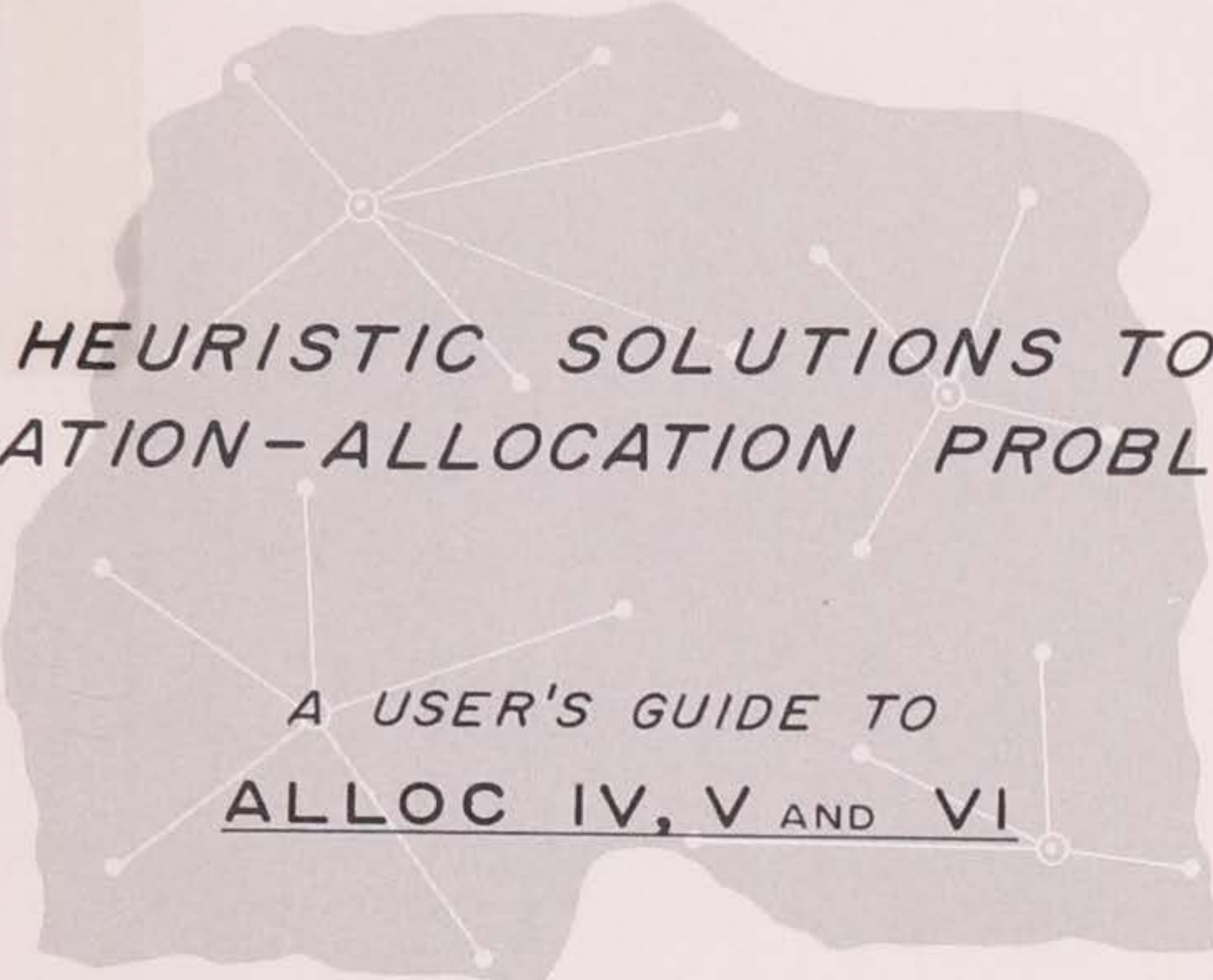


G1  
.M66  
no. 7  
1980



*HEURISTIC SOLUTIONS TO  
LOCATION-ALLOCATION PROBLEMS:*

*A USER'S GUIDE TO  
ALLOC IV, V AND VI*

*BY*

*EDWARD L. HILLSMAN*

MONOGRAPH NUMBER 7  
DEPARTMENT OF GEOGRAPHY  
THE UNIVERSITY OF IOWA  
IOWA CITY, IOWA 52242

JULY, 1980

HEURISTIC SOLUTIONS TO  
LOCATION-ALLOCATION PROBLEMS:

A User's Guide to  
ALLOC IV, V and VI

by

Edward L. Hillsman

Monograph Number 7

Department of Geography

The University of Iowa

Iowa City, Iowa 52242

July, 1980



The University of Iowa Monograph Series

- No. 1 McCarty, John C. Hook and Duane S. Knos, The Measurement of Association in Industrial Geography, 1956.
- No. 2 Thomas, Edwin N., Maps of Residuals from Regression: Their Characteristics and Uses in Geographic Research, 1960.
- No. 3 McCarty, Harold H. and Neil E. Salisbury, Visual Comparison of Isopleth Maps as a Means of Determining Correlations Between Spatially Distributed Phenomena, 1961.
- No. 4 Salisbury, Neil E. and Gerard Rushton, Growth and Decline of Iowa Villages: A Pilot Study, 1963.
- No. 5 Salisbury, Neil E., James C. Knox, and Richard A. Stephenson, The Valleys of Iowa: Valley Width and Stream Discharge Relationships in the Major Streams, 1968.
- No. 6 Rushton, Gerard, Michael F. Goodchild and Lawrence M. Ostresh, Jr., Computer Programs for Location-Allocation Problems, 1973.
- No. 7 Hillsman, Edward L., Heuristic Solutions to Location-Allocation Problems: A User's Guide to ALLOC IV, V and VI, 1980.

