).

At the end of Step 1, we obtain either integer or continuous values for the variables $L^c_{ij}$, $X^d_{ijp}$, and $X_{jkpd}$ (and fortuitously for $y^m_k$, $X^d_k$, and $Z^p_j$).

Step 2 If all the variables related to:
   - communication lines ($L^c_{ij}$)
   - routing disciplines ($x^d_{ijp}$ and $x_{jkpd}$)

are integer, go to Step 4 to derive the optimal value of the other variables. Otherwise, get to Step 3 to use the BBB algorithm.
This step allows us to determine whether some of the variables may become integer during the process of the continuous optimization.* If this happens, these variables are called "fortuitous variables," and should be scheduled, in order to eliminate as many as possible, from the graph. The criterion which enables us to schedule them is described in paragraph G of Appendix A. The test used in this step works as follows: If, as a consequence from Step 1,

a) The variables $y_k^m$, $x_k^d$, $L_{ij}^c$, and $Z_j^p$ are integer, go to Step 4.

b) Only some variables are integer, schedule them according to criterion described in paragraph G of Appendix A, and go to Step 3.

c) All the variables are continuous, go to Step 3 without using the scheduling criterion.

Step 3 The Binary Bounded Branch and Bound Method

This step allows us to determine the integer optimal values of the variables $L_{ij}^c$, $X_{ijp}^d$, and $X_{jkpd}^c$. To do so, the method solves a set of continuous nonlinear programming problems, corresponding to the nodes of the arborescence. These continuous problems are put in a master

---

*There is no way to evaluate a priori the number of variables that can become integer as a result of the use of Step 1. This number depends on the input parameters of each problem.
list. At iteration 1, the master list contains problem P(S). This problem was already solved in Step 1. At any iteration t, let \( \hat{x} \) be the best integer solution obtained so far. We set \( \hat{\phi} = \phi(\hat{x}) \) [or, \( \hat{\phi} = +\infty \) if no integer solution was obtained]. The general procedure to be used is:

1. If the master list is empty, terminate the computations;
   
   [If \( \hat{\phi} < \infty \), then \( \hat{x} \) is the optimal solution.
   
   If \( \hat{\phi} = +\infty \), then there is no solution to the problem.]

   If the master list is not empty, go to (2).

2. Solve the last problem put in the master list (call it WP).
   
   Let \( X^* \) be the solution obtained, with \( \phi^* = \phi(X^*) \).
   
   If \( \phi^* > \hat{\phi} \), remove WP from the master list and go to (1).
   
   Otherwise go to (3).

3. If all \( x^*_j, j \in E \) (i.e. all the variables \( L^{c}_{ij}, X^{d}_{ijp} \) and \( X^{kpd}_{j} \)) are integers, then \( X^* \) is an integer solution better than \( X \). Let \( \hat{X} = X^* \), and \( \hat{\phi} = \phi^* \), remove WP from the master list, and go to (1).
   
   Otherwise, go to (4).

4. Choose a non-integer \( X^*_\beta, \beta \in E \) and consider the following three problems:
   
   (a) the problem derived from WP with \( X^*_\beta = [X^*_{\beta}] \)
   
   (b) the problem derived from WP with \( X^*_\beta = [X^*_{\beta}] + 1 \)

   where the symbol \([ \ ]\) means the integer value.
(c) the problem derived from WP with a certain variable

\[ X_\alpha = \begin{cases} 
[X_\alpha] - 1 \\
\text{or} \\
[X_\alpha] + 1
\end{cases} \]

- If both problems (a) and (b) were not already solved, add them to the master list and go to (2).

- If both problems (a) and (b) were already solved, remove WP from the master list, add to this list problem (c), and go to (2).

- If only either problem (a) or problem (b) was solved, remove WP from the master list, add to this list problem (c) and the unsolved problem (either problem (a) or problem (b)) and go to (2).

The main differences between this method and the traditional branch and bound methods are the following:

- Steps (1), (2) and (3) are the same in both methods. But, traditional branch and bound methods deal with linear problems, whereas our method deals with nonlinear problems.

- Step (3) of traditional branch and bound methods can be summarized as follows (15):

  - Choose a non-integer \( X_\beta, \beta \in E \), and add to the master list the following two new problems:
(a) the problem derived from WP with $X = \lfloor X \rfloor$

(b) the problem derived from WP with $X = \lfloor X \rfloor + 1$

As a consequence, the same variable $X_\beta$ can be chosen at each node of the arborescence, thus, increasing considerably the number of problems stored in the master list. In other words, Step 3 of traditional branch and bound methods leads to an increase of the number of the nodes of the arborescence, thus making it very difficult to solve large-scale integer problems. On the contrary, by using Step 3 of our method, we force the variable $X_\beta$, once fixed at a given integer value, to keep this value in all the nodes of its descendence. As a consequence, we limit a priori the number of nodes (i.e. problems) to be stored in the master list. This number is equal to $N$, where $N$ is the number of the variables of the problem and not $2^N$ as in the case of traditional branch and bound methods. Hence, we can solve integer large-scale problems, which is the case of the models developed in Chapter Three, for distributed information systems. In order to keep a reasonable size to this chapter, we describe all the mathematical aspects of the binary bounded branch and bound methods in Appendix A.

As a result of the use of Step 3, we are able to have the optimal integer values of the variable $L_{ij}^c$, $X_{ijp}^d$ and $X_{jkp_d}$. We can now proceed to Step 4, to derive the optimal values of the remaining variables, namely $y_k^m$, $x_k^d$ and $z_j^p$. 
Step 4  Use the following lemmas to obtain the optimal values of the remaining variables \(y_k^m, x_k^d\) and \(z_j^p\).

Lemma 1
If \(x_{i,j,p}^d = 1\), then \(z_j^p = 1\) for all \(i,j,p,d\)

Proof:
By definition, \(x_{i,j,p}^d = 1\), means that the traffic from node i (where the user is), has to be processed by program p located at node j, before access to database d is gained. As a consequence, \(z_j^p = 1\).

Lemma 2
If \(x_{j,k,p,d}^d = 1\), then \(x_k^d = 1\) for all \(j,k,d\)

Proof:
By definition, \(x_{j,k,p,d}^d = 1\) means that the transaction from node j to database d should be routed to node k. Therefore, \(x_k^d = 1\).

Lemma 3
Let \((x_k^d)^*\) be the optimal value of \(x_k^d\). Then, \(y_k^m = (x_k^d)^*\) for all \(m\) such that

\[
\begin{cases}
  z_j^p < \sum_m y_j^m, & \forall j \in J = K, \text{ and } p \in P \\
  \sum_{i,j,p,d} a_{i,j,p,d} x_{i,j,p,d}^d + \sum_{i,p,d} b_{i,p,d} x_i^d + \sum_{i,p,d} c_{i,p,d} x_i^d + \sum_{m \in M} (Q_{m,p,d} + U_{m,p,d}) x_{i,p,d}^d + \sum_{k \in K} m_{k,m} y_k^m & \forall i \in I = K, \forall j \in J = K, \forall p \in P, \forall d \in N, \text{ and for all } k \in K \\
  x_k^d \leq \sum_m y_k^m, & \forall k \in K, \forall d \in N
\end{cases}
\]
Proof:

In our model, we stated that the databases and programs should reside only at the nodes where at least a computer of capacity \( k_m \) is allocated, therefore

\[
x_k^d \leq \sum_m y_k^m, \quad \forall k \in K, \; d \in N
\]

\[
z_j^p \leq \sum_m y_j^m, \quad \forall j \in J = K
\]

By using Lemma 1, we have the optimal allocation of databases \((x_k^d)^*\). As a consequence, the computers are allocated at the same nodes \( k \). To determine which capacity to allocate to a computer situated at node \( k \), it is necessary to satisfy the following constraint:

\[
\sum_{i,j,p,d}^d \alpha_{ip} x_{ip}^d x_{jp}^d x_{kp}^d + \sum_{i,p,d}^p \beta_{ip} x_{ip}^d + \sum_{i,p,d}^p (q_{ip}^d + u_{ip}^d) x_{ip}^d x_{ip}^d x_{ip}^d \leq \sum_m^m k y_k^m
\]

\[
\forall i \in I = K, \quad \forall j \in J = K, \quad \forall p \in P, \quad \forall d \in N, \text{ and for all } k \in K
\]

The optimal solution will be obtained for the variables \( y_k^m \) satisfying the constraints above and minimizing the allocation cost of computers.

As we can see, Step 4 takes advantage of the special structure of the problem to obtain the optimal values of the variables not considered explicitly in Step 3. Step 4 is very important in the sense that it allows us to limit the branch and bound process of Step 3. By doing so, we do not search for the optimal values of all the variables, but only for a part of them (represented by the set \( E \)). As a consequence, we can solve large-scale problems with savings in terms of CPU time.
At the end of Step 4, all the optimal values for the variables are determined, thus leading to the optimal configuration of the distributed system.

IV. The Computer Implementation of the Algorithm

This section is devoted to the description of the computer implementation of the algorithm described in Section III. We will concentrate mainly on the hierarchical structure of the computer code. A description of only the new subroutines (and not of those belonging to the continuous code GRG) will be provided.

The computer model is written in FORTRAN, and has about 4000 statements. The hierarchical structure of its logical modules is depicted in Figure 18. The description of the main program and the subroutines used is given below.

MAIN This is the main program. It is the monitor program. It has the control of the program's execution, calling the most important subroutine BBB. It contains all the following data needed to run the model.

1. The number of nodes of the network.
2. The number of initial computers, their capacities and their costs.
3. The number of initial databases and programs, and the costs associated with them.
4. The number of types of communication lines and their
Figure 18 - HIERARCHICAL STRUCTURE OF THE COMPUTER MODEL
associated costs.
5. The distances between nodes of the network.
6. The costs of communication between the nodes.
7. The number of variables and constraints of the problem.
8. A starting set of values for the variables of the problem.
   It does not have to be necessarily feasible.

**BBB** This subroutine contains most of the features of the binary BBB algorithm. All the rules of the algorithm described in Appendix A are programmed. It calls subroutine OPTIM to initiate the solution for the continuous problem. It also calls subroutines ARBITR and CHOIX to choose the separation variable.

**OPTIM** It provides a way to evaluate the successive values of the optimal solutions, choosing the best one and transmitting it to BBB. It also fixes successively all the variables at their integer values.

**ARBITR** Once a separation variable $\beta$ is chosen, subroutine ARBITR allows us to:

\[
\text{optimize } X_\beta
\]

subject to:

\[
h(x) \leq 0
\]

\[
[X_\beta] \leq X_\beta
\]
by calling GRG. This optimization process allows us to keep $X_B$ at its integer value or to refuse it.

CHOIX It is the subroutine which chooses the separation variable by using the criterion of pseudo-cost described in paragraph H of Appendix A.

FECON In this subroutine, we describe the objective function of our problem. It is presented in the following manner:

$$\Phi = \sum_{i=1}^{9} \text{COST}_i$$

where COST$_i$, for all $i$, are the cost components of the objective function described in Chapter Three. As for the remaining subroutines, this subroutine is provided automatically. Therefore, the user does not have to specify the objective function.

CONTR It provides a way to write the constraints of the problem. The constraints are defined as:

$$VC(i) = h(x_i)$$

for all of the $X_i$'s of the problem.

GRADF It provides the means of deriving the gradient of the objective function. This gradient is defined, and

$$D\Phi(i) = \frac{\partial \Phi}{\partial x_i}, \quad \text{for all } i$$
In this subroutine, we derive the Jacobian of the constraints. The computer code needs to know the Jacobian of each constraint. This is provided by the interface of the following form:

\[ \frac{\partial V_i}{\partial x_j} \]

for all \( i \) and \( j \)

The reader should be aware of the fact that the last four subroutines constitute the interface with the binary BBB method. They are fully automatic, independent of the size of the problem or the particular set of data used in the distributed system. In fact, they are the FORTRAN translation of the general mathematical model as it is described in Chapter Three.

All the other subroutines that are not described in Section IV, are those used by the GRG (code for continuous nonlinear programming problem). They will not be described here, since the reader can find a detailed implementation of them in (12). Let us stress the fact that the only reason why GRG was used here is because it has been ranked first by Colville among all the other nonlinear programming codes. In fact, our binary BBB method can be used in conjunction with other nonlinear codes, but it will require some programming efforts as we have done for the present version.
V. Conclusion

In this chapter, a mathematical programming method which enables us to solve the distributed information systems models, was presented. Its main advantages are:

1. It leads to the true optimal solution rather than near-optimal solutions as it is the case for most of the heuristics used in the past.

2. It takes advantage of the special structure of the models to gain some core memory and CPU time.

The aim of the computer implementation of the algorithm is to allow us to solve real-life examples. This computer code and the model interface will be used in the next chapter to solve real-life examples.
REFERENCES

Chapter Four


(11) AKOKA, J., "Methodes Arborescentes de Resolution des Programmes Non-lineaires Totalelement ou Partielement Discrets," These 3eme Cycle, Universite de Paris, VI, Paris, 1975


CHAPTER FIVE
APPLICATION OF THE DESIGN MODELS
FOR DISTRIBUTED INFORMATION SYSTEMS

I. Introduction

One of the weaknesses of the approaches used in the past to develop models for distributed systems is the lack of application of such models to real-life examples. Although this aspect is crucial if one wants to prove the usefulness of these types of models, very seldom a discussion of their applicability to real-life examples is given. The validity of the model is rarely examined. The aim of this chapter is to show that our models for the design of distributed information systems are realistic enough to be applied to real-life examples.

The plan of this chapter is the following. Section II will be devoted to the application of our global model and the solution procedure to a real-life company. First, we will describe the settings, then we will present the company's experimental network; finally, we will discuss the solution obtained using our model, and analyze the main differences between both solutions. Some sensitivity analysis will be performed. In Section III, we will apply the model related to the distributed database system using the same set of data as Casey and Levin-Morgan. This will allow us to compare the results obtained, and point out the advantages of our model and the solution procedure used.
II. Application of the Global Model to the Design of a Distributed Information Systems for a Large Bank

As we have said before, the purpose of this section is to apply the global model to the design of a real-life distributed information system. To do so, we will first describe the settings and their data processing group. Then, we will briefly explain the bank's top management requirements for a distributed system. To allow a comparison with the solution provided by the use of our model, we will present the bank's experimental network. Finally, we will develop our solution and discuss all the results obtained, including a sensitivity analysis.

A. The Settings

In 1863 a small regional bank was formed in Europe. This was the beginning of an important expansion throughout the world:

1875: Creation of an office in Madrid
1876: Opening of a representation in Geneva
1878: The bank opened offices in Moscow, Alger and Alexandria

Between 1879 and 1913, the bank pursued its expansion throughout the world. During the era of the First World War, the bank was ranked first in the entire world. But, due to the two world wars and the crisis of 1939, the bank suffered setbacks, and in 1945 it was nationalized.

A new period of expansion started in 1946. All the traditional activities of the bank were developed. Today, with more than $18 billion
in deposits, the bank is one of Europe's largest banks. It is also one of the continent's major data communications users, serving customers in 41 countries and 2300 local branches through a worldwide financial network. The total number of persons working for the bank is 47,200, among them 10 percent are managers, and 42 percent are middle management.

B. The Data Processing Group

The data processing group is in charge of:

- The definition and the implementation of all the data processing activities
- The management of all the computer centers
- The development of new applications and structures including the issue of decentralization.

It has three functional units and four hierarchical units. The functional units are in charge of the long-run aspects of the data processing activities, while the hierarchical units are responsible for the short-run and daily activities of the data processing department.

1. Functional units

   (i) Foreign relations

   This unit is in charge of the relations, in terms of data processing, in the bank with foreign organizations (banks, insurance companies, administrations, etc.)
(ii) **Management**

Its role is to develop a three-year program. It also controls budget and the execution of new applications. Finally, it defines the policies of recruitment. This group is divided into three categories:

- budget management
- personnel management
- general management

(iii) **System Development**

This third group is in charge of system development, documentation, standards and formation of personnel.

2. **Hierarchical units**

There are four hierarchical units where are in charge of:

- management of the computer centers
- development and implementation of applications
- methods of data processing
- organization and implementation

The number of employees of the data processing group is 860. Its annual budget is $86 million. Its composition is the following:
The evolution of the budget for the equipment is given below:

<table>
<thead>
<tr>
<th>Year</th>
<th>Equipment</th>
<th>Personnel</th>
<th>System Development</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1975</td>
<td>$30M</td>
<td>$20M</td>
<td>$24M</td>
<td>$86M</td>
</tr>
<tr>
<td>1976</td>
<td>$38M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1977</td>
<td>$42M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1978</td>
<td>$51M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1979</td>
<td>$52M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td>$61.5M</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The evolution of the budget for the equipment is given below:

C. The Decentralization

Until 1973, the bank was heavily centralized. The equipment used was of different generations and was not adaptable for an effective management. The top management of the Bank wanted a data processing system that can be characterized by:

- Decentralization and specialization of data entry
- Integration of all the management aspects, including accounting, funds transfers, customers accounts, transactions, etc.

The top management imposed some constraints:

- The new system should be capable of replacing the old system without any difficulties in the daily functioning.
- Reliability was a prime factor.
- Choice of the most cost effective system.
D. The Bank's Experimental Network

The bank is experimenting with a multilevel distributed processing network. It is a message switching network (Figure 19).

Dual C-9000 central processors connect to a common data center through a switching arrangement, with one unit operating on line and the other standing by for automatic switchover to cover a failure in the primary machine. (Figure 20). The bank utilizes this dual-processor message switching to receive, store and send funds transfer and related data throughout the world. Included in the message switching network are dedicated lines for terminals at branches with high traffic volume and dial-up links for telex stations. Programmable Channel Termination Groups (PCTG's) connect network communication lines to the message switching network. (Figure 21) Dual PCTG's are used to offset the effects of failure in either of them. The multilevel distributed processing network will operate through system centers near City A, City B, and City C. Each of the three systems centers will contain redundant Collins C8562 front-end message switchers which, with ten PCTG's, will serve multiple IBM 370/168 hosts. (Figure 22).

Operating on a 24-hour, seven-day basis, the network carries roughly 9000 messages a day. The bank expects that traffic volume will increase to 15,000 daily messages. The network will integrate remote concentrators and about 1000 microprocessors controlling 6000 teller and back-office positions. It will permit centralized control of the bank's financial data communication, will offer a variety of real-time, point of sale customer services from IBM 3601 terminals at
FIGURE 19. The Bank's Distributed Network
FIGURE 20. The Bank's Dual-processor message switching system
FIGURE 21. The Message-Switching Network
FIGURE 22. A multilevel distributive processing network
teller positions.

Through the dedicated network, messages are transmitted directly to department user terminals. Permanent records are kept on all messages sent or received and on special data about messages to or from the Telex network. Disk recording of all message traffic allows retrieval, while aiding communications security and message accountability.

The bank's network will handle a variety of message traffic: inquiries and responses concerning customer accounts, transaction inputs and acknowledgement by the host processors, inter-center clearing traffic and batch I/O for account posting and journalizing. Host processor-generated reports will be delivered to branches daily.

Plans call for the C8562's to interconnect the geographically dispersed host processors over 9600 bps intercenter trunks and interface with the ten remote PCTG's via 9600-bps trunks. The PCTG's will take about 155 SDLC lines at 2400 (or more) bps to serve the microprocessors.

Each C8562 center will have two dual nine-channel tape units, a 600-lpm printer and supervisory control positions with system status and alarms printers.

When the system is fully implemented, each front end will be able to handle 115 inquiry/response transactions a minute giving an average terminal response time of 1.5 seconds. Message capacity is based on an average 64-character input and 50-character output.

The bank system is prepared for future expansion. Among the possibilities are the addition of automatic cash dispensers, use of packet switching networks and added distributed processing capability. The
network also has potential for interfacing networks such as SWIFT, the Society for Worldwide Interbank Financial Telecommunications.

E. Application of our Model

In order to apply our model, we first gathered data. It was a difficult task since, for some of the data, approximations were made. For example, the communication cost for updates was estimated equal to the communication cost for queries. In fact, there is reason to believe that updates cost is lower than queries cost, since most of the updates are usually made during the night when communication cost rate is lower. But, in the bank's example, a significant part of the updates (more than 60 percent) is made during the day. Therefore, it was considered by the management that the overall cost for updates was not too far from the actual cost of queries, which was well known.

Another estimation was made for the monthly cost of databases. The bank did not have an exact evaluation of the set-up and organizational costs for databases. But, we were told by the bank, that, according to their experience, a monthly cost of $10,000 was reasonable.

Finally, the bank's data processing group tend to see as negligible the storage cost of programs operating on the databases. This is true for a month, but is unreasonable for a long period of time. Therefore, as a reasonable estimation, a $10 per month for program storage was considered as more accurate. In summary, based on the bank's experience, the only costs that were estimated were:

- communication cost for updates
- monthly set-up and organizational cost for databases
- storage cost of programs.

All remaining costs (described below) were the exact evaluation of the bank. The set of possible nodes to design the distributed system is given in the figure below.

Using the reliability constraint described in Chapter Three (page 112) we have been able to reduce the number of possible nodes to four, namely cities A through D. The reason is that the bank felt that the only nodes which are reliable for good telephone service and prompt maintenance are the four nodes described above.
The other potential nodes correspond to smaller cities where the quality of service of the telephone system was considered as being less reliable. It was also considered that in terms of reliable maintenance, the time needed to restart the system after a failure would have been longer in the small cities where it is more difficult to have in-house repair shops and spare parts, as in the big cities. Besides, the four nodes described above are dominant in terms of customers using the bank's services.

The set of data used in our model is described below. In the following pages, nodes 1, 2, 3, 4 will respectively correspond to cities A through D.

The costs are expressed in American dollars.

The bank is using one type of "integrated database" and one type of program operating on the database. The type of program consists of several application programs accessing the different components of the database. Therefore, in the following pages, \( p = d = 1 \).

\[
Q_{ij} = U_{ij} \quad \text{Cost per Transaction shipped (in \$)}
\]

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Query Traffic (No. of Transactions)</th>
<th>Update Traffic (No. of Transactions)</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,500,000</td>
<td>650,000</td>
<td>0</td>
<td>0.015 0.03 0.018</td>
</tr>
<tr>
<td>2</td>
<td>1,340,000</td>
<td>650,000</td>
<td>0.015</td>
<td>0 0.039 0.054</td>
</tr>
<tr>
<td>3</td>
<td>1,500,000</td>
<td>650,000</td>
<td>0.030</td>
<td>0.039 0 0.021</td>
</tr>
<tr>
<td>4</td>
<td>1,500,000</td>
<td>650,000</td>
<td>0.018</td>
<td>0.054 0.021 0</td>
</tr>
</tbody>
</table>
Notice that node 2 has less query traffic due to the fact that the bank has less activities in that city. The query and update traffic represent the average load, and not the peak load.

(ii) **Querying and updating costs** (in $)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>22,500</td>
<td>45,000</td>
<td>27,000</td>
</tr>
<tr>
<td>2</td>
<td>20,100</td>
<td>0</td>
<td>52,260</td>
<td>72,368</td>
</tr>
<tr>
<td>3</td>
<td>45,000</td>
<td>58,500</td>
<td>0</td>
<td>31,500</td>
</tr>
<tr>
<td>4</td>
<td>27,000</td>
<td>81,000</td>
<td>31,500</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>9,750</td>
<td>19,500</td>
<td>11,700</td>
</tr>
<tr>
<td>2</td>
<td>9,750</td>
<td>0</td>
<td>25,350</td>
<td>35,100</td>
</tr>
<tr>
<td>3</td>
<td>19,500</td>
<td>25,350</td>
<td>0</td>
<td>13,650</td>
</tr>
<tr>
<td>4</td>
<td>11,700</td>
<td>35,100</td>
<td>13,650</td>
<td>0</td>
</tr>
</tbody>
</table>

**Querying Costs:** $Q_{ij}^d$

**Updating Costs:** $U_{ij}^d$

Notice that:

- querying cost between nodes 2 and 4 is different from querying cost between nodes 4 and 2. This is due to the fact that there are more queries emanating from node 4 to 2 (1,500,000 transactions) than from node 2 to node 4 (1,340,000 transactions)
- updating cost is cheaper than querying cost since there are less updates than queries
(iii) Costs of computers and databases (monthly rental costs)

<table>
<thead>
<tr>
<th>Computer</th>
<th>Capacity (Number of transactions)</th>
<th>Price in Dollars</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM 370/168</td>
<td>$K_1 = 13,000,000$</td>
<td>$C_1 = 45,000$</td>
</tr>
<tr>
<td>IBM 370/158</td>
<td>$K_2 = 7,000,000$</td>
<td>$C_2 = 35,000$</td>
</tr>
</tbody>
</table>

The capacity of the computer is expressed in terms of throughput. In other words, the capacity is expressed in terms of transactions that can be processed per unit of time. For example, the IBM 370/168 can process 18,000 transactions per hour (the transaction is based on a length of 60 characters), or 13,000,000 transactions per month.

The monthly cost of the database is considered equal to $10,000. This includes set-up cost and organizational cost. As we have said earlier, this figure is the estimation of the bank. It is represented in the model by the parameter $D_d$.

(iv) Costs of communication line

<table>
<thead>
<tr>
<th>Variable cost $B_c$</th>
<th>$100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed cost $- B_l_c$</td>
<td>$200</td>
</tr>
</tbody>
</table>

To connect the nodes of the network, the bank was considering one type of communication line, with a 9600 bps capacity.
(v) **Distances between nodes (in kilometers) - \( D_{ij} \)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>500</td>
<td>240</td>
<td>260</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>0</td>
<td>290</td>
<td>760</td>
</tr>
<tr>
<td>3</td>
<td>240</td>
<td>290</td>
<td>0</td>
<td>450</td>
</tr>
<tr>
<td>4</td>
<td>260</td>
<td>760</td>
<td>450</td>
<td>0</td>
</tr>
</tbody>
</table>

(vi) **Storage cost of database and programs**

The monthly storage costs of database and programs are:

<table>
<thead>
<tr>
<th>Node</th>
<th>Program-S(_{jp})</th>
<th>Database-C(_{kd})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>500</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>500</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>500</td>
</tr>
</tbody>
</table>

The bank considers the concept of "total database" which includes all the data corresponding to all customers, and not only the data of the customers of a particular region. The storage cost of database is considered as different from the set-up and organizational cost.
(vii) **Expansion factors**

- for queries \( \alpha = 1 \)
- for updates \( \beta = 1 \)

This means that a message sent by user to a program has the same length as the consequent message sent by programs to database.

(viii) **Return flow of information**

No evaluation of the return flow of information has been given to us by the bank. Therefore, \( \gamma'_{ij} = \gamma_{jk} = 0 \).

By using the global model described in Chapter Three, the algorithm presented in Chapter Four and the data provided by the bank, we obtain the following results (Figure 23).

- optimal allocation of computers at nodes 1, 3, 4. The computer allocated to node 1 is an IBM 370/168. The computers allocated to nodes 3 and 4 are both 370/158 (i.e., \( y_1^1 = 1, y_3^2 = 1 \) and \( y_4^2 = 1 \))

- optimal allocation of databases at nodes 1, 3 and 4 (i.e. \( x_1^1 = 1, x_3^1 = 1, x_4^1 = 1 \))

- optimal allocation of programs at nodes 1, 3 and 4 (i.e. \( z_1^1 = 1, z_3^1 = 1, z_4^1 = 1 \))

- optimal allocation of communication lines between nodes 1-2, 1-3 and 1-4 (i.e. \( L_{12}^1 = 1, L_{13}^1 = 1 \) and \( L_{14}^1 = 1 \))
the following optimal routing disciplines for messages:

. optimal routing of messages between users and programs:

\[ x_{111}^1 = 1, \ x_{211}^1 = 1, \ x_{331}^1 = 1, \ x_{441}^1 = 1 \]

Notice that \( x_{111}^1, x_{331}^1, x_{441}^1 \) represent the routing of local processing. \( x_{211}^1 \) indicates that the processing requirement of node 2 should be routed to node 1. Intuitively, one can expect that this processing should be routed to node 3, which is the nearest. However, since the cost of shipping the queries and updates to node 3 is greater than shipping them to node 1, it is economically more advantageous to ship them to node 1.

. optimal routing disciplines between programs and databases

\[ x_{1111}^1 = 1, \ x_{3311}^1 = 1, \ x_{4411}^1 = 1 \]

Using this information, we obtain the following distributed system:

---

**FIGURE 23. Our Solution**

---
where
\[ \begin{align*}
P &= \text{program} \\
D &= \text{database} \\
C_1 &= \text{computer IBM 370/168} \\
C_2 &= \text{computer IBM 370/158}
\end{align*} \]

If we take into account the locations of fixed branches of the bank that should be serviced by the network, we obtain the following overall bank's network (Figure 24). It should be clear that the locations of these branches were not determined by our algorithm, but were given to us by the bank. They are considered as fixed locations. Therefore, our algorithm did not have to determine the optimal locations of the PCGT's. Our solution and the bank's experimental network are presented in Figure 25. Let us compare these solutions and analyze the main differences.

(i) **The Optimal Locations of the Computers**

Our solution recommends the allocation of computers to nodes 1, 3 and 4. The bank's experimental network allocates computers to nodes 1 and 3, but instead of node 4 it uses node 2. There are several reasons that can explain this difference. First, the bank did not use any optimization model to determine the optimal locations of computers. Instead, an intuitive way which takes into account the existence of past computer centers, was used. Therefore, the bank's network was not optimal. A second reason for the difference is the fact that the bank kept node 2 for social and political arguments. In the past, node 2 played an important role in the data processing activities of the bank. Therefore, it was not
C1 = IBM 370/168
C2 = IBM 370/158
P = program
D = database
= dual processor collins system
= minicomputer + PC T G
= intelligent terminal at teller

FIGURE 24. The Recommended Bank's Network
### Figure 25. Comparison of the Two Solutions

<table>
<thead>
<tr>
<th>Allocation of computers at nodes</th>
<th>The Bank's Solution</th>
<th>Our Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3</td>
<td></td>
<td>1, 3, 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of computers</th>
<th>3 IBM 370/168</th>
<th>2 IBM 370/158</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 IBM 370/168</td>
<td>1 IBM 370/168</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Allocation of databases at nodes</th>
<th>1, 2, 3</th>
<th>1, 3, 4</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Allocation of programs at nodes</th>
<th>1, 2, 3</th>
<th>1, 3, 4</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Allocation of communication lines between nodes</th>
<th>1 - 2 4 - 1</th>
<th>1 - 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 - 3 1 - 3</td>
<td>1 - 3</td>
<td></td>
</tr>
<tr>
<td>3 - 4 2 - 4</td>
<td>1 - 4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimal routing of messages between users and programs</th>
<th>To the nearest node</th>
<th>X_{111} \times 1 \times 331 \times 1 X_{441}</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Optimal routing of messages between programs and databases</th>
<th>To the nearest node</th>
<th>X_{111} \times 3314 X_{4414}</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Cost (monthly)</th>
<th>$341,380</th>
<th>$298,880</th>
</tr>
</thead>
</table>

desirable for social and political reasons to lay off all the people involved or to displace them. (Notice that this constraint can be incorporated in our model by letting $y_k = 1$ for the concerned node $k$.)

(ii) The Capacities of the Computers

The bank's network is composed of three IBM 370/168. Our solution recommends the use of one IBM 370/168 in node 1 and two IBM 370/158 in nodes 3 and 4. Given the actual bank's query and update traffic, the use of 3 IBM 370/168's is not justified. This is clearly not an optimal solution. One reason that can explain the use of the three IBM 370/168's is the fact that the bank was expecting an increase of its activities in the short run. This stems from the potential growth of each node in terms of clients and transactions. Therefore, the management of the bank wanted a network that can handle the 60 percent of transactions increase (from 9000 messages a day to 15,000 messages a day, or 50 percent increase at each node). That is probably the most serious reason that can justify the use of three IBM 370/168's. However, an analysis of the table below, shows that:

<table>
<thead>
<tr>
<th>Node</th>
<th>Type of Computer Allocated</th>
<th>Present Capacity of the Computer Allocated</th>
<th>Present Processing Requirement</th>
<th>Expected Processing Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IBM 370/168</td>
<td>13 million</td>
<td>12,500,000</td>
<td>16,250,000</td>
</tr>
<tr>
<td>2</td>
<td>IBM 370/168</td>
<td>13 million</td>
<td>5,930,000</td>
<td>6,819,500</td>
</tr>
<tr>
<td>3</td>
<td>IBM 370/168</td>
<td>13 million</td>
<td>6,250,000</td>
<td>7,187,500</td>
</tr>
</tbody>
</table>
(i) At node 1, the expected processing requirement is equal to 16,250,000 transactions (obtained by adding to the present processing requirement, the 50 percent increase expected at node 1 and the 50 percent increase expected at node 4). This processing requirement clearly exceeds the capacity allocated to node 1. Therefore, in order to handle the expected increase, the bank should expand the capacity at node 1, probably by adding complementary secondary storage.

(ii) At node 2, the expected processing requirement does not exceed the capacities allocated. However, there is no need for an IBM 370/168 since the capacity of the IBM 370/158 can handle the expected processing requirement. Therefore, the use of 2 IBM 370/168's is not justified, even if we take into account the expected increase. Using our solution, the table below shows that:

<table>
<thead>
<tr>
<th>Node</th>
<th>Type of Computer Allocated</th>
<th>Present Capacity of the Computer Allocated</th>
<th>Present Processing Requirement</th>
<th>Expected Processing Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IBM 370/168</td>
<td>13 million</td>
<td>9,580,000</td>
<td>12,454,000</td>
</tr>
<tr>
<td>3</td>
<td>IBM 370/158</td>
<td>7 million</td>
<td>6,250,000</td>
<td>7,187,500</td>
</tr>
<tr>
<td>4</td>
<td>IBM 370/158</td>
<td>7 million</td>
<td>6,250,000</td>
<td>7,187,500</td>
</tr>
</tbody>
</table>
(i) As for the bank's solution, an expansion of the secondary storage at node 1 is needed. (This is due to the fact that there is some local processing requirement which was not taken into account.)

(ii) The expected processing requirement being slightly larger than the present capacity of the IBM 370/158. There is no need for the use of two IBM 370/168's at nodes 3 and 4, but some additional secondary storage may be needed.

In summary, our solution does provide the possibility to handle the expected increase in terms of transactions. It should be noticed here that even if we allow the use of three IBM 370/168's, this is not going to change the optimal locations of the computers. In other words, even if we take into account the increase in terms of transactions, there is no reason to allocate a computer to node 2.

(iii) The Optimal Locations of Databases and Programs

The bank's network stores copies of the database and of the program at nodes 1, 2, 3. This is due to the fact that the computers are located in those nodes. Similarly, our solution recommends the storage of the copies of the database and program at nodes 1, 3 and 4. This is also due to the fact that the computers are located at the same nodes. Given the facts that the development and storage costs are the same at each node, the difference in terms of locations of the data bases and programs does not lead to a difference in terms of total cost, since
both solutions are recommending the storage of three copies of database and program.

(iv) Optimal Allocation of Communication Lines

The bank uses a fully connected network. Our solution recommends the following links: nodes 1-2, 1-3 and 1-4. The existence of a fully connected network is a consequence of: (i) the lack of explicit modelling of message routing disciplines; (ii) the desire to assure high reliability. By having a fully connected network, the bank avoided facing the issue of optimal routing of messages, since it is always possible in a fully connected network to find a path for transactions emanating from every node. But this is not an optimal way to route the messages. The solution of the bank increases the cost of communication lines allocation. Another reason that can explain the use of a fully connected network is the one related to reliability. The bank increases the reliability of the network by having a fully connected network, which allows to the user different links if one or more communication lines are in breakdown. This argument is partly true in the sense that the reliability of the network depends also on the number of computers, databases and programs. In order to achieve a full reliability, one has to have a computer, a database and a program at each node of the network. In other words, a fully connected network (in terms of communication lines) does not necessarily assure complete reliability. As a consequence, except for communication lines, our solution achieves roughly the same kind of reliability, but at a lower cost. Besides, if we allow ourself to have a fully connected network in order to assure
the same level of reliability, we will have an incremental cost of only $2,100, representing the cost of adding the following communication lines:

<table>
<thead>
<tr>
<th>communication line between nodes</th>
<th>fixed cost</th>
<th>variable cost</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 - 3</td>
<td>$200</td>
<td>$290</td>
<td>$490</td>
</tr>
<tr>
<td>2 - 4</td>
<td>$200</td>
<td>$760</td>
<td>$960</td>
</tr>
<tr>
<td>3 - 4</td>
<td>$200</td>
<td>$450</td>
<td>$650</td>
</tr>
</tbody>
</table>

If we add this amount to the cost of our initial solution ($298,880), we will have a monthly cost of $300,900. Our solution will still be $46,480 less expansive than the bank solution, with the same degree of reliability, as explained below:

<table>
<thead>
<tr>
<th>Cost Components</th>
<th>Our Solution (including high reliability)</th>
<th>The Bank Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>communication lines</td>
<td>$3,700</td>
<td>$3,700</td>
</tr>
<tr>
<td>computers</td>
<td>115,000</td>
<td>135,000</td>
</tr>
<tr>
<td>databases</td>
<td>31,500</td>
<td>31,500</td>
</tr>
<tr>
<td>programs</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>communication cost of queries and updates</td>
<td>150,670</td>
<td>177,150</td>
</tr>
<tr>
<td>Total</td>
<td>$300,900</td>
<td>$347,380</td>
</tr>
</tbody>
</table>
(v) **Optimal Routing Disciplines**

As we have seen before, the bank does not have an optimal routing discipline. Our solution allows us to lower the cost of communication for queries and updates by utilizing the optimal routing disciplines determined by our model. This is not the case for the bank's solution, as a consequence of the fact that the bank did not explicitly tackle the problem.

(vi) **Optimal Cost**

The optimal cost using the bank's solution is $347,380 per month. Our solution indicates a monthly cost of $300,900. The difference $46,480 is due to the following reasons:

- In our solution, we have a better allocation of the computers and their capacities. This permits a lower equipment cost.
- In our solution, we have a better allocation of programs and databases
- In our solution, we have a better allocation of communication lines and routing disciplines. The combination of those two elements permits the lowering of the communication cost for queries and updates.

The effect of all these elements allows us to design a distributed system at a lower cost than the bank's solution. The difference of $46,480 is not negligible, especially when one keeps in mind that this is a monthly saving, which is equivalent to a saving of $557,760 a year.
One can argue that our solution does not take into account expected increase in terms of the bank's activities, in the long-run. It is our opinion that in the long run, it is always possible to expand the two 370/158 to two 370/168's. The incremental cost that this change will require cannot offset the savings (about $557,760 a year) resulting from our solution. This is especially true when one keeps in mind that the 370/158 is fully compatible with the 370/168. Therefore, there is no need to convert programs or databases.

In summary, our solution provides the same kind of services but at a lower cost. An important question to ask is: how robust is the solution if we vary some important factors? To answer this question, let us perform some sensitivity analysis.

F. Sensitivity Analysis

In this section we will vary two important factors and evaluate the consequences on the overall network design. The two factors are:

- communication cost of queries and updates
- the query traffic at node 2

The reason for varying the communication cost of queries and updates resides in the fact that the public communication system used by the bank's network will change and will offer a lower rate for the users. It is believed that the rate for communication lines will be proportional to the distance between nodes (that was not the case before). In this case, since communication cost is an important element in the overall
cost of the network, we can expect some changes in terms of optimal solutions.

The reason why we are varying the query traffic at node 2 is to evaluate to what extent it is possible to have a change in the network if one node expects an increase of query traffic. This is particularly important for node 2 for which our solution and the bank's solution have differences. Therefore, the possible cases are the following:

<table>
<thead>
<tr>
<th>old communication cost for queries and updates</th>
<th>old query traffic at node 2</th>
<th>new communication cost for queries and updates</th>
<th>new query traffic at node 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>Case 3</td>
<td>Case 2</td>
<td>Case 4</td>
</tr>
</tbody>
</table>

Since Case 1 was the subject of the preceding section, we will perform only the sensitivity analysis related to cases 2, 3 and 4. The new communication costs for queries and updates are proportional to the distances between the nodes, and are equal to (in dollars per transaction shipped):
The new query traffic at node 2 is considered equal to 1,500,000 transactions per month (roughly 12 percent increase). Thus making it as important as the other nodes.

Using this new set of data, we obtain the following results (Figure 26):
### FIGURE 26. Sensitivity Analysis Results

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
<th>Case 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation of computers at nodes</td>
<td>1, 3, 4</td>
<td>1, 3, 4</td>
<td>1, 3, 4</td>
<td>1, 3, 4</td>
</tr>
<tr>
<td>Types of computers</td>
<td>2 IBM 370/158*</td>
<td>2 IBM 370/158+</td>
<td>2 IBM 370/158*</td>
<td>2 IBM 370/158+</td>
</tr>
<tr>
<td></td>
<td>1 IBM 370/168**</td>
<td>1 IBM 370/168++</td>
<td>1 IBM 370/168**</td>
<td>1 IBM 370/168++</td>
</tr>
<tr>
<td>Allocation of databases at nodes</td>
<td>1, 3, 4</td>
<td>1, 3, 4</td>
<td>1, 3, 4</td>
<td>1, 3, 4</td>
</tr>
<tr>
<td>Allocation of programs at nodes</td>
<td>1, 3, 4</td>
<td>1, 3, 4</td>
<td>1, 3, 4</td>
<td>1, 3, 4</td>
</tr>
<tr>
<td>Allocation of communication lines between nodes</td>
<td>1 - 2</td>
<td>2 - 3</td>
<td>1 - 3</td>
<td>2 - 3</td>
</tr>
<tr>
<td></td>
<td>1 - 3</td>
<td>1 - 3</td>
<td>1 - 3</td>
<td>1 - 3</td>
</tr>
<tr>
<td></td>
<td>1 - 4</td>
<td>1 - 4</td>
<td>1 - 4</td>
<td>1 - 4</td>
</tr>
<tr>
<td>Optimal routing of messages between users and programs</td>
<td>$x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$</td>
<td>$x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$</td>
<td>$x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$</td>
<td>$x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$, $x_{111}^1$, $x_{331}^1$</td>
</tr>
<tr>
<td>Optimal routing of messages between programs and databases</td>
<td>$x_{211}^1$, $x_{441}^1$, $x_{231}^1$, $x_{441}^1$, $x_{211}^1$, $x_{441}^1$, $x_{231}^1$, $x_{441}^1$</td>
<td>$x_{211}^1$, $x_{441}^1$, $x_{231}^1$, $x_{441}^1$, $x_{211}^1$, $x_{441}^1$, $x_{231}^1$, $x_{441}^1$</td>
<td>$x_{211}^1$, $x_{441}^1$, $x_{231}^1$, $x_{441}^1$, $x_{211}^1$, $x_{441}^1$, $x_{231}^1$, $x_{441}^1$</td>
<td>$x_{211}^1$, $x_{441}^1$, $x_{231}^1$, $x_{441}^1$, $x_{211}^1$, $x_{441}^1$, $x_{231}^1$, $x_{441}^1$</td>
</tr>
<tr>
<td>Cost (monthly)</td>
<td>$298,880$</td>
<td>$215,340$</td>
<td>$301,280$</td>
<td>$216,778$</td>
</tr>
</tbody>
</table>

* at nodes 3 and 4  
** at node 1  
+ at nodes 1 and 4  
++ at node 3
From the previous figure, we can draw the following conclusions:

1. With the new rate for communication cost of queries and updates, the overall design of the network does not change, even if we take into account the potential increase of query traffic at node 2. However, the value of the optimal cost is roughly 28 percent lower than the optimal cost obtained with the original cost of queries and updates. Finally, the use of an IBM 370/168 is shifted to node 3.

2. When the query traffic is the same at all the nodes (1,500,000 transactions), the network configuration does not change, but there is an increase in terms of optimal cost. The cost of the optimal solution is $301,280, when the query traffic is the same at all the nodes. The cost of the optimal solution is $298,880, when node 2 has only 1,340,000 query transactions. This represents an increase of less than one percent.

3. The optimal cost of the solution obtained when we consider the new communication rate in conjunction with the new query traffic is about 28 percent less expensive than the optimal solution obtained when we consider the old communication rate used in conjunction with the new query traffic. Notice that here too, the use of the IBM 370/168 is shifted to node 3.

4. In all the four cases, our solution performs better in terms of cost than the bank's solution, as it is shown below:
To explain it, we must consider the following facts:

- the cost of communication for queries and updates represents in all the preceding cases about 58 percent of the overall computer network cost

- the sum of the distances between node 2 and the three other nodes is bigger than for the others, as shown below:

<table>
<thead>
<tr>
<th>Sum of the distances from node</th>
<th>In kilometers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to the three other nodes</td>
<td>1,000</td>
</tr>
<tr>
<td>2 to the three other nodes</td>
<td>1,550</td>
</tr>
<tr>
<td>3 to the three other nodes</td>
<td>980</td>
</tr>
<tr>
<td>4 to the three other nodes</td>
<td>1,470</td>
</tr>
</tbody>
</table>
If we consider that all the nodes have the same amount of queries and update traffic (as in cases 3 and 4), and given the high cost of communication for queries and updates, there is a tendency to eliminate as a potential computer network node the one with the biggest sum of the distances to the other nodes. In our case, it is node 2. This is not the only reason. An optimal routing discipline for messages contributed also to the same effect. It allows the reduction of overall cost of communication by minimizing the number of potential nodes of the network.

As a conclusion, we can say that the sensitivity analysis confirmed most of the conclusions drawn after the first application of our model (case 1). However, from the figure above we can also draw the following conclusions:

(i) In the bank's solution:

a) The costs of the system in case 1 and case 3 are the same. This is due to the fact that the 160,000 transactions increase is processed locally in node 2, where a computer exists, therefore not incurring an additional communication cost. This observation is also valid for the costs of the system in case 2 and case 4.

b) The introduction of the expected new communication rate lowers the overall system's cost from $347,380 to $241,918 a month. This is almost a 31 percent reduction. The consequences of the introduction of a new communication rate will be discussed in the next pages.
(ii) **In our solution:**

a) The costs of the system in case 1 and case 3 are different (respectively $298,880 and $301,280 a month). This can be explained by the following fact: In our solution, we do not allocate a computer to node 2. Therefore, all the query (and update) traffic is sent to node 1. Hence, an increase of queries at node 2 will lead to an increase of the overall communication cost. This remark is valid for the costs of the system in cases 2 and 4.

b) The introduction of the expected new communication rate does lower the overall cost of the system but not in the same proportion as in the bank's solution. The reduction incurred is about 26 percent of the overall cost ($298,880-$215,340), whereas in the bank's solution, the reduction was 31 percent. This can be explained by the fact that in the bank's solution, the original communication cost of queries and updates was very high (due to a non-optimal allocation of databases, programs and computers). Therefore, a decrease in the communication rate will have a bigger effect than in our solution. As a consequence, the differences in terms of cost between our solution and the bank's solution will tend to decrease. For example, the difference between our solution and the bank's solution when we consider the old communication rate, is $48,500, whereas this difference is $26,570 if we consider the new communication rate. This will have a consequence on our final recommendations.
Recommendations

1. The case where the old communication rate prevails

If in the near future, there will not be any change in the structure of the present communication rate, we recommend the use of the distributed system described in Figure 24 (our solution) to which we add communication lines between nodes 2-3, 2-4, and 3-4. By doing so, we achieve a fully connected network leading to the type of high reliability desired by the management of the bank. The cost of such a system will be $300,900 per month (see page 166), therefore $46,480 less expensive than the actual bank's experimental system. This will lead to a savings of $557,760 a year.

2. The case where the new communication rate is introduced

If in the near future, the new communication rate is introduced, we recommend the use of the actual bank's experimental network. To explain the logic behind this recommendation, let us consider the following facts: In the case of the new communication rates, the cost of the bank's solution is $241,918 per month. Our solution leads to a cost of $215,340 per month. Therefore, the difference is $26,578 per month, or $318,936 a year. Although our solution is less expensive than the bank's experimental network, but given the fact that it recommends the abandon of node 2, it will be politically and socially unwise to lay out or displace the personnel involved in node 2. This will create some social perturbations in the activities of the bank that can lead to some losses of clients.

To what extent the bank should choose one of the two recommendations
will depend on the management assessment of the capabilities and date of introduction of the new communication system.

Let us now turn to the application of our second model, the distributed database systems model.

III. Optimal Distributed Database Systems -- An Application of the Second Model

The aim of this section is:

1. to apply the distributed database model developed in Chapter Three to the same set of data used by Casey and Levin-Morgan. This will allow us to compare their results with ours and analyze the differences

2. To investigate to what extent the optimal location of databases is independent from their storage cost

3. To evaluate the effect of the return flow of information on the optimal assignment of databases.

4. To draw some conclusions about the approach taken in this thesis and previous approaches (mainly Casey's and Levin-Morgan's approaches).

In order to compare our model with previous models, let us first recall our formulation:
\[ \min F = \sum_{d,k} C_{kd} x^d_k \quad \text{(OF1)} \]

\[ + \sum_{j,p} S_{jp} Z^p_j \quad \text{(OF2)} \]

\[ + \sum_{i,j,p,d} Q_{ip}^d q_{ij} x^d_{ijp} (1 + \gamma_{ij}) \quad \text{(OF3)} \]

\[ + \sum_{i,j,p,d} U_{ip}^d u_{ij} x^d_{ijp} \quad \text{(OF4)} \]

\[ + \sum_{j,k,p,d} \alpha_{jpd} q_{jk} x^d_{jkpd} (1 + \gamma_{jk}) \quad \text{(OF5)} \]

\[ + \sum_{j,k,p,d} \beta_{jpd} u_{jk} x^d_{jkpd} \quad \text{(OF6)} \]


subject to:

\[ \sum_{k} x^d_k \geq 1, \quad \forall d \in N \quad \text{(CST1)} \]

\[ \sum_{j} Z^p_j \geq 1, \quad \forall p \in P \quad \text{(CST2)} \]

\[ \sum_{j} x^d_{ijp} = 1, \quad \forall i \in I = K, \ p \in P, \ d \in N \quad \text{(CST3)} \]

\[ \sum_{k} x^d_{jkpd} = Z^p_j, \quad \forall j \in J = K, \ p \in P, \ d \in N \quad \text{(CST4)} \]

\[ \sum_{i} x^d_{ijp} \leq \sigma Z^p_j, \quad \forall j \in J = K, \ p \in P, \ d \in N \quad \text{(CST5)} \]

\[ \sum_{j} x^d_{jkpd} \leq \sigma x^d_k, \quad \forall j \in J = K, \ p \in P, \ d \in N \quad \text{(CST6)} \]
There are mainly three important differences with the Levin-Morgan's model:

(1) Our model takes into account the return flow of information. This aspect is modeled in the components (OF3) and (OF5) of our objective function. That is an important aspect that changes the final optimal allocation of databases and programs, as it will be shown in the following pages. Levin-Morgan's models (as well as previous models of distributed database systems) ignored this aspect.

(2) The main contribution of Levin-Morgan's model resides in the fact that it can handle the case of heterogeneous computer networks. In other words, they did not assume independencies between programs and databases. However, only one aspect of this dependency is considered, namely the one stating that in order to access a database, one should first access the program operating on the database. The routing variable used in their model is:

\[
    x_{jkd} = \begin{cases} 
        1 & \text{if transactions from node } j \text{ to database } d \text{ are routed to node } k \\
        0 & \text{otherwise}
    \end{cases}
\]
As it can be seen, this routing discipline variable does not indicate which program (among all those residing at node \( j \)) can process the database \( d \) located at node \( k \). This is a serious shortcoming of their model, since it cannot handle the case where a full-dependency between programs and databases exists. In other words, their model ignored the fact that in order to access a specific database, it is necessary to access first only the program \textit{strictly related} to this particular database. This is a more general case than the one considered by Levin-Morgan’s model. To allow us to deal with the case of full-dependency, we defined the following variable:

\[
x_{jkpd} = \begin{cases} 
1 & \text{if there are transactions from node } \ j \text{ to database } \ d \text{ located at node } \ k, \\
\text{and which is processed by program } \ p \\
0 & \text{otherwise}
\end{cases}
\]

This permits us to define routing disciplines to databases via their related programs. The effects of this routing disciplines are modeled in the components \((OF5)\) and \((OF6)\) of our objective function.

\( (3) \) In their formulation, Levin-Morgan state that:

\[
\sum_{k} x_{jkpd} \geq 1, \quad \forall j \in J = K, \ d \in N
\]

(In other words, we have to assure that every transaction to every file via every program and from every node, will have a
predefined route.)

This type of constraint has two important shortcomings:

(i) It does not take into account the case of full-dependency between programs and databases (see paragraph 2 above.)

(ii) It contradicts the definition of $x_{jkd}$ which states that there is a routing discipline between programs and databases only if there exists a program at node $j$. By taking the sum over $k$ for all nodes $j$, there always exists at least one routing discipline at every node $j$, whether or not a program is located there. As a consequence, a contradiction of their definition will always occur. To avoid these two shortcomings, our formulation states that:

$$\sum_{k} x_{jkpd} = z_j^p, \forall j \in J = K, p \in P, d \in N$$

By doing so: (i) we assure that every transaction to every database via only the related program $p$ and from every node, will have a predefined route. (ii) A routing discipline exists from node $j$ only if $z_j^p \neq 0$ (i.e. only if a program exists at node $j$).

From the solution procedure viewpoint, there is an important difference between Levin-Morgan's enumeration method and our algorithm. In Levin-Morgan's work, databases and programs are separated and a
staged minimization approach is tried. The staging used is the following:

\[
\begin{array}{ccc}
\text{(1)} & \text{(2)} & \text{(3)} \\
\text{Min} & \text{Min} & \text{Min} \\
\text{databases} & \text{programs} & \text{message} \\
\text{location} & \text{location} & \text{routing} \\
\text{(given databases}} & \text{(given databases} & \text{and programs} \\
\text{location)} & \text{location)} & \\
\end{array}
\]

To solve minimization problem (2), they assume that storage costs for programs are zero. This assumption implies that programs are basically stored everywhere. As a consequence, program locations are chosen a priori by the designer. This contradicts Levin-Morgan's argument that it is necessary to minimize the number of copies of a program due to the problems of maintaining updated versions of a program in an heterogeneous network. In other words, Levin-Morgan solve the problem by avoiding it. This may invalidate their solution in the case of homogeneous computer networks, where the programs are not a priori excluded from some nodes.

On the contrary, in our approach we do solve the problem, including its program cost component, without choosing a priori the locations of the programs.

This leads us to a more general conclusion. In most of the cases, the introduction of new additional constraints (such as storage constraints, dependencies between databases, etc.) will lead to a violation
of Levin-Morgan's decomposition technique, thus making their solution procedure non-applicable; whereas our solution procedure can handle any type of additional constraints, due to the fact that it is a general mathematical programming algorithm.

Some of the differences between the distributed database system model presented in Chapter Three, and Casey's and Levin-Morgan's models are illustrated by the following example. The data for this example was taken from Casey's five-node example and is given below.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Cost per Megabyte Shipped</th>
<th>Query Traffic</th>
<th>Update Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>1  2  3  4  5</td>
<td>Q_{ij} = U_{ij}</td>
<td>Q_{ip}^{d}</td>
</tr>
<tr>
<td>1</td>
<td>0  6 12 9 6</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6  0 6 12 9</td>
<td>24</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>12 6 0 6 12</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>9  12 6 0 6</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>6  9 12 6 0</td>
<td>24</td>
<td>8</td>
</tr>
</tbody>
</table>

We make the same assumptions as Levin-Morgan:

- The expansion factors $\alpha$ and $\beta$ are equal to 1.
- Only one program and one database are considered.
- Both queries and updates must be processed by program $p$, which allow us to access the database.
- Program $p$ can be processed only at node 2 and node 3.
A. First Case -- No return flow of information and no storage cost of database

We first assume that there is no return flow of information to queries and that the storage cost of database and program is negligible. This is exactly the case treated by both Casey and Levin-Morgan. Therefore, \( C_{kd} = S_{jp} = 0 \), and \( \gamma_{ij} = \gamma_{jk} = 0 \). Figure 27 summarizes the optimal costs associated with both Casey's and Levin-Morgan's model and our model.

**FIGURE 27. Cost of Database Assignments: Case 1**

<table>
<thead>
<tr>
<th>Location of database copies at node</th>
<th>Program/File Location of Dependence</th>
<th>Program/File Dependence Casey</th>
<th>Program/File Dependence Levin/Morgan</th>
<th>Program/File Dependence Akoka/Chen (distributed database model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>860</td>
<td>1830</td>
<td>1830</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>972</td>
<td>972</td>
<td>972</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1038</td>
<td>1038</td>
<td>1038</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>915</td>
<td>1896</td>
<td>1896</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>915</td>
<td>2085</td>
<td>2085</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>852</td>
<td>1110</td>
<td>1110</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>774</td>
<td>1260</td>
<td>1260</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>726</td>
<td>1758</td>
<td>1758</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>867</td>
<td>2037</td>
<td>2037</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>856</td>
<td>762</td>
<td>762</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>730</td>
<td>1188</td>
<td>1188</td>
<td></td>
</tr>
<tr>
<td>Location of database copies at node</td>
<td>Program/File Independence Casey</td>
<td>Program/File Dependence Levin/Morgan</td>
<td>Program/File Dependence Akoka/Chen (distributed database model)</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------</td>
<td>--------------------------------------</td>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>735</td>
<td>1179</td>
<td>1179</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>804</td>
<td>1176</td>
<td>1176</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>729</td>
<td>1299</td>
<td>1299</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>753</td>
<td>2013</td>
<td>2013</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>810</td>
<td>912</td>
<td>912</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>762</td>
<td>1386</td>
<td>1386</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>759</td>
<td>1317</td>
<td>1317</td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>756</td>
<td>1452</td>
<td>1452</td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>753</td>
<td>1497</td>
<td>1497</td>
<td></td>
</tr>
<tr>
<td>145</td>
<td>705</td>
<td>1995</td>
<td>1995</td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>760</td>
<td>954</td>
<td>954</td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>765</td>
<td>999</td>
<td>999</td>
<td></td>
</tr>
<tr>
<td>245</td>
<td>717</td>
<td>1425</td>
<td>1425</td>
<td></td>
</tr>
<tr>
<td>345</td>
<td>711</td>
<td>1452</td>
<td>1452</td>
<td></td>
</tr>
<tr>
<td>1234</td>
<td>792</td>
<td>1176</td>
<td>1176</td>
<td></td>
</tr>
<tr>
<td>1235</td>
<td>789</td>
<td>1167</td>
<td>1167</td>
<td></td>
</tr>
<tr>
<td>1245</td>
<td>741</td>
<td>1593</td>
<td>1593</td>
<td></td>
</tr>
<tr>
<td>1345</td>
<td>735</td>
<td>1713</td>
<td>1713</td>
<td></td>
</tr>
<tr>
<td>2345</td>
<td>747</td>
<td>1215</td>
<td>1215</td>
<td></td>
</tr>
<tr>
<td>12345</td>
<td>771</td>
<td>1413</td>
<td>1413</td>
<td></td>
</tr>
</tbody>
</table>
The optimal database assignment under Casey's assumptions is obtained when database copies are stored at nodes 1, 4, 5. The cost associated with this assignment is 705. However, when the assumption of dependency between databases and program is made, the utilization of Casey's model leads to suboptimal results. The associated cost is 1995. The optimum found by Levin-Morgan is when copies of the database are stored at nodes 2 and 3. This is the same database assignment found by our algorithm.

However, a careful analysis shows that the optimal solutions provided by Levin-Morgan's model and our model are:

<table>
<thead>
<tr>
<th>Levin-Morgan</th>
<th>Akoka-Chen</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2 = x_3 = 1$</td>
<td>$x_2 = x_3 = 1$</td>
</tr>
<tr>
<td>$z_2 = z_3 = 1$</td>
<td>$z_2 = z_3 = 1$</td>
</tr>
<tr>
<td>$x_{121} = x_{221} = x_{331} = x_{431} = x_{521} = 1$</td>
<td>$x_{121} = x_{221} = x_{331} = x_{431} = x_{521} = 1$</td>
</tr>
<tr>
<td>$x_{121} = x_{221} = x_{331} = x_{421} = x_{521} = 1$</td>
<td>$x_{2211} = x_{3311} = 1$</td>
</tr>
<tr>
<td>All the remaining variables are equal to zero</td>
<td>All the remaining variables are equal to zero</td>
</tr>
</tbody>
</table>

In Levin-Morgan's results, the existence of $x_{121}$, $x_{421}$ and $x_{521}$ contradicts their definition of $x_{jkd}$. These variables are equal to zero in our results, which conforms to the definition of $x_{jkd}$. This aspect was discussed in page . Notice that the existence of these additional variables did not lead to an increase of the objective function. This is due only to the fact that we have only one program and one database. In the case of multiple programs and multiple
databases, an increase of the optimal value of the objective function will occur, thus leading to a suboptimal solution for the Levin-Morgan model.

B. Second Case - Existence of Storage Cost for Database

Let us relax the assumption made in the previous section and consider that there exist a cost related to the storage of the database (but not for program). It is generally considered that, given that the storage cost is the same at each node, the optimal assignment of database will be the same as in the previous case. This may be true for small databases where the storage cost is negligible. For real life database, the storage cost is important, and therefore can have an effect on the optimal location of databases. Let us show it. For a storage cost $C_{kd}$ of $500$ a month (which is a reasonable estimation for large database, the same storage cost was used in the bank's example), our model gives us the following results (Figure 28).

FIGURE 28. Cost of Database Assignment - Case 2 (storage cost = $500$)

<table>
<thead>
<tr>
<th>Database Location</th>
<th>Akoka/Chen's Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2330</td>
</tr>
<tr>
<td>2</td>
<td>1472</td>
</tr>
<tr>
<td>3</td>
<td>1538</td>
</tr>
<tr>
<td>4</td>
<td>2396</td>
</tr>
<tr>
<td>5</td>
<td>2585</td>
</tr>
<tr>
<td>Database Locations</td>
<td>Akoka/Chen's Model</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>12</td>
<td>2110</td>
</tr>
<tr>
<td>13</td>
<td>2260</td>
</tr>
<tr>
<td>14</td>
<td>2758</td>
</tr>
<tr>
<td>15</td>
<td>3037</td>
</tr>
<tr>
<td>23</td>
<td>1762</td>
</tr>
<tr>
<td>24</td>
<td>2188</td>
</tr>
<tr>
<td>25</td>
<td>2179</td>
</tr>
<tr>
<td>34</td>
<td>2176</td>
</tr>
<tr>
<td>35</td>
<td>2299</td>
</tr>
<tr>
<td>45</td>
<td>3013</td>
</tr>
<tr>
<td>123</td>
<td>1912</td>
</tr>
<tr>
<td>124</td>
<td>2886</td>
</tr>
<tr>
<td>125</td>
<td>2817</td>
</tr>
<tr>
<td>134</td>
<td>2912</td>
</tr>
<tr>
<td>135</td>
<td>2992</td>
</tr>
<tr>
<td>145</td>
<td>3495</td>
</tr>
<tr>
<td>234</td>
<td>2454</td>
</tr>
<tr>
<td>235</td>
<td>2499</td>
</tr>
<tr>
<td>245</td>
<td>2925</td>
</tr>
<tr>
<td>345</td>
<td>2952</td>
</tr>
<tr>
<td>1234</td>
<td>3176</td>
</tr>
<tr>
<td>1235</td>
<td>3167</td>
</tr>
<tr>
<td>1245</td>
<td>3593</td>
</tr>
<tr>
<td>1345</td>
<td>3713</td>
</tr>
</tbody>
</table>
As we can see, the optimal solution obtained without considering the storage cost of databases occurs when the copies of the database are stored at nodes 2 and 3. When we consider explicitly the storage cost of database, the new optimal solution is obtained when we store the database only at node 2. This is due to the fact that when we have a high cost of storage of the database (which is the case for large databases), it is preferable to minimize the number of copies of the database to be stored. In this example, if the database is stored at nodes 2 and 3, the total cost is $1762. The storage cost is only $1000 (about 57 percent of the total cost), while the communication cost for queries and updates is $762 (about 43 percent of the total cost). Therefore, the storage cost is higher than communication cost. This tends to push the storage of database to a solution that minimizes the number of copies. This precisely is the case when we store the database at node 2 only.

An important conclusion that can be drawn from this example is that the optimal location of databases are not independent from their storage costs, when this storage cost is not negligible or is the same at every node of the network. Does this conclusion hold when the storage cost is different from one node to the others?

Given the structure of our distributed database systems model, and if we assume that the storage cost of databases is different at

<table>
<thead>
<tr>
<th>Database Locations</th>
<th>Akoka/Chen's Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>2345</td>
<td>3215</td>
</tr>
<tr>
<td>12345</td>
<td>3913</td>
</tr>
</tbody>
</table>
each node (this is possible when the copies of the database are stored on different storage devices such as disks or tapes), then it is possible to consider once again that the optimal assignment of databases is dependent on their storage cost. To show it, let us use the results obtained in Case 1. Let us consider that in nodes 1, 3, 4 and 5, it is possible to use slow speed disks in order to store the database, while at node 2 we can use high-speed disks. Furthermore, let us consider the storage cost at nodes 1, 3, 4 and 5, equal to $100, whereas the storage cost at node 2 is equal to $700. Using the model we will obtain the assignment given in Figure 29. As we can see, the optimal location of the database copies changes. The new optimal solution, equal to $1,138 is obtained when the database is stored only at node 3.

FIGURE 29. Different Storage Costs at the Nodes

<table>
<thead>
<tr>
<th>Location of database copies at node</th>
<th>Program/File Dependence Akoka/Chen</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1930</td>
</tr>
<tr>
<td>2</td>
<td>1672</td>
</tr>
<tr>
<td>3</td>
<td>1138 - new optimal solution</td>
</tr>
<tr>
<td>4</td>
<td>1996</td>
</tr>
<tr>
<td>5</td>
<td>2185</td>
</tr>
<tr>
<td>12</td>
<td>1910</td>
</tr>
<tr>
<td>13</td>
<td>1460</td>
</tr>
<tr>
<td>14</td>
<td>1958</td>
</tr>
<tr>
<td>15</td>
<td>2237</td>
</tr>
<tr>
<td>Location of database copies at node</td>
<td>Program/File Dependence Akoka/Chen</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>23</td>
<td>1562 - previous optimal solution</td>
</tr>
<tr>
<td>24</td>
<td>1488</td>
</tr>
<tr>
<td>25</td>
<td>1979</td>
</tr>
<tr>
<td>34</td>
<td>1376</td>
</tr>
<tr>
<td>35</td>
<td>1499</td>
</tr>
<tr>
<td>45</td>
<td>2213</td>
</tr>
<tr>
<td>123</td>
<td>1912</td>
</tr>
<tr>
<td>124</td>
<td>2386</td>
</tr>
<tr>
<td>125</td>
<td>2317</td>
</tr>
<tr>
<td>134</td>
<td>1652</td>
</tr>
<tr>
<td>135</td>
<td>1697</td>
</tr>
<tr>
<td>145</td>
<td>2195</td>
</tr>
<tr>
<td>234</td>
<td>1854</td>
</tr>
<tr>
<td>235</td>
<td>1899</td>
</tr>
<tr>
<td>245</td>
<td>2325</td>
</tr>
<tr>
<td>345</td>
<td>1752</td>
</tr>
<tr>
<td>1234</td>
<td>2176</td>
</tr>
<tr>
<td>1235</td>
<td>2167</td>
</tr>
<tr>
<td>1245</td>
<td>2593</td>
</tr>
<tr>
<td>1345</td>
<td>2113</td>
</tr>
<tr>
<td>2345</td>
<td>2215</td>
</tr>
<tr>
<td>12345</td>
<td>2513</td>
</tr>
</tbody>
</table>
As a conclusion we can state the following:

1. If the storage cost of database is null or negligible, the optimal locations of database copies are independent from their storage cost. But it is not recommended to store the databases at every node of the network. This will increase the cost of updates.

2. If the storage cost of database is not negligible, and if it is different at every node of the network, the optimal location of database copies is not independent from their storage cost. Therefore, it is recommended to minimize the number of copies to be stored in the distributed system.

3. If the storage cost of database is not negligible, and even if it is the same at every node of the network, the optimal location of database copies is not independent from their storage cost. The recommendation described in point 2 still holds.

Another important aspect not considered by Levin-Morgan's model is the effect of the return flow of information on the optimal location of databases. Let us investigate this effect in more detail.

C. Third Case -- Existence of the Return Flow of Information

Let us relax the assumptions made in the previous section and consider that there exist a return flow of information for queries. In our case, we consider that the return flow of information is two
times the length of the query. Of course, the model is very general and considers the return flow of information as an input parameter. The value of such parameters is organization (or application) dependent. A possible way to estimate it is by using some econometric method of estimation using past data.

For the storage cost of database, we consider two different cases:

(a) the first case is when the storage cost of database is negligible

(b) the second case is when the storage cost of database is equal to $500 per month

The results obtained are summarized in Figure 30, and, the following remarks can be made:

FIGURE 30. Cost of Database Assignments—Case 3

<table>
<thead>
<tr>
<th>Location of database copies at node</th>
<th>(a) Storage Cost Negligible</th>
<th>(b) Storage Cost $500 per month</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4854</td>
<td>5354</td>
</tr>
<tr>
<td>2</td>
<td>2556</td>
<td>3056</td>
</tr>
<tr>
<td>3</td>
<td>2766</td>
<td>3266</td>
</tr>
<tr>
<td>4</td>
<td>5064</td>
<td>5564</td>
</tr>
<tr>
<td>5</td>
<td>3981</td>
<td>4481</td>
</tr>
<tr>
<td>12</td>
<td>2694</td>
<td>3694</td>
</tr>
<tr>
<td>13</td>
<td>3042</td>
<td>4042</td>
</tr>
<tr>
<td>Location of database copies at node</td>
<td>(a) Storage Cost Negligible</td>
<td>(b) Storage Cost $500 per month</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>14</td>
<td>4206</td>
<td>5206</td>
</tr>
<tr>
<td>15</td>
<td>5091</td>
<td>6091</td>
</tr>
<tr>
<td>23</td>
<td>1770</td>
<td>2770</td>
</tr>
<tr>
<td>24</td>
<td>3636</td>
<td>4656</td>
</tr>
<tr>
<td>25</td>
<td>2793</td>
<td>3793</td>
</tr>
<tr>
<td>34</td>
<td>2922</td>
<td>3922</td>
</tr>
<tr>
<td>35</td>
<td>3033</td>
<td>4033</td>
</tr>
<tr>
<td>45</td>
<td>4893</td>
<td>5893</td>
</tr>
<tr>
<td>123</td>
<td>1968</td>
<td>3469</td>
</tr>
<tr>
<td>124</td>
<td>2970</td>
<td>4470</td>
</tr>
<tr>
<td>125</td>
<td>2901</td>
<td>4401</td>
</tr>
<tr>
<td>134</td>
<td>3180</td>
<td>4680</td>
</tr>
<tr>
<td>135</td>
<td>3318</td>
<td>4818</td>
</tr>
<tr>
<td>145</td>
<td>4443</td>
<td>5943</td>
</tr>
<tr>
<td>234</td>
<td>1986</td>
<td>3486</td>
</tr>
<tr>
<td>235</td>
<td>2007</td>
<td>3507</td>
</tr>
<tr>
<td>245</td>
<td>3039</td>
<td>4539</td>
</tr>
<tr>
<td>345</td>
<td>3180</td>
<td>4680</td>
</tr>
<tr>
<td>1234</td>
<td>2184</td>
<td>4184</td>
</tr>
<tr>
<td>1234</td>
<td>2205</td>
<td>4205</td>
</tr>
<tr>
<td>1245</td>
<td>3177</td>
<td>5177</td>
</tr>
<tr>
<td>1345</td>
<td>3456</td>
<td>5456</td>
</tr>
<tr>
<td>2345</td>
<td>2223</td>
<td>4223</td>
</tr>
<tr>
<td>12345</td>
<td>2421</td>
<td>4421</td>
</tr>
</tbody>
</table>
1. If we compare the results obtained in case 1 (page 184) and in case 3 - (a), we can see that there is no change in terms of optimal location of the database copies. In both cases, the optimal solution is obtained when the copies of the database are stored in nodes 2 and 3. This can be shown if we recall the formulation of the objective function.

\[
\min F = \sum_{d,k} C_{kd} X_k^d 
\]

\[
+ \sum_{j,p} S_{jp} Z_j^p 
\]

\[
+ \sum_{i,j,p,d} Q_{ip}^d Q_{ij} X_{ijp}^d (1 + \gamma_{ij}) 
\]

\[
+ \sum_{i,j,p,d} U_{ip}^d U_{ij} X_{ijp}^d 
\]

\[
+ \sum_{j,k,p,d} \beta_{jk}^p Q_{jk} X_{jkpd} (1 + \gamma_{jk}) 
\]

\[
+ \sum_{j,k,p,d} \beta_{jk} U_{jk} X_k^d 
\]
Since the storage costs are negligible, expressions (1) and (2) are negligible. Besides, the return flow of information does not have any effect on the updating costs. Therefore, expressions (4) and (6) are constant. If we let $(1 + \gamma_{ij}) = \gamma_1$ and $(1 + \gamma_{jk}) = \gamma_2$ for all $i,j,k$ we can rewrite expressions (3) and (5) in the following manner.

\[
\gamma_1 \sum_{i,j,p,d} q_{ip}^d q_{ij}^d x_{ijp}^d
\]

\[
\gamma_2 \sum_{j,k,p,d} a\lambda_{jp}^d q_{jk}^d x_{jp}^d
\]

Therefore, the role of $\gamma_1$ and $\gamma_2$ is the same at every node of the network. Their effect will not change the optimal location of the database as long as the query cost is bigger than the update cost. This can be obtained by minimizing the number of copies of the database to be stored. That was precisely the aim of case 1 (when we considered the return flow of information and the storage cost equal to zero). Therefore, the optimal solution will not change. Notice, however, that the optimal cost will change due to the cost of the return flow of information.

The main conclusion that can be drawn is that the optimal location of databases is independent from the return flow of information, given that the total query cost is larger than the total update cost and given that the storage cost of database is negligible.

Would this conclusion be valid if the storage cost is not negligible? To answer this question, let us compare the results obtained
in case 2 (page 187) and those obtain in case 3-(b). The only differences between both cases is that in the latter we consider the existence of a return flow of information equals two times the length of the queries. The results show that when there is no return flow of information but only a storage cost (equal to $500 a month), the optimal location of the database is at node 2. If we keep the same storage cost and we take into consideration the return flow of information, the optimal solution changes to nodes 2 and 3. This is due to the fact that if the database is stored at a single node, the incremental cost of queries due to the return flow of information is more important than the storage cost. This tends to shift the location of the database from a single to multiple nodes. As a consequence, we can draw the following conclusion: Given the existence of a storage cost for the database, their optimal location is not independent from the return flow of information.

In summary, we have been able to apply our model for distributed database systems and show that:

1. Although similar to Levin-Morgan's model, our model is more general since it assumes full-dependency between programs and databases. It is also more accurate than Levin-Morgan's model (see page 180).

2. We have shown that the existence of storage cost has an effect on the optimal location of databases.

3. For the first time in distributed database models, we have introduced the concept of return flow of information and we have shown that it has some consequences on the optimal
location of databases in the network.

4. That our solution procedure can handle additional constraints.
   (This is not the case of Levin-Morgan's procedure which can be violated by the introduction of new constraints.)

IV. Conclusion

The contribution of this chapter is to present some applications of our models and the solution procedure for the design of distributed information systems. The models developed were shown to be feasible and useful. The global model allowed us to design the distributed information system of a large bank. Our solution proved to be better in terms of optimal solution than the bank's experimental network. The model for distributed database systems performed better than past models. Some conclusions about the dependency/independency between storage cost and the optimal location of databases were indicated. For the first time, we have shown the effect of the return flow of information on the optimal location of databases. To draw all these conclusions, a mathematical programming algorithm and its computer implementation were used, and proved to be realistic. This is particularly true given the type of models (nonlinear, nonconvex models), and their sizes (the bank example has 54 variables and 87 constraints).

The computer code developed for our algorithm makes it easier for users designing and evaluating distributed systems. The users need only to enter the data related to their problems without having to deal with any programming aspects. This code, although slower than heuristics,
leads to optimal solutions which may be a critical factor in real-life examples. It costs only $6 to run the bank's example. Although it is very difficult to evaluate the cost of running a larger problem, our experience with the algorithm and its computer code allows us to set an upper bound of one minute of CPU time, to run a problem having about 600 variables.

Actually, on an IBM 370/168, it takes roughly three seconds CPU time to run the distributed database model composed of 60 integer variables and 92 constraints. To run the global model (which is more difficult since it has nonlinearities and non-convexities in both the objective function and the constraints), it takes about seven seconds CPU time. Our experience with the computer code shows that although the number of nodes grows exponentially with the number of integer variables, CPU time increases at a much slower rate. This is due to the fact that our algorithm limits a priori the number of nodes stored in the computer main memory, and refuses certain nodes without the optimization procedure.

Finally, besides its usefulness in the design of distributed information systems, our global model can be used to help solve the issue of centralization versus decentralization of information systems. This is the aim of the next chapter.
CHAPTER SIX

APPLICATION OF THE GLOBAL MODEL TO THE ISSUE OF CENTRALIZATION VERSUS DECENTRALIZATION OF INFORMATION SYSTEMS

I. Introduction

The purpose of this chapter is to investigate to what extent it is possible to use the global model developed in Chapter Three to help solve the issue of centralization versus decentralization of information systems. It is our opinion that such a model, besides its usefulness in determining the optimal configuration of distributed information systems can help bring a rigorous management science approach to the issue of centralization-decentralization. This is especially true when one considers most of the qualitative literature which is too general to draw from it any meaningful help in terms of decision regarding optimal solutions.

This chapter can be considered as an extension of Chapter Three, since it mainly expands the global model to be used in the centralization-decentralization issue. Since the main purpose of this thesis is to develop design models and solution procedures for distributed information systems, it is beyond the scope of this chapter to apply the model proposed here. This chapter is organized as follows: In Section II, we present a critical survey of the approaches used in the past, to solve the issue. The purpose of this section is twofold: first, it will help us to understand the complexity of the issue; second, it will give
II. Critical Survey of the Methods Used in the Past to Solve the Issue of Centralization Versus Decentralization of Information Systems

The issue of centralization versus decentralization of computer resources is not a new one; it has been widely discussed and hotly debated for at least two decades now. The interest in this issue was partly motivated by the feeling that such a costly expense in terms of investment and operating budget should be used to the fullest possible potential. In addition to this factor, it was becoming more and more apparent that, within a corporation, immense political power rested largely on whoever controlled the data processing facility. Lately the advances in network technology and the advent of efficient low cost mini and micro computers has brought the debut of distributed data processing and in effect thrown new fuel into the centralization/decentralization fire. Of the voluminous literature published on this subject, we first concentrate on key articles relating to one aspect of the problem: the centralization/decentralization decision. Management faced with decisions regarding proper long range directions toward optimal configurations of hardware, software and personnel find little by way of guidelines to follow. There seems then to be a real need for a rigorous decision model to provide management with an approach to solving this dilemma. Ernest Dale (1)
states: "the proper balance between centralization and decentralization often is decided by necessity, intuition, and luck because of the immense variety of possible human behavior and vast multiplicity of minute, undiscoverable causes and effects that cannot be encompassed in any principal or standard of evaluation." In addition, the solution seems highly dependent on the characteristics, philosophies, and objectives of the particular organization for which the decision is to be made. According to George Glaser (2), "the organizational approach to data processing should be consistent with the overall organizational approach of the company in which it functions." It should be becoming clear that the problem is not only of major importance but of substantial complexity also.

Having surveyed many articles available in the literature, with few exceptions most articles fit into one of the following categories:

1. general discussion of advantages and disadvantages of various configurations as viewed from a decision-making perspective.
2. establishment of decision criteria from specific corporate functions
3. proposed decision model by which management can make qualitative decisions about organizational directions based on specific data processing applications
4. discussion of distributed systems as being a new and attractive approach to the centralization/decentralization decision.

The first group of articles is very general and focusses on discussions of advantages and disadvantages of various configurations. From a functional point of view, most applications could be accomplished by either
centralized or decentralized approaches. However, as G.A. Champine (3) states, "each of the two approaches has advantages and disadvantages. In general the advantages of a centralized approach are the disadvantages of a distributed approach and vice versa." For example, some of the advantages and disadvantages he lists are:

"Centralized advantages/distributed disadvantages"
* Operations economy
* Hardware economy of scale
* Unified control
* Easy interfile communications
* Easy update/retrieval
* Compatibility

"Distributed advantages/centralized disadvantages"
* Communication failsoft capability
* Central site failsoft capability
* Lower communication data rate and costs
* Configuration flexibility
* High speed performance (fast response and high transation rate)
* Modular upgrade

Dozens of authors have written similar articles citing specific advantages and disadvantages. Some of them are:

Reynolds (4), who argued that three economic considerations have to be taken into account: personnel to operate the hardware, data processing applications programming efforts and the computing. Both considerations
can lead to some saving when centralization is chosen, which is the case of Hughes Aircraft Corporation.

Kieder (5), argues that two considerations are critical in arriving at the most effective type of organization for a particular corporation (i.e., corporate structure itself irrespective of data processing tasks performed, and size and location.)

Wofsey's (6) article is mainly a discussion of the respective advantages and disadvantages of both systems (i.e. centralized and decentralized).

Finally, Burnet and Nolan (7) argue that the technology has now matured to the stage where the cost of using a mini for certain data processing jobs compares favorably with using a portion of the capacity of a large machine.

In some articles this approach takes a more general form. Louis Fried (8) exemplifies this in his article when he states, "As part of the continuing discussion that is almost as old as the computer industry, there has been as many reasons advanced for decentralization as for centralization. However, in contrast to the arguments for centralization, which center around efficiency, the arguments for decentralization center around effectiveness." It is my contention that this first group of articles is too general and diverse to draw any meaningful generalizations from, in terms of decisions regarding optimal solution. As John Rockart, et. al. (9) state, "The articles on the advantages and disadvantages of centralization and/or decentralization abound in the literature."
Since different authors have different assumptions and approach the problem somewhat differently, their arguments are not strictly comparable."

This then brings me to the second group of articles: the discussion of centralization/decentralization in terms of corporate functions. These I believe are far more useful in that they lean in a more productive direction. Norton (10) reiterates this point by stating, "Centralization is meaningless when applied as a generality to information systems. Indeed, the concept of centralization must be approached in terms of specific functions which make up operations and management of an organization's information system." Accordingly, Norton groups information systems related activities into three categories: systems development, systems operations, and systems management. Each of these categories can be defined functionally as follows:

**Systems Development:** This includes system design, the development of detailed specs and programs, implementation plans, and maintenance plans.

**Systems Operations:** This includes the editing and control of input and output, updating data files, processing, and reporting of results.

**Systems Management:** This includes planning long range directions and projects, and maintaining control over the entire facility.

He then goes on to more rigorously define these activities and observes that the administrative planning and control tasks undoubtedly have more influence on the effectiveness and efficiency of an information system than other variables. Carl H. Reynolds (4) takes a similar approach to that of Norton's. He divides data processing facilities with regard
to "the computing hardware," "personnel required to operate the hardware," and "data processing applications programming efforts." In my opinion, these categories less rigorously define the activities of a data processing facility and are therefore less useful.

These insights into the fragmentation of the problem lead us to the third category of articles. Rockart et. al. (11), follow Norton's reasoning that activities performed by information systems are three distinct processes: systems operation, systems management, and systems development. Since each is an independent process, the decision to centralize/decentralize can be made independently for each one. The authors further segment the problem by dividing the decision with respect to applications of the facility. Their proposal is then basically that decisions to centralize or decentralize can be made separately for each of Norton's processes (system development, system operations, and system management) for each separate application of the data processing facility. Rockart's model does offer general guidelines for management to follow. In my opinion, though not rigorously solving the problem, it takes a step in the right direction. Still, Rockart relies on mainly qualitative methods of evaluation and this seems to be the most serious shortcoming to his proposal. However, his division of decisions with regard to applications opens the door to quantitative evaluation methods in determining optimal data processing configurations.

The last group of articles discusses distributed data processing as a new and promising trend in data processing structure, which could eliminate the whole centralization/decentralization problem. This is, in my opinion, far too boastful a claim. However, John Lusa (12) states,
"Some people are still arguing the comparative merits of centralizing or decentralizing infosystems activities. While the discussion goes on at a somewhat academic level, a relatively new phrase, if not necessarily representing a new concept, may keep the discussion at that level. Distributed processing has blossomed into major prominence as a technique for increasing the efficiency of a data processing operation to the benefit of the users." This new trend brought on by network technology and the advent of low-cost mini and micro computers has indeed created an appealing alternative for certain situations. Other authors such as John W. Luke (13), Richard G. Canning (14), and Tien Chi Chen (15), to name a few, take similar positions in favor of distributed data processing.

However, from a decision making aspect of the problem, this means more objective questions like: Where to put what size computer, What database to put where, and Which program will be stored where; must be addressed. A model that answers these and many other similar questions concerning optimal systems configurations is, in my opinion, the solution to some aspects of the centralization/decentralization decision problem. Distributed processing may well be the computing phenomenon of the 1980's, and it may well be the solution to some problems. However, mini computers and network technology will not solve all data processing problems.

In conclusion I would like to offer Robert L. Patrick's (16) observation, "A mini is a good solution, sometimes. Decentralization -- or distributed processing, or distributed computing, or whatever -- is a good solution sometimes, but they are only good solutions to some problems. As in most things we do, the important work is in deciding whether the solutions we like fit the problems we have."
In surveying the qualitative literature, little was found in the way of hard conclusions. The centralization/decentralization decision process is still very subjective at best. Since no two management styles are exactly alike, in the end the decision may possibly be decided on the basis of personal preference; that is, on how a particular manager "likes" to manage.

In addition to the qualitative approaches to the issue of centralization versus decentralization of information systems, there exist an emerging quantitative approach to this issue. One of the first models using this approach is due to Streeter (17). It is a very simple model which determines the optimal number of computing facilities. He assumes an economy of scale cost function proportional to the square root of the number of facilities. The main result of his model is that a company with user locations within a thousand mile radius is best served by one or two computing locations, at most. The main disadvantage of the model is that it does not consider the possibility of designing a computer network. Besides, his model does not indicate where computing facilities should be located and where databases and programs should be stored. In summary, the work of Streeter, although relevant to system operation only, is still incomplete and ignores important aspects of the issue.

Chen et. al. (18) extended Streeter's work and studied a different class of configuration. In their paper, they consider a decentralized configuration where the computers are organized in the form of hierarchical network. Their model, an unconstrained integer nonlinear programming
class of configuration. In their paper, they consider a decentralized configuration where the computers are organized in the form of hierarchical network. Their model, an unconstrained integer nonlinear programming problem, determines which users should be served by remote computers and which by a centralized facility. The objective function is more realistic than the one used by Streeter and includes individual costs.

These two models suffer from three main setbacks:

- They assume a topology
- They do not take into account constraints
- They do not incorporate important factors related to the organizational aspects of the issue of centralization/decentralization.

In summary, we can say that so far the most complete model proposed to solve this issue is the one developed by Rockart et. al. (11). Like Norton (10), they divided the information system into three subsystems: System operation, system development and system management. To facilitate the decision, they further subdivided the first two dimensions into Logical Application Groups (LAGS). They define a LAG as a complete application system and claim that it is possible to deal with the C-DC of one LAG at a time. Then they provide division and factor tables to evaluate an effective range of solutions. Although their model constitutes the most comprehensive approach to the problem so far, it suffers from several setbacks.

1. The approach taken is not rigorous. This is especially true when one considers the use of circles and squares as a scaling
2. No attempt of quantification is made. Besides the lack of a scaling tool, there is no classification method allowing a discrimination between the factors. The quantifiable factors such as those related to system operations are evaluated only by qualitative methods.

3. The evaluation of the factors is mostly subjective. For example, it is argued that "response/turn around time" is a critical factor for decentralization. It seems to us that for a given application, we can obtain the same response time (probably at a lower cost) with a centralized system provided that the centralized computer has the capacity (or is given an additional capacity) to handle all the tasks.

4. The degree and the nature of the centralization/decentralization is not indicated. Rockart's model deals only with the broad issue of whether to centralize or to decentralize the information systems. But it does not indicate the degree of centralization or decentralization. By degree of centralization/decentralization, we mean the number of computer centers and their locations, the number of corporate databases and their geographical situation. All this information cannot be obtained when Rockart's model is the only model used in the decision process.

5. Little attention is given to the costs factors. Besides, no
attempt to minimize the cost of the system (centralized or decentralized). This can be a handicap since most of the organizations are not necessarily interested in changing their configuration if there is no major savings.

6. The approach taken to look at the three dimensions (system operations, systems development, and system management) is the same (basically a decision table). This is acceptable as long as both dimensions of the information system are similar in terms of factors involved and their nature. This can be hardly the case. The factors involved in system operations are mainly technological factors, easily quantifiable and particularly suitable for optimization processes. On the other hand, the factors involved in systems development tend to be qualitative. Therefore suitable to "decision table" approach. Finally, the factors involved in system management tend to be related to management style, management philosophy and political arguments. Those types of factors can hardly be evaluated in an efficient way using a decision table.

7. The model does not indicate clearly what factors are most appropriate relatively to system operations, system development and system management.

8. Subdivision of each dimension of the information system into LAGS can lead to cost suboptimization.
9. Choosing the best alternative configuration on the basis of the cost benefit ratio may lack convincing power for non-data processing managers involved in the final decision.

It is our opinion that a reasonable solution to the issue of centralized/decentralized systems should:

- be a mixture of quantitative and qualitative approaches
- avoid the pitfalls indicated above

The aim of this chapter is to achieve such goals.

Before presenting our model, let us mention that an updated table of the advantages and disadvantages of centralized and decentralized information systems is given in Appendix B. This can help the reader to fully understand the complexity of the issue.

III. The Decision Support Model

A. Introduction

As proposed by Norton (10) and already accepted by Rockart et. al. (11), we divide the information systems function into the three activities that it performs: systems operation, systems development and systems management. Decisions to centralize or decentralize can be taken independently along the three dimensions. But, since these dimensions are different in nature and involve different factors, the methods to facilitate the decisions along each dimension should be different. We do not propose to subdivide the centralized/decentralized decision into independent LAG's as proposed by Rockart, since it can lead to cost
suboptimization. As we stated earlier, although the factor "cost" is not the most important, the method used by Rockart to select the "best" alternative (i.e. on the basis of the cost/benefit ratio) lacks convincing power for non data processing managers.

It is our opinion that the C/DC decision should, at the final state, be a subjective process in which many qualitative factors should be considered. But, there are some cases where quantitative models can be useful. For example, systems operation is suitable for the application of this type of technique. Systems operation involves mainly cost factors; therefore optimization techniques can be used. This type of optimization model can be used as a tool to evaluate alternative configurations and to design the optimal configuration. In this chapter, we propose the following approach to facilitate the C/DC decision.

1. We divide the information system in three dimensions: system operation, system development and system management

2. Since system operation is mainly quantifiable, we use a revised version of the optimization model described in Chapter Three.

3. We will develop a decision table with only the factors related to system development and define a classification method for this decision table that can help in the decision-making process.

4. For system management, a number of rules taking into account the characteristics of the corporation involved will be presented.

By using this approach, we simplify the decision making process for each system, and we use adequate methods for each specific subsystem.
The remainder of this section is a detailed version of the method described above.

B. System Operation!: An Optimization Model

A careful analysis of the different factors associated with system operations, can convince one that they can be quantifiable. In fact, system operations consist of few sub-processes: edit and control, updating, processing and reporting. These sub-processes are cost oriented and can be integrated in an optimization sub-model.

The aim of such a submodel is to determine at the lower cost the optimal configuration, including the allocation of the personnel, computers, databases, programs, communication lines and routing disciplines. No prior assumption is made about the nature of the configuration. Depending on individual cases, the configuration provided by the model can be either centralized or decentralized optimally. The model described below is in fact an extension of the first model developed in Chapter

The components of the model are the following:

- Personnel cost
- Hardware equipment cost
- Database cost
- Programs cost
- Communication line costs
- Breakdown cost

Let us first describe the objective function representing the overall cost that should be minimized.
1. **Equipment and Personnel Costs**

Let \( f_m \) be the cost of equipment and personnel, and \( f_m = (C_m + C_{l_m} + M_m + S_m + CO_m - I_m) \), where:

- \( C_m \) = cost of computer \( m \) where its capacity is \( K_m \)
- \( C_{l_m} \) = set-up cost
- \( M_m \) = maintenance cost
- \( S_m \) = salaries of personnel involved in running the center \( m \). It may include analysts, programmers, etc., costs and salaries
- \( CO_m \) = cost of commodities attached to equipment \( m \) (space, electricity, etc).
- \( I_m \) = cost of existing computer (if any) at center \( m \) before the change (\( I_m = 0 \) if no equipment exists.) This allows to take into account existing equipment since most organizations do not start from scratch.

The cost allocation of personnel and equipment is:

\[
Z_1 = \sum_{i,m} f_m y_{i,m}^m
\]

where

\[
y_{i,m}^m = \begin{cases} 
1 & \text{if equipment } m \text{ is allocated to node } i \\
0 & \text{otherwise}
\end{cases}
\]

2. **Database Cost**

Let \( f_n \) be the cost of databases, \( f_n = (C_{in} + D_{ln} + D_n - IO_n) \) where:

- \( D_n \) = set-up cost of database \( n \) (averaged over all nodes)
- \( D_{ln} \) = organizational cost and maintenance cost of database \( n \)
The cost allocation of databases is, therefore, equal to:

\[ Z_2 = \sum_{i,n} f X^n_{i,n} \]

where

\[ X^n_i = \begin{cases} 1 & \text{if database } m \text{ is allocated to node } i \\ 0 & \text{otherwise} \end{cases} \]

3. **Cost of Communication Lines**

Let \( f_c \) be the cost of communication lines.

\[ f_c = (B_c + MB_c + (BO_c * D_{ij}) - IC_c) \]

where:

- \( B_c \) = fixed cost of installing a communication line of capacity \( c \)
- \( BO_c \) = variable cost per unit of distance for installing a communication line of capacity \( c \)
- \( MB_c \) = maintenance cost for communication line of capacity \( c \)
- \( IC_c \) = cost of existing communication lines
- \( D_{ij} \) = matrix of distances between nodes \( i \) and \( j \)

The cost of communication lines is, therefore:

\[ Z_3 = \sum_{i,j,c} f_c L^c_{ij} \]

where

\[ L^c_{ij} = \begin{cases} 1 & \text{if a communication line of capacity } c \text{ is nodes } ij \\ 0 & \text{otherwise} \end{cases} \]
4. **Programs Cost**

Let $f_p$ be the cost of programs.

$$
 f_p = (S_{ip} + SC_p + SM_p - IS_p)
$$

where:

- $S_{ip} = \text{storage cost of program } p \text{ at node } i$
- $SC_p = \text{set-up cost of program } p \text{ (creation cost, averaged over all nodes)}$
- $SM_p = \text{maintenance cost of program } p$
- $IS_p = \text{cost of existing program}$

The allocation cost of programs is:

$$
 Z_4 = \sum_{i,p} f_p Z_i^p
$$

where

- $Z_i^p = \begin{cases} 
 1 & \text{if program } p \text{ is allocated to node } i \\
 0 & \text{otherwise}
\end{cases}$

5. **Communication Costs**

The communication costs of queries and updates are exactly the same as those described in Chapter Four and are equal to:

$$
 Z_5 = \sum_{i,j,p,d} Q_{ip}^d Q_{ij} X_{ijp}^d (1 + \gamma_{ij}) \text{ communication costs of queries from nodes to programs}
$$

$$
 + \sum_{i,j,p,d} U_{ip}^d U_{ij} X_{ijp}^d \text{ communication costs of updates from nodes to programs}
$$

$$
 + \sum_{j,k,p,d} \alpha_{jpd} X_{jkd}^d (1 + \gamma_{jk}) \text{ communication costs of queries from programs to databases}
$$
+ \sum_{j,k,p,d} \beta_{jk} U_{jpd} X_{kd} \quad \text{communication costs of updates from}
\text{programs to databases}

where:

\( Q_{ip}^d \) = query traffic from node i to database d via program p

\( Q_{ij} \) = communication cost per query unit from node i to node j

\( U_{ip}^d \) = update traffic from node i to database d via program p

\( U_{ij} \) = communication cost per update unit from node i to node j

\( \alpha \) = expansion factor for queries

\( \chi_{jkpd} = \begin{cases} 1 & \text{if transactions from node } j \text{ to database } d \text{ are rerouted to node } k \\ 0 & \text{otherwise} \end{cases} \)

\( Y_{ij} \) = return flow of information from program located at node j to user located at node i

\( \lambda_{jp} = \sum_{i} Q_{ip}^d \chi_{ijp} \) = query traffic to database d processed at node j

\( \beta \) = expansion factor for updates

\( \mu_{jp} = \sum_{i} U_{ip}^d \chi_{ijp} \) = update traffic to database d processed at node j.

6. **Breakdown Costs**

In order to assure some reliability, we consider three different breakdown costs:

\[ Z = \sum_{i,m,p,d} [C_{ip} (Q_{ip}^d + U_{ip}^d)] y_i \] 

\( y_i \) = cost for computers breakdown
\[
\begin{align*}
+ \sum_{i,j,c,p,d,m} [C_2 P_1^c (Q_{ip}^d + U_{ip}^d)] L_{ij}^c & \text{ cost for communication lines breakdown} \\
+ \sum_{i,n,p,d} C_3 P_d (Q_{ip}^d + U_{ip}^d) X_{i}^n & \text{ cost for database breakdown}
\end{align*}
\]

where:

- \( C_1 \) = constant of proportionality (which may include organizational "breakdown" costs, such as lost orders, etc.)
- \( P_m \) = probability of breakdown of computer \( m \)
- \( C_2 \) = constant of proportionality
- \( P_1^c \) = probability of breakdown of communication line \( l \) of capacity \( c \)
- \( C_3 \) = constant of proportionality
- \( P_d \) = probability of breakdown of database

The set of constraints is exactly the same as the one described in Chapter Three, and is given below.

\[
\begin{align*}
\sum_{j} L_{ij}^c & \geq 1, \forall i \in I = K, c \in C & \text{Existence of communication lines for users} \\
\sum_{j} X_{ijp}^d & = 1, \forall i \in I = K, p \in P, d \in N & \text{Assure that every transaction to every database and computer and from every node will have a defined route, but only for programs related to databases}
\end{align*}
\]
Assure residency of databases, programs and computer in accordance with the defined routes. \(\sigma\) is equal to the number of nodes.

\[
\begin{align*}
\sum_{i,j,p,d} \alpha x_{ip,jp}^d x_{jkpd}^d & \leq \sigma \sum_{j} z_{p}^d \\
\sum_{j,k,p,d} \beta u_{ip,kp}^d x_{ijp}^d & \leq \sum_{i,p,d} \sigma x_{k}^d
\end{align*}
\]

\(\forall i \in I = K, \forall j \in J = K,\)

Total processing requirement does not exceed the node capacity

\(x_i^n < \sum_{m} y_i^m\) \hspace{1cm} Existence of database with respect to computer

\(z_i^p < \sum_{m} y_i^m\) \hspace{1cm} Existence of program with respect to computer

\[
\begin{align*}
\sum_{r,p,d} \alpha q_{rp}^d (1+\gamma_{rij}) x_{rip}^d x_{ijpd}^d & + \sum_{r,p,d} \alpha q_{rp}^d (1+\gamma_{rij}) x_{rip}^d x_{ijpd}^d + \sum_{r,p,d} \beta u_{rip}^d x_{rd}^d \\
+ \sum_{r,p,d} \beta u_{rip}^d x_{rd}^d & + \sum_{p,d} q_{ip}^d (1+\gamma_{ij}) + \sum_{p,d} u_{jp}^d x_{jp}^d + \sum_{p,d} (1+\gamma_{ij}) + \sum_{p,d} u_{jp}^d x_{jp}^d \leq \sum_{c} c_{ij}^d
\end{align*}
\]

for all \(i \in I = K\) and \(j \in J = K\) with \(j > i\)

\[
\begin{align*}
\sum_{k} x_{k}^d & \geq 1, \hspace{1cm} \forall d \in N \\
\sum_{j} z_{j}^p & \geq 1, \hspace{1cm} \forall p \in P
\end{align*}
\]
Binarary variables: $v^m_i, x^n_i, z^p_i, L^c_{ij}, x^d_{ijp}, X_{jkpd}$

This optimization model, although taking into account most of the factors involved in system operation, is not larger in terms of number of variables than the one described in Chapter Three. It has exactly the same structures as the global model, but incorporates more parameters related to system operations.

By solving it, a user can obtain either a centralized or decentralized configuration depending on its input parameters.

When used, the model can indicate whether to centralize or decentralize and the degree of centralization or decentralization.

It indicates the optimal allocation of personnel and equipment, the assignment of databases and computer networks, the type of communication network and the message routing discipline.

Finally, it helps to compare different configurations and evaluate the benefits of them.

Let us now turn to the second subsystem: system development.

C. System Development: A Decision Table

As defined by Norton, system development is the process of designing and implementing new computerized information systems. System development includes four major sub-processes: functional design; detailed specs and programming; implementation; and maintenance. As pointed out by Rockart: "It is not necessary that each sub-process be centralized, distributed or decentralized to the same extent. Yet these sub-processes
are interrelated and must be considered in conjunction with each other."

In order to facilitate the C/DC decision, we first isolate the factors directly related to system development. Then, we define a ranking method that should be used when the decision table relating factors to sub-processes is established. This ranking-method leads us to a rule that can be used to decide which subprocesses should be centralized or decentralized.

(i) The Factors

The aim of this section is to isolate the factors that are directly related to system development. The assumption is that, although most of the factors described in Rockart's paper are relevant, only some of them are the most influential in system development. For example, the factor "response time", although relevant, is critical only for system operation and not for system development. This discrimination of factors allows us to supply the decision table and to relate to the subprocess only the factors that are critical to them, thus making the decision more accurate. A careful analysis of all the factors involved, leads us to retain only the following:

- Profit and loss responsibility
- Diversity of industries
- Geographic locations
- Current status of DP
- Current status of system development group
- Degree of specialization of organization's subunit
- Size of subunit
- Experience with DP
- Types of application (old, new, scientific, etc.)
- Integration of functions critical
- Degree of DP expertise required
- Geographical centers of database or files
- Degree of interaction between applications

The reader should be aware that for specific organizations additional factors may be needed. In such cases, those factors can be added to the original ones. When we regroup the sub-processes and the factors, we obtain from the decision-table on the following page.

Since most of the factors listed above are subjective and organizational dependent, it is not useful to indicate, as Rockart does, the strength of each factor. Instead, we let each company evaluate the factors using the following scale:

1. = dominant argument for decentralization
2. = moderate argument for decentralization
3. = indifferent
4. = moderate argument for centralization
5. = dominant argument for centralization
<table>
<thead>
<tr>
<th>Factors</th>
<th>Functional Design</th>
<th>Programming</th>
<th>Implementation</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profit and loss responsibility</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diversity of industries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geographic locations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current status of DP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current status of SD group</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degree of specialization of organizational subgroup</td>
<td></td>
<td></td>
<td>Pij</td>
<td></td>
</tr>
<tr>
<td>Size of subunit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experience with DP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Types of application</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration of function critical</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degree of expertise required</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geographical location of databases or files</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degree of interaction between applications</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Let $P_{ij}$ be the evaluation given by the company for subprocess $j$ using factor $i$. Let $\text{Max}_{j} (P_{ij})$ be the "best" value for factor $i$. We calculate the following expression:

$$R_{ij} = \frac{P_{ij}}{\text{Max}_{j} (P_{ij})}$$

If $n$ is the number of factors used, we can have a ranking for subprocess $j$ using the following formula:

$$R_{j} = \frac{1}{n} \sum_{i=1}^{n} \frac{P_{ij}}{\text{Max}_{j} (P_{ij})}$$

The decision to centralize or decentralize a given subprocess $j$ will be taken using the following rule:

**RULE:**

If $R_{j} \geq \frac{\bar{n}}{n}$, all the subprocesses $j$ must be centralized.

If $R_{j} < \frac{\bar{n}}{n}$, all the subprocesses $j$ must be decentralized.

Where $\bar{n}$ is equal to the median of the scale used (in our case, $\bar{n} = 3$).

Both the decision table and the rule are easy to use.

The table can be used to compare divergent opinions in the same company, therefore evaluating different alternatives for each subprocess. The table can be expanded and the scale can be changed according to
company preference. Finally, we did not put "political" factors in the
decision table because we feel that:

- there is a lack of solutions to this issue
- it outweighs all the factors described.

Finally, let us analyze the most important subsystem: system manage-
ment.

D. System Management: Guidelines

With Glaser, we believe that the most important aspect of information
systems is the system management process since any decision related to it,
can have a long-lasting effect in the overall information system. It
includes two aspects:

- management control
- planning

Fried [20] pointed out that management control and information systems
consist of:

- Monitoring budgets and performance
- Auditing progress on major projects
- Applying management guidelines to the selection of major projects
- Lowering conflicts in costs allocation
- Avoiding dispersing responsibility
- Controlling EDP in the areas of costs, use of resources and
effectiveness.

Planning activities consist of:

- Maintaining a concise description of the current status of EDP
  systems, hardware, personnel, costs, etc.
- Gathering corporate and divisional systems requirements and priorities
- Developing an annual systems plan that is consistent with the resources available to accomplish the desirable project
- Reviewing the system plan for potential impact on hardware capacity and staffing
- Advising top management of the alternatives available for achieving planned objectives.

These important functions can explain why the system management centralization-decentralization decision is the most critical. As pointed out by Rockart: "A too hasty decision with regard to the locus of system management can lead to long-lasting organizational effects."

This aspect involves management style, psychology of people, politics, and other subjective factors. It is our opinion that quantitative models and decision tables are not adequate to decide whether to centralize or decentralize the management of information systems.

Glaser's rule is very useful. He stated that: "There should be internal consistency between the several organizational philosophies and the organization's approach to data processing." But, this principle is too broad and cannot be efficiently used in real situations. We suggest the following principles to achieve an effective decision for system management.

1. **Principle one: Size of the organization**

   For very small organizations, decentralization is impractical and should be avoided. For very large organizations, especially those functioning as loss and profit centers, centralization is impractical
and should be avoided.

2. **Principle two: Type of organization's industry**

There are mainly three types of organization's industry:

- **Single Industry**: It means that the organization is specialized in a single industry. For example, Polaroid.

- **Related Industries**: It means that the organization is present in different but related industries.

- **Unrelated Industries**: This is the case of an organization practicing the art of diversification. For example, the Tenneco Corporation has eight different and unrelated activities: oil and natural gas, packaging, farm and construction equipment, auto exhaust systems, chemicals, fruits and vegetables, natural gas pipelines and shipbuilding.

For an organization with several independent divisions (such as conglomerates) and being in unrelated industries, centralization is undesirable. The reason is that usually these kinds of conglomerates have decentralized management and it would be awkward just to centralize their information systems. This also will contradict Glaser's rule. For organizations with related industries, centralization is not recommended if the management of the divisions is decentralized.

The case of an organization with a single industry and all the other ones not mentioned above are studied below.

3. **Principle three: Remaining cases**

Above are mentioned some recommendations for very small, very large or very centralized organizations. For all the other cases, and given
the actual trend, it is our opinion that those organizations should create a centralized system management group. This group, if centralized, will be able to perform the tasks of planning and control described above. This is not to say that all system management must be centralized. For example, the overall annual planning and control for the organization can be performed by this group, which in turn can delegate to the divisions the authority to control the daily activities of the information system.

IV. CONCLUSION

In this section we presented a decision support model related to the issue of centralization versus decentralization of information systems. The main advantage of the model is that it incorporates both qualitative and quantitative aspects of the issue. In the past, researchers tended to take into account only the qualitative or the quantitative aspects of the C-DC issue. By decomposing the information system in three components (i.e. system operations, system development and system management) and by applying to each component a specific sub-model (i.e. optimization for system operations, decision table for system development and guidelines for system management), we facilitate the decision-making process for organizations facing this issue. It is beyond the scope of this thesis to apply this model in real-life examples. It is obvious that we do not pretend that the decision support model presented in this chapter is the final answer to the issue of centralization-decentralization of information systems. More work is needed to refine, test,
validate and apply it. But it can be seen as an alternative model to Rockart et. al.'s model. Rockart's model can be characterized as a macroscopic model, whereas our model captures the same effect as Rockart's model but in more detail and precision with regard to the degree and the nature of the centralization or decentralization.

It is our opinion, that although these principles can be applied, it is the responsibility of the top management, accordingly with its management style, philosophy and other subjective factors, to decide whether to centralize or decentralize the system management of information systems.

In summary, we have shown how our global model can be expanded to be used in the issue of centralization versus decentralization of information systems. Other expansions of the model are discussed in the next chapter and can serve as a good basis for further research.
Chapter Six

(5) KIEDER, S.P., "Once again, Centralize or Decentralize," Info-systems, December 1976
(8) FRIED, L. "Centralization: to be or not to be," Infosystems, January 1976
(11) ROCKART, J.F., J.S. LEVENTER, and C.V. BULLEN, "Centralization


(13) LUKE, J.W., "Unravelling the Confusion of Distributed DP," Infosystems, December 1976

(14) CANNING, R.G., "Is Your Future Distributed Systems?", EDP Analyzer, Vol. 11, No. 8, August 1973

(15) CHEN, C.T., "Distributed Intelligence for User Oriented Computing," AFIPS, Vol. 41, Fall 1972

(16) PATRICK, R.L., "Decentralizing Hardware and Dispersing Responsibility," Datamation, May 1976


(20) FRIED, L., "Centralization: To be or not to be," (Revised 7/76) 1-01-01, Auerbach Publishers, Inc., 1976
CHAPTER SEVEN

SUMMARY, PERSISTING PROBLEMS AND
DIRECTIONS FOR FURTHER RESEARCH

I. Summary

This dissertation extends the state of the art of distributed systems. The problem of optimal design of distributed information systems has been studied. A rigorous mathematical programming algorithm was presented. A computer code for the algorithm was developed. An application of the models and the algorithm was performed using data from a real-life setting. Finally, a decision support model for the issue of centralization versus decentralization of information systems was presented.

Let us analyze in more detail our contribution.

A. The Design and the Optimization of Distributed Systems

So far, much of the research done in this area can be characterized by a piece-meal approach. When designing a distributed system, most of the authors did not fully take into account most of the aspects of distributed databases systems. They mainly concentrated on the issue of computer network design. On the other hand, when they studied the issue of distributed database systems, they assumed the existence of a fully connected network.

The piece-meal approach can lead to suboptimal configurations. In this thesis, a model is developed and encompasses most of the aspects of
distributed systems, including the optimization of computer networks and
distributed database systems. When used, the model can lead to an
optimal design of distributed systems, characterized by:

- an optimal allocation of computers over the network, including
  the capacities of such computers
- an optimal allocation of the databases over the network
- an optimal allocation of the programs operating on the data-
  bases of the network
- an optimal allocation of the communication lines and their
  capacities
- an optimal routing discipline to be used to transmit queries
  and updating requests in the computer network

To our knowledge, this is the first time that such a complete
model is developed. A method to derive a model for the design of
distributed database systems only is described. This leads to a model
which determines the optimal database/programs locations in the network
and the optimal message routing disciplines. This model includes also
the aspects related to the return flow of information in the network.
The results showed that this aspect can have an influence on the optimal
locations of databases and programs.

B. The Solution Procedure

So far, most of the authors that have been studying the design
problem of computer networks and distributed database systems, used
heuristics methods to solve their models. Most of the time, no proof
of convergence of these methods are given. Very seldom, some indications
about how these methods are operational are given. Finally, most of
these heuristics lead to a suboptimal solution. As a consequence, the
results obtained when using these methods in real-life applications can
be suboptimal, therefore leading to a very costly solution. To avoid all
these disadvantages, we developed a pure mathematical programming algor-
ithm that can allow to obtain an optimal solution. Since our models
contain non linearities and integer variables, the algorithm is a non-
linear integer mathematical programming procedure. No assumptions are
made about the convexity of the objective function. Although this
algorithm is slower than the traditional heuristics, we made the assump-
tion that companies prefer a real optimal configuration even if it takes
minutes of CPU time to obtain it. In fact, the cost to run the computer
code is very marginal for the companies, but the economies obtained can
be very important in real-life applications. The main characteristics
of the method are:

- The number of nodes of the arborescence that have to be stored
  in the computer is at most equal to n, where n is equal to the
  number of variables of the problem. Therefore, we can solve
  large-scale problems.

- It can solve integer problems where the objective function and
  the constraints are nonlinear, without any assumptions about
  the convexity of the problem. Therefore, we can solve real-life
problems.

- The CPU time needed to solve large problems seems to be reasonable

- It has the property of heuristic methods, in that a rapid solution can be obtained in a very reasonable CPU time.

C. The Computer Code

A fortran computer code of this method was programmed and tested. Besides, an interface with the model was built, thus facilitating the use of the computer code. The user is asked to enter only the data related to its problem, without having to care about the description of the model, and the program (about 4000 thousand Fortran statements). To our knowledge this is the first time that a computer code for integer nonlinear programming problems has been proposed, tested, and used for the design of distributed information systems.

D. Real-life Applications

The global optimization model for computer network and distributed database systems has been applied, using data of one of the largest banks in Europe. The results obtained show that the present experimental network used by the bank is not optimal. The derived model for distributed database systems was applied using the same set of data as Casey and Levin-Morgan. The results show:

- that our algorithm performs better than the algorithms of the authors cited above. The true optimal solution is obtained only
by our solution procedure

- The introduction in our model of the return flow of information has a definite consequence on the optimal locations of the databases. This aspect was not taken into account by other authors.

E. The Issue of Centralization Versus Decentralization of Information Systems

So far, other research in this area has focused on the comparison of the advantages and the disadvantages of respectively centralized and decentralized information systems. But, this previous research did not offer guidelines that can help management to face effectively the decisions regarding proper long range directions. An exception to this literature is the decision-model developed by Rockart et. al. But, this model suffers from a lack of rigorous formulation. Besides, the model is too broad to be really operational. Finally, it does not indicate the type and the degree of centralization or decentralization. In this thesis, an alternative model was presented. It divides the information system in three components: systems operations, systems development and systems management. Decisions to centralize or decentralize can be taken independently along the three dimensions. To facilitate the decision for the system operation components, we provided an optimization model which can lead to optimal centralized or decentralized configurations. For the system development component, we proposed a decision table relating each sub-process of the system to the factors involved. A
ranking method facilitating the decision process was described. Finally, several guidelines pertaining to system management were proposed. These guidelines take into account the type and the size of the organization facing the issue of centralization/decentralization. Our model, by dividing the information systems in three different components, and by using the proper tool for each component, allows the decision-maker to evaluate different possible solutions.

In summary, our models and the solution procedure have extended the state of the art of distributed management information systems. Two important aspects of distributed management information systems were given particular attention:

- the design, modeling and optimization of distributed systems including computer networks and distributed database systems
- the solution procedure and its application to real-life examples

To some of the problems mentioned in our state of the art study, we proposed specific solutions, real-life oriented. Although we showed that the future trend is mainly toward distributed systems, we do not claim that all the problems inherent to such systems have been solved. There are some persisting problems in the area of distributed systems; the following section is devoted to these issues.
II. Persisting Problems in Distributed Information Systems

A. Introduction

In Chapter Two, we showed that a number of computer networks and distributed database systems are now in operation. This means that the technical know-how for using distributed systems exists today. In Chapter Six, we also showed that some managerial issues are solved. Although there exist some technical issues still unsolved, little attention has been devoted to issues such as regulatory, economic, legal and social issues. In general, little attention has been devoted to the non-technical issues in distributed information systems. As stated by Harslem and Heafner (1): "There is no question but that the social, political and legal problems (rather than the technical one) will delay the coming of the computer utility network." The aim of this section is not to propose solutions to these issues. This is beyond the scope of this thesis. But we rather describe them and give adequate references to the reader interested in such issues.

B. Some Persisting Problems

1. Social Issues

The area covered by social issues is very large. As defined by Enslow (2): "The social issues are those that create any important cross influences between users and non users of communication networks." An example of a social issue is the data traffic which has a definite impact on other users of the communication systems.
Another important social issue is privacy. Privacy can be defined as the right to keep certain types of information confidential and for private use only. This leads to a discussion of legal issues.

2. Legal Issues

One factor that has an important impact on the use of computer networks is privacy of information. It is clear that this aspect was inadequately taken into account by computer network designers.

Another major aspect of legal issues is the one related to the regulatory environment of commercial networks.

Finally, security of databases and files is an issue of paramount importance. It is our opinion that it should be included in the basic system design.

3. Economic Issues

One topic that needs more investigation is communication cost. Recent development, such as specialized common carriers and value-added networks catering to data transmission customers, make the design problem of distributed systems even more difficult. Often overlooked is the cost of the database and its use. A challenging problem is how to charge the customer who uses networks. It is still very difficult to design an accounting system for heterogeneous distributed systems. Additional thought must be given to obtaining consistent charges for executing the same program under different loading conditions.
4. Management Issues

Some of the management issues related to distributed systems have been studied in this dissertation. Still, there are some persisting problems, such as those relating to the operational management of distributed systems. Stefferud (3) mentions five important points:

"1. Communications networks facilitate large-scale sharing of computer facilities across major organizational boundaries, with the result that difficult new problems are being forced to the attention of management at all levels.

2. Many organizations have grown dependent on their computer facilities. Now with networks, that dependency will be shifting to foreign (outside) computing facilities. Organizations are undergoing power structure shifts which will threaten management with rethinking their organization structures.

3. The management problems of sharing have not been solved in pace with technical network developments. Sharing is now feasible but the management problems are unsolved. A substantial effort is required to find some way for organizations to afford the risk of becoming dependent upon foreign facilities if networking is to become acceptable.

4. The role of the technician must be fully understood in relationship to the politics of the power structure shifts that are the result of networking developments.

5. Finally, what is the role of top management in evaluating or directing the use of a network?"

In the last five years, some satisfactory answers have been given to some of the problems described above. The best source of information can be found in (4) and (5). Some additional information can be found in (6) and (7). But, much of the two latter are devoted to still completely unsolved questions, such as:

- reliability of computer networks
- Reliability of computer networks
- Integrity of distributed database systems and communication networks
- Security in distributed systems
- Currency control

Some recent developments can be found in (8) and (9). As pointed out by Emery et al. (10):

"Earlier efforts at resource sharing were strongly inhibited by technical and economic considerations. Although these issues are still important, many potential applications have now become perfectly feasible from both a technical and economic standpoint. The fact that much of this potential is still unrealized suggests that there are other restrictions limiting network sharing.

Habit, inertia, and lack of suitable incentives are all partly responsible for the relatively low volume of network sharing that does take place. Perhaps the single most important inhibitor, however, is the difficulty experienced by most persons trying to use a remote resource. The documentation that exists is very often of poor quality, and personal assistance is frequently hard to obtain. These deficiencies in user services must be corrected before widespread sharing can be expected to occur. High quality user services can do much to break down resistance to network sharing and, in turn increase the size of the market so that suppliers are motivated to improve their services still further."

Without any doubt, these issues are going to be the subject of further research.

III. Directions for Further Research

To complete our understanding of distributed systems, endless research can be directed towards possible answers to the issues described in Section II of this Chapter.
In the area of management issues in distributed systems, the first step would be to validate and refine the models of centralization versus decentralization. An extensive application of this model can lead to clarify the most important factors that should be taken into account in the final decision. Further research in this direction is very promising and the author, himself, is very interested in incorporating these elements in the models.

Another area of research would be to refine the models of computer networks and distributed database systems. The models can be more robust by incorporating reliability factors. A broad area of research can be opened by "what if" types of questions. This may include "what if" questions with regard to computer, databases and programs compatibility and pricing schemes. By using sensitivity analysis of response time and pricing schemes, some insight to the managerial issues of networks can be gained. Further research can be done to understand to what extent the two models can be used in an interactive design process, thus, helping the designer to study different configurations. When the scale of the distributed systems is very large, one can use the models for regional sub-optimization and then use the results in a global optimization process. This area of research seems to be very promising. It can help solve hierarchical distributed systems.

An entire area of research would be to optimize the computer codes needed to solve our models. One can use our code as an interactive design system allowing the users to evaluate different configurations, different pricing schemes and different sets of parameters. That would allow use
of our model from two different viewpoints: designer and user viewpoints. From the user's point of view, these models and the computer code can be applied as a tool for decision support systems. It can help the user to decide where to store databases, programs and how to design a computer network. From an organization's point of view, the models can be used as a tool to help decide whether to have its own distributed system or to join commercial or private networks. When used in conjunction with the centralization-decentralization model, the two models of Chapter Three can help organizations define their information processing needs and tools.

Finally, from the designer's point of view, these models can be used to evaluate the costs and benefits of centralized and decentralized systems. As the reader can see, a wide range of further research is available and needs to be looked into.

In summary, the models we outlined can give more flexible and effective control to the managers dealing with the issues of designing and operating distributed systems. In particular, the models developed can allow him to test his judgements and the consequences of the assumption he used before the implementation stage.

(2) ENSLOW, P.H., "Non Technical Issues in Network Design -- Economics, Legal, Social and Other Considerations," COMPCON, 73.


APPENDIX A

MATHEMATICAL DESCRIPTION OF STEP 3
OF OUR ALGORITHM
STEP 3 -- The Binary Bounded Branch and Bound Method

Let us recall the formulation of our problem:

\[
\begin{align*}
\text{Min } & \quad \phi(x), \quad x \in \mathbb{R}^n \\
\text{subject to: } & \\
& h_i(x) \leq 0 \quad i = 1, 2, \ldots, m \quad (c-1) \\
& 0 \leq x_j \leq 1, \quad j \in J = 1, 2, \ldots, n \quad (c-2) \\
& x_j = 0, 1 \quad j \in E = J
\end{align*}
\]

a) Brief Outline of the Method

Let \( S = \{ x \in \mathbb{R}^n | x \text{ satisfies (c-1) and (c-2)} \} \)

a) By solving the following problem:

\[
\begin{align*}
(P_o) \quad & \text{Max } \phi(x) \\
& \text{s.t.} \\
& \quad x \in S
\end{align*}
\]

we obtain \( x^o \) the optimal solution of \((P_o)\) (if it exists)

b) If \( x^o \) is integer: END

c) Otherwise, \( \exists j \in E, |x^o| \) is not integer. We proceed to the separation of \( S \)

into two subsets \( S_1 \) and \( S_2 \) such that:

\[
S_1 = \{ x \in \mathbb{R}^n | x \text{ satisfies (c-1) and (c-2) and} \\
\quad x \in (a, \lfloor x^o \rfloor_{j_1}, b_{j_1}) \}
\]

\[
S_2 = \{ x \in \mathbb{R}^n | x \text{ satisfies (c-1) and (c-2) and} x \in (\lfloor x^o \rfloor_{j_1} + 1, b_{j_1}) \}
\]

\( (\lfloor x^o \rfloor_{j_1}) \) means the integer value of \( x^o \). To \( S_1 \), we associate the following problem.

\[
(P_1) \quad \left\{ \begin{array}{l}
\text{Max } \phi(x) \\
\text{s.t.} \\
\quad x \in S_1
\end{array} \right.
\]
To $S_2$, we associate problem $P_2$:

\[
(P_2) \begin{cases}
\text{Max } \phi(x) \\
\text{s.t.} \\
x \in S_2
\end{cases}
\]

Then, we solve one of the problems and put the other one in a list.

d) If all the $(x_i)_{i \in E}$ are integer, go to e. Otherwise go to c.

e) If this solution is better than the first one, store it. Otherwise, refuse the node.

If there is a problem in the list, solve it and go to d. Otherwise: END

b) The Oriented Graph Associated with the Problem

Consider a set $A$ such that $A \subset E$. Let $x_A = x_j | j \in A$ and $\bar{x}_A$, the integer components of $x_A$ satisfying (c-2). To $S=(A, \bar{x}_A)$ we associate the following problem:

\[
P(S) \begin{cases}
\text{Max } \phi(x) \\
h_i(x) \leq 0 \quad i \in I \\
0 \leq x_j \leq 1 \quad j \in J \\
x_j = \bar{x}_j \quad j \in A
\end{cases}
\]

Let $\hat{x}_S$ and $\hat{\phi}_S$ be the optimal solution of $P(S)$.

a) The couple $S=(A, \bar{x}_A)$ represents a node of the graph $G$.

b) To each node $S$, we associate its "level" in the graph, called $t_S$.

$t_S$ is equal to the number of components of $x$ which are integers.

c) If $A=\emptyset$ (we have only (c-1) and (c-2)), we solve the continuous nonlinear programming problem. The correspondent node $S_0$ is called the "root" of the arborescence.

d) When $E(S) = E$, (where $E(S) = \{ j \in E | x_j(S) \text{ is integer} \}$) constraint (c-3) is
therefore satisfied. We have found a solution to problem (P).

The corresponding node is called "terminal node".

e) If \( S \) is not a "terminal node", let's consider the variable

\[ \beta \in E(S) \] and let us define a successor \( T \). The level of \( T \) is \( t^*_s + 1 \).

\( T \) is defined by the couple \( (B, x^\gamma) \) where:

\[
B = E(S) \cup \beta \\
\hat{x} = x_j \text{ w. } \forall \in E(S) \\
j \\
\hat{x}_\beta = [x_\beta(S)] \text{ or } [x_\beta] + 1
\]

f) In general, if \( S = (A, \hat{x}_A) \) is not a terminal node, we consider the nodes \( T \) (with level \( t^*_s + 1 \)) defined by:

\[
\beta \in E(S) \\
T = (B, \hat{x}_\beta) \\
B = E(S) \cup \beta \\
\hat{x} = x_j \text{ w. } \forall \in E(S) \\
j \\
\hat{x}_\beta = [x_\beta(S) + \gamma]
\]

i) If \( \gamma = -1, -2 \), the nodes \( T \) are in the left wing of the graph. The origin of the left wing is the node corresponding to \( \gamma = -1 \).

(ii) If \( \gamma = 0, 1, 2 \), the nodes \( T \) are in the right wing. Its origin is the node corresponding to \( \gamma = 0 \).

g) \( T \) is called the successor of \( S \) if \( ST \) is an arc of the arborescence.

\( T \) is called the descendant of \( S \) if an elementary path exists between \( S \) and \( T \).
c) **Different States of the Nodes in the Arborescence**

We use the following graphical language:

- `O` accepted node
- `x` refused node

**R₁:**

So is accepted but all its descendants are refused

**A₂:**

Node $S₀$ is accepted.

**O₂:**

$T$ is a descendant of $S$. It is accepted. It belongs to the right wing originated from $S$. No nodes of the opposite wing were investigated.

**A₀₂:**

Same as for $O₂$ but for a left wing

**R₂:**

The left wing is refused.

**A₀₂:**

The right wing is refused.

**O₃:**

$S$ is accepted, the right wing is refused, the left wing not yet explored.

**R₃:**

$S$ is accepted, the left wing is refused, the right wing not yet explored.

**O₃:**

$S$ is accepted. All its descendents are refused. Left wing not yet explored.

**A₀₃:**

Same as $O₃$ but the right wing is not yet explored.
S is accepted, its descendants are refused. The left wing is also refused.

Same as RRA3 but the right wing is refused.

S is accepted. Both wings originated from S are refused. S, which has been previously accepted, will now be refused because all its descendants are refused.

d) RULES RELATED TO THE GRAPH G

We shall use the following rules in order to use the graph G.

Rule 0: If t=0, examine the node $S_0$. Therefore, we solve problem (P) without constraint (C-3). In order to do so, we use the GRG algorithm. We let $\phi = +\infty$.

Rule 1: If the descendants of $S_O$ are refused (this case is possible if $\hat{\phi}_S > \hat{\phi}$), END of the exploration. Then two possibilities exist: we have the solution to problem (P) or the constraints (C-1), (C-2) and (C-3) are incompatibles.

Rule 2: If at the level t, a node $S$ is accepted and if its descendants are not refused, examine a successor at the level t+1. In this case, two possibilities exist:

(i) accept T if \[
\begin{cases}
\hat{\phi}_T = +\infty \\
\phi_T < \hat{\phi}
\end{cases}
\]
T is not a terminal node
(ii) refuse T

In all cases add 2 to the current level

Rule 3

- If at a level \( t \), we have one of the following possibilities:

```
RRA 3
```

T is accepted but all its
descendants are refused, and
the left wing is also refused

```
ARR 3
```

T is accepted but all its
descendants are refused and
the right wing is also re-
fused

refuse node T which does not improve the objective function.
Therefore, refuse all its descendants.

- If at a level \( t \), we have the following case:

```
RR4
```

S is accepted, both wings or-
ginated from s are refused

Then:

(i) refuse s if it does not improve the objective function

(ii) if the variable used at node s is at one of its bounds,
refuse the successor corresponding to the wing of s

- If at level \( t \), we have one of the following cases:
T is accepted, the right wing is refused, left wing not yet explored

T is accepted, the left wing is refused, right wing not yet explored

examine the origin of the opposite wing and accept or refuse it.

Rule 4  When we move up in the arborescence (i.e. when t diminish), if we have ARO3, ORA3 (respectively ARR3, RRA3) and if S corresponds to an upper bound (respectively to a lower bound) of (P), refuse the wing corresponding to S.

Rule 5  If we have the following possibility:

RR4:  \[ \frac{\text{x}}{\text{x}} \]  two wings refused.

Refuse all the successors originated from both wings subtract 1 from current level.

Rule 6  If S is terminal (i.e. \( \hat{x}_j(S) \) is integer \( V_j \epsilon \mathbb{E} \)) then S is refused.

If \( \hat{\phi}_S \) < \( \hat{\phi} \), then \( \hat{x}(S) \) becomes the best solution.

If \( \hat{\phi}_S \) > \( \hat{\phi} \), then the best solution is the current one.
e) **Transformation of the States of the Nodes by the Rules**

Using the rules described in (d), we obtain the following transformation table:

<table>
<thead>
<tr>
<th>State</th>
<th>The level before</th>
<th>Rule Applied</th>
<th>States after</th>
<th>The level after</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>0</td>
<td>1</td>
<td>end of the exploration</td>
<td></td>
</tr>
<tr>
<td>$A_2$</td>
<td>$t$</td>
<td>2</td>
<td>$OA_2, AO_2, OR_3, R_3$</td>
<td>$t+1$</td>
</tr>
<tr>
<td>$OA_2$</td>
<td>$t$</td>
<td>2</td>
<td>$OA_2, AO_2, OR_3, R_3$</td>
<td>$t+1$</td>
</tr>
<tr>
<td>$RA_2$</td>
<td>$t$</td>
<td>2</td>
<td>$OA_2, AO_2, OR_3, R_3$</td>
<td>$t+1$</td>
</tr>
<tr>
<td>$AO_2$</td>
<td>$t$</td>
<td>2</td>
<td>$OA_2, AO_2, OR_3, R_3$</td>
<td>$t+1$</td>
</tr>
<tr>
<td>$AR_2$</td>
<td>$t$</td>
<td>2</td>
<td>$OA_2, AO_2, OR_3, R_3$</td>
<td>$t+1$</td>
</tr>
<tr>
<td>$OR_3$</td>
<td>$t$</td>
<td>3</td>
<td>$AR_2, RR_4$</td>
<td>$t$</td>
</tr>
<tr>
<td>$RO_3$</td>
<td>$t$</td>
<td>3</td>
<td>$RA_2, RR_4$</td>
<td>$t$</td>
</tr>
<tr>
<td>$ORA_3$</td>
<td>$t$</td>
<td>3</td>
<td>$OA_2, OR_3$</td>
<td>$t$</td>
</tr>
<tr>
<td>$ARO_3$</td>
<td>$t$</td>
<td>3</td>
<td>$AO_2, RO_3$</td>
<td>$t$</td>
</tr>
<tr>
<td>$RRA_3$</td>
<td>$t$</td>
<td>3</td>
<td>$RA_2, RR_4$</td>
<td>$t$</td>
</tr>
<tr>
<td>$ARR_3$</td>
<td>$t$</td>
<td>3</td>
<td>$AR_2, RR_4$</td>
<td>$t$</td>
</tr>
<tr>
<td>$RR_4$</td>
<td>$t$</td>
<td>5</td>
<td>$R_1, ORA_3, ARO_3$</td>
<td>$t$</td>
</tr>
</tbody>
</table>
Let $S$ be an accepted node at the level $t$. The solution of problem $P(S)$ gave us $\hat{\mathbf{x}}(S)$ and $\hat{\lambda}_S$. Besides we have $t = \text{card } [E(S)]$.

The node $T$, successor of $S$ may be:

(i) at the same level $t$ as $S$. In this case, it may be a successor in the same wing or at the origin of the opposite wing of $S$.

(ii) at the level $t+1$. In this case, $T$ is at the origin of a wing. The other wing is still unexplored if we move down in the arborescence, or closed if we move up in the arborescence.

In order to solve $P(T)$, we can choose one of the following strategies:

(a) If the integer value to try is $[\hat{x}_\beta]$

We solve the following auxiliary problem:

\[
\begin{align*}
\text{Min } & x_\beta \\
\text{s.t. } & h_i(x) \leq 0 \quad & i = 1, \ldots, m \\
& a_j \leq x_j \leq 1_j \quad & j \in J = \{1, 2, \ldots, n\} \\
& x_j = \hat{x}_j \quad & \forall j \in E(S) \\
& [\hat{x}_\beta] \leq x_\beta
\end{align*}
\]
(b) If the integer value to try is \( \lceil \hat{x}_\beta \rceil + 1 \)

In this case, we solve the following auxiliary problem:

\[
P_s \begin{cases} 
\text{Max } x^s_{\beta} \\
\text{s.t.} \\
h_i(x) \leq 0 & i = 1, \ldots, m \\
0 \leq x_j \leq 1 & j \in J = \{1, 2, \ldots, n\} \\
x_j = x^e_j & \forall j \in E(S) \\
\lceil x^s_{\beta} \rceil \leq [\hat{x}_\beta] + 1 
\end{cases}
\]

Notes

- \( \hat{x}(S) \) is a feasible point to the problems \( P_i \) and \( P_s \).
- We can solve \( P_i \) and \( P_s \) using the same code (GRG) as the one used to solve \( P(S) \).
- Let \( x^i = \{x^i_1, x^i_2, \ldots, x^i_n\} \) and \( x^s = \{x^s_1, x^s_2, \ldots, x^s_n\} \) be the optimal solutions to \( P_i \) and \( P_s \).

(a) if \( x^i_{\beta} > [\hat{x}_\beta] \) (respectively \( x^s_{\beta} < [\hat{x}_\beta] + 1 \)) we can conclude that a solution that has \( x^s_{\beta} = [\hat{x}_\beta] \) (respectively \( [\hat{x}_\beta] + 1 \)) will be infeasible for \( P(T) \). Therefore, we don't have to solve \( P(T) \). We can refuse node \( T \) without solving \( P(T) \).

(b) if \( x^i_{\beta} = [\hat{x}_\beta] \) (respectively \( x^s_{\beta} = [\hat{x}_\beta] + 1 \)) the solution \( x^i \) of \( P_i \) (respectively \( x^s \) of \( P_s \)) is infeasible for \( P(T) \).
8) SCHEDULING OF THE FORTUITOUS INTEGER VARIABLES

By solving $P(S)$, we obtain the optimal solution $(x(S), \hat{\phi}_S)$.

Let $E(S) = \{ j \in E | x_j(S) \text{ is integer} \}$

We have $ACE(S) = E$.

If $E(S)-A \neq \emptyset$, we can say that by solving $P(S)$, we obtained one or more "fortuitous integer variables".

By obtaining $\hat{x}(S)$ and $\hat{\phi}_S$, we have $\hat{x}_j = x_j, \forall j \in E$. If by solving $P(S)$ we have made appear $j_1 \in E-A$ such that:

$$x_{j_1} = a_{j_1}$$

or

$$x_{j_1} = b_{j_1}$$

The variable $x_{j_1}$ is called "fortuitous integer variable". Let $E_f = j_1, j_2, \ldots, j_f = \text{set of the index of } E \text{ corresponding to fortuitous integer variables in the solution } x(S)$. We have:

$$(E(S) \subseteq E)$$

and

$$(E(S) = E_f \subseteq A)$$

If, while investigating the node $S$, the level was $t$, it becomes $t+f$ after the analysis of $x(S)$. The search for the fortuitous integer variables is made on the variables which are not integer at the level $t$. The order of discovering these variables is dependent on the order in which will be ranked the variables which are not yet integer. In order to eliminate many nodes from the arborescence, it will be interesting to have at the highest levels, the fortuitous integer variables which in its descendence there is a probability to
find a good integer solution. Therefore, we will have to change the scheduling of those fortuitous integer variables.

**CRITERION USED TO MODIFY THE SCHEDULING OF THE FORTUITOUS INTEGER VARIABLES (FIV)**

We have indicated that the FIV are variables at their bounds. They are therefore non-basic variables. In such a case, the components corresponding to the reduced gradient are zero. Therefore, we can use the following criterion:

"RANK THE FIV in the decreasing order of their reduced gradient components". If the absolute value of a component of the reduced gradient is small, a small augmentation of the value of the considered point, will have no effect on the objective function. We can expect that an integer solution found in the descendance of this point will lead to a value of the objective function not too far from the continuous optimal solution.
h) Choice of the Separation Variable

Let $S$ be an accepted node at the level $t$. The solution of $P(S)$ gives us $\hat{x}(S)$, $\phi_S$ and $E(S)$. When we want to investigate a successor $T$ at the level $t+1$, we have to choose:

(a) an index $\beta \in E-E(S)$

(b) the wing originated from $S$ and containing $T$.

We call $E-E(S)$ or any subset of $E-E(S)$, the "choice set".

In linear programming, we can compute for the variables included in the choice set, penalties. Those penalties allow us to choose a separation variable and a wing. In nonlinear programming, those penalties are not applicable. Therefore, we propose

Criterion - Put in a Waiting List the Problem Corresponding to the biggest pseudo-cost

\[ x = [x_j] \quad \quad \quad \quad \quad x = [x_j] + 1 \]

\[ j \quad j \quad j \quad j \]

Let $R_K$ be a node of the arborescence, and $R_l$ and $R_m$ its direct successors obtained by adding respectively $x_j \leq [x_j]$ and $x_j \geq [x_j] + 1$. 

\[ x = [x_j] \quad \quad \quad \quad \quad x = [x_j] + 1 \]

\[ j \quad j \quad j \quad j \]
Define:

\[ \phi_K = \text{value of the objective function at node } R_K \]
\[ \phi_m = \text{value of the objective function at node } R_m \]
\[ f_j = \text{the decimal part of } x_j \]

We define the lower pseudo-cost relative to \( x_j \), the quantity

\[ A_j(K) = \frac{\phi_K - \phi_1}{f_j} \]

and the upper pseudo-cost relative to \( x_j \), the quantity

\[ B_j(K) = \frac{\phi_K - \phi_m}{1-f_j} \]

\( A_j(K) \) is the diminution of the objective function corresponding to a decrease of one unit of \( x_j \).

\( B_j(K) \) is the diminution of the objective function corresponding to an increase of one unit of \( x_j \).

The expression \( \max [\max (A_j, B_j)] \) gives us an index \( i \) and one pseudo-cost (either \( A_i \) or \( B_i \)).

(i) If the maximum is reached for \( A_i \), put in the list the problem obtained by adding to \( R_K \) the constraint

\[ x_i < [x_i] \]

(ii) If the maximum is reached for \( B_i \), we put in the list the problem obtained by adding to \( R_K \) the constraint

\[ x_i > [x_i] \]

In both cases, we solve immediately the problem which was not put in the list.
i) **CONVERGENCE OF THE ALGORITHM**

The number of nodes in the graph $G$ is finite. Besides, we never meet twice the same node. Therefore, after a finite number of steps, the algorithm gives us a solution to problem $(P)$. If it is not the case, we can conclude that the constraints $(C-1)$, $(C-2)$ and $(C-3)$ are incompatibles.

j) **NUMBER OF NODES STORED IN THE MAIN CORE OF THE COMPUTER**

Traditional branch and bound methods have a big disadvantage: the number of nodes to be stored is increasing very rapidly (about $2^n$). For large-scale mathematical programming problem, one has to use secondary storage, in order to store all the informations relative to each node. The disadvantage associated with the usage of secondary storage is that the execution time increases very rapidly.

In order not to use secondary storage (and therefore in order to reduce the execution time), one has to limit the number of nodes to be stored in the main memory. A careful look at the different states of the arborescence using our method, indicates that we store at the most an accepted node at each level. Using the fact that we store the root but not the terminal node, the number of nodes stored is equal to $N$, where $N$ is the number of integer variables of the problem $(P)$. We can therefore expect to solve large-scale integer nonlinear programming problem in a reasonable CPU time.
APPENDIX B

ADVANTAGES AND DISADVANTAGES OF CENTRALIZED AND DECENTRALIZED SYSTEMS
A. Advantages of centralized and decentralized systems

(4) Organizational Considerations

<table>
<thead>
<tr>
<th>Centralized Systems</th>
<th>Decentralized Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Easier consolidation of company-wide operating results</td>
<td>- Profit and loss responsibility</td>
</tr>
<tr>
<td></td>
<td>- Familiarity with local problems</td>
</tr>
<tr>
<td>- Ease of control and coordination by corporate management</td>
<td>- Rapid response to local needs</td>
</tr>
<tr>
<td></td>
<td>- Special programs and services can be tailored to division</td>
</tr>
<tr>
<td>- Enhances corporate consolidation</td>
<td>- Easier communication between DP and user</td>
</tr>
<tr>
<td></td>
<td>- &quot;Hands on&quot; experience for users possible</td>
</tr>
<tr>
<td>- Can lead to integration of other administrative functions</td>
<td>- More flexibility in dealing with crises and changes in plans</td>
</tr>
<tr>
<td></td>
<td>- Better service -- under user control</td>
</tr>
<tr>
<td>- Easier to implement and maintain standards</td>
<td>- &quot;Flexibility in aligning EDP with organization's philosophy&quot;</td>
</tr>
<tr>
<td></td>
<td>- Higher share of raw computing power available to user</td>
</tr>
<tr>
<td>- Small user access to large CPU</td>
<td>- Feeling of exclusive use by user organization</td>
</tr>
<tr>
<td></td>
<td>- When use standard equipment: Developed or shared basis</td>
</tr>
</tbody>
</table>
Centralized Systems

(ii) Cost Considerations

- Economies of scale in mainframes
- Economies of scale in mass storage devices
- Reduced record storage duplication

Decentralized Systems

- Transfer of personnel between divisions
- Reduce number of separate equipment studies
- Can move computer between departments
- Objectives oriented
- Less bureaucracy
- Less possibility of hostility between users and corporate data processors
- Reduced over-all system complexity
- Less competition for priority of service
- Familiarity with local problems

(II) Cost Considerations

Centralized Systems

- Economies of scale in mainframes
- Economies of scale in mass storage devices
- Reduced record storage duplication

Decentralized Systems

- Modest start-up costs
- Modest incremental expansion costs
- Low start-up costs
- Low incremental expansion costs
- High cost/performance ratio
Centralized Systems

- Fuller utilization of processing capability
- Shared development and operating costs
- Economies of integrated requirements

Decentralized Systems

- Implementation of software is less costly
- Prolonged mainframe life

(iii) Personnel Considerations

Centralized Systems

- General shortage of competent DP personnel
- More efficient use of personnel talents
- Larger and more expert pool of consultants
- Broader career opportunities more attractive
- Higher standards due to more competitive salary levels
- Personnel turnover less critical
- Cross fertilization

Decentralized Systems

- Greater interest and motivation at local level
- Identification with the mission of the sub-organization
- Less risk of personnel turnover
- More opportunity to communicate with line management
- Less skilled personnel required
- Use of unskilled personnel
(iv) **Technical Considerations**

**Centralized Systems**
- More sophisticated software, better service to programmers and users
  - System software can provide help
  - Greater selection of programming language, debugging aids, etc.
- Can handle large programs — no need to break up program
- Easier implementation of database technology

**Decentralized Systems**
- Easier to add application/services (especially O-L)
- Forces modular programming easier to debug and maintain
- Less specialized support
- Smaller programs -- need handle only one local situation
- Large numbers of application programs and systems tools
- Easy to satisfy "hands on" requirement for testing purposes
- More fault tolerant design
- Easier to add new services
- Less specialized support
- Newer hardware technology on the average
- Higher reliability
- Better data communications
B. Disadvantages of centralized and decentralized systems

(i) Organizational Considerations

<table>
<thead>
<tr>
<th>Centralized Systems</th>
<th>Decentralized Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management problems</td>
<td>People in data processing</td>
</tr>
<tr>
<td>associated with large staffs.</td>
<td>required to serve two masters</td>
</tr>
<tr>
<td>More likely to cause political problems</td>
<td>No professional EDP management</td>
</tr>
<tr>
<td>More rigid. Any change may have serious ramifications</td>
<td>Separate equipment acquisition, studies, and interchangeability.</td>
</tr>
<tr>
<td>Requires more top management involvement</td>
<td>Problems of network management:</td>
</tr>
<tr>
<td></td>
<td>. Income allocation</td>
</tr>
<tr>
<td></td>
<td>. Expense allocation</td>
</tr>
<tr>
<td></td>
<td>. Assigning performance responsibility</td>
</tr>
<tr>
<td></td>
<td>. Agreement on priorities</td>
</tr>
<tr>
<td>More vulnerable to corporate overhead reduction</td>
<td></td>
</tr>
</tbody>
</table>

(ii) Cost Considerations

<table>
<thead>
<tr>
<th>Centralized System</th>
<th>Decentralized System</th>
</tr>
</thead>
<tbody>
<tr>
<td>May require costly controls</td>
<td>Some idle resources</td>
</tr>
<tr>
<td>Danger of expensive overhead</td>
<td>Possible duplication of software costs</td>
</tr>
<tr>
<td></td>
<td>Moderate cost for extensive conversion</td>
</tr>
</tbody>
</table>
(iii) **Other Considerations**

**Centralized Systems**

- System is complex and resource consuming
- Multiprogramming limits programmers
- Chances for Peter Principle effects
Appendix C

Detailed Version of the Computer and Communication Lines Capacity Constraints
1. The Communication Line Capacity Constraint

To determine the communication line capacity constraint between node i and node j, we will consider the following four cases:

a) **Case where the database is at node j and the program at node i**

In this case, the query traffic will be:

\[ \sum_{r,p,d} \alpha Q_{r}^{d}(1 + \gamma_{j}) X_{r}^{d} X_{i}^{j} Y_{r}^{d} \]

and the update traffic will be:

\[ \sum_{r,p,d} \beta U_{r}^{d} X_{r}^{d} X_{i}^{d} \]

b) **Case where the database is at node i and the program at node j**

The query traffic will be:

\[ \sum_{r,p,d} \alpha Q_{r}^{d}(1 + \gamma_{i}) X_{r}^{d} X_{j}^{i} Y_{r}^{d} \]

The update traffic will be:

\[ \sum_{r,p,d} \beta U_{r}^{d} X_{r}^{d} X_{j}^{d} \]

c) **Case where the database and the program are at node i (and the user at node j)**

The overall traffic will be:
\[ \sum_{p,d} [Q_{jp}^d (1 + \gamma_{ji}) + U_{jp}^d] X_{jp}^d \]

d) **Case where the database and the program are at node j (and the user at node i)**

The overall traffic will be:

\[ \sum_{p,d} [Q_{ip}^d (1 + \gamma_{ij}) + U_{ip}^d] X_{ip}^d \]

By summing all the four cases and letting the result be less or equal \( \sum_{c} Q_{c} L_{ij}^d \), we obtain:

\[ \sum_{r,p,d} Q_{rp}^d (1 + \gamma_{rj}) X_{rip}^d X_{jp}^d + \sum_{r,p,d} Q_{rp}^d (1 + \gamma_{ri}) X_{rjp}^d X_{jp}^d \]

\[ \sum_{r,p,d} U_{rj}^d X_{rjp}^d + \sum_{r,p,d} U_{ri}^d X_{rjp}^d + \sum_{p,d} [Q_{ip}^d (1 + \gamma_{ij}) + U_{ip}^d] X_{ip}^d \]

\[ \sum_{p,d} [Q_{jp}^d (1 + \gamma_{ji}) + U_{jp}^d] X_{jp}^d < \sum_{c} Q_{c} L_{ij}^d \]

2. **The Computer Capacity Constraint**

Two types of requests should be considered: the requests to database and the request to program. The overall situation can be represented by the following graph:
Let us suppose that the database is at node $k$.

a) **Process the requests to the database**

The processing requirement will be:

For the queries:

$$\sum_{i,j,p,d} \alpha_{ip}^d X_{ijp}^d X_{jkpd}$$

For the updates:

$$\sum_{i,p,d} \beta_{ip}^d X_{ik}^d$$

b) **Process the requests to the program (even if we have the database at the same node)**

The processing requirement will be:

$$\sum_{i,p,d} (Q_{ip}^d + U_{ip}^d) X_{ikp}^d$$

By taking the sum of all the components and by letting the result be less or equal $\sum_{m}^{k} \gamma_{m}^m$, we obtain:

$$\sum_{i,j,p,d} \alpha_{ip}^d X_{ijp}^d X_{jkpd} + \sum_{i,p,d} \beta_{ip}^d X_{ik}^d + \sum_{i,p,d} (Q_{ip}^d + U_{ip}^d) X_{ikp}^d < \sum_{m}^{k} \gamma_{m}^m$$

$\forall i \in I = K, \forall j \in J = K, \forall p \in P, \forall d \in N$, and for all $k \in K$

Notice that requests to a node having both a program and a database are counted twice.