Monographs 1-5 are out of print, but zerographic or microfilm copies are available from:

University Microfilms  
Attn.: Books Editorial Department  
300 North Zeeb Road  
Ann Arbor, Michigan 48106

Monograph No. 6 can be obtained from:

CONDUIT  
International Headquarters  
100 Lindquist Center for Measurement  
The University of Iowa  
Iowa City, Iowa 52242

Please write to them for complete information on price and ordering procedures.

Monograph No. 7 can be obtained postpaid from:

Monograph Series  
Department of Geography  
The University of Iowa  
Iowa City, Iowa 52242, U.S.A.

Availability of Programs for Monograph No. 7

Fortran source programs are available on tape at a cost of \$25 including tape. The 6 programs in the monograph are written in order as separate unlabelled files, with each file including the source code. They are recorded at 1600 bpi, in 80-character logical records, 1600 characters to a block on a 9-track tape, in EBCDIC code. Write to:

Departmental Programmer  
Department of Geography  
The University of Iowa  
Iowa City, Iowa 52242, U.S.A.

Copyright 1980, Department of Geography, The University of Iowa.



TABLE OF CONTENTS

Preface	vi
Acknowledgements	xi
Chapter:	
I. Concepts and Terminology for the ALLOC Programs	1
II. Algorithms	
III. Running ALLOC V	
IV. A Data Structure for Large p-median Problems	
V. ALLOC VI	
VI. RETRENCH	
Literature Cited	

## PREFACE

During the six years since Monograph 6 was issued (Computer Programs for Location-Allocation Problems), the faculty and students of the Department of Geography have conducted much research on solving facility location problems. Some of this work was basic research on how to solve problems more efficiently, or on how to solve large problems at all. This work led to a complete recoding of the original ALLOC program (Chapter 9 in Monograph 6), and to the development of new heuristic algorithms. Some of the research also involved applying facility location models to situations faced by public agencies in the State of Iowa. This work led to the development of new strategies for solving very large problems, to another complete recoding of the ALLOC program, to the development of another algorithm, and to the incorporation of new features into the program to make it more useful for analyzing facility location problems.

The computer programs that resulted from this research are substantially more efficient, substantially more flexible, and substantially more powerful than the original ALLOC program, and it is for this reason that we have decided to make them available. The programs described in this monograph have the following advantages over ALLOC: (1) they permit the solution of problems involving larger numbers of nodes--up to 500 nodes for ALLOC IV and ALLOC V, and up to 3,000 nodes for ALLOC VI; (2) they require less computer time; (3) they are more flexible to use, particularly when working with maximum distance constraints; (4) they contain additional algorithms; and (5) the program codes for ALLOC IV and ALLOC V are easier to modify when a special analysis requires changes to the program.

The monograph describes six computer programs. Three of the programs, ALLOC IV, ALLOC V, and ALLOC VI, contain heuristic algorithms for solving p-median problems. ALLOC IV and ALLOC V can solve moderate-sized problems, involving up to several hundred locations. ALLOC VI uses special methods of storing and manipulating data to solve much larger problems, involving up to several thousand locations. As did the original ALLOC program, these three programs combine more than one algorithm. This approach increases the user's access to individual algorithms, and it makes it easier to select or combine different algorithms to meet the needs of specific analyses.



The remaining three programs prepare data for use by the three ALLOC programs. DISTANCE computes distances between locations given by coordinates, for use by ALLOC IV and ALLOC V. UNRAVEL takes distance information computed by DISTANCE (or by the shortest path algorithm, SPA, described in Chapter VIII of Monograph Number 6), and reorganizes it for use by ALLOC VI. If the problem to be solved is very large, further reorganization of the data may be necessary, and the program RETRENCH takes the distances from UNRAVEL and performs the needed operations on them.

A major effort has been made to keep the programs compatible with each other. Thus, the three ALLOC programs use similar data and can be discussed using similar concepts and terminology. Existing data that has been used with the original ALLOC program can be used with ALLOC IV or ALLOC V, and UNRAVEL can convert it to a form that ALLOC VI can use. Collectively, the programs in this monograph form the beginning of a general, integrated system of computer programs for location-allocation analysis.

### Organization

The first portion of the monograph discusses the range of applications of the ALLOC programs and their general features. Thus, Chapter I describes the p-median problem, the data needed for solving it, some of the ways that it can be used to analyze other location problems, and the ways that some of the features of the ALLOC programs may be used to solve it. Chapter I also includes directions for using the simple program DISTANCE to compute distance information for ALLOC IV and ALLOC V.

Chapter II describes the heuristic algorithms available in the ALLOC programs. Two of the algorithms have never been described in the published literature. One of these was designed to solve large p-median problems while using the normal method for storing distance data, as in the original ALLOC program and in ALLOC IV and ALLOC V. The second was designed to solve a multiobjective location planning problem. Both of these algorithms were developed by Gerard Rushton and Edward Hillsman at The University of Iowa, in 1973 and 1975 respectively. The second chapter also discusses the use and limitations of the different algorithms, and it presents computation times and storage requirements for the ALLOC programs.

The second portion of the monograph gives directions for preparing data for the ALLOC programs. Chapter III does this for ALLOC IV and ALLOC V. Chapter IV describes the data structure, or method of organizing distance data, that ALLOC VI uses, and it describes how to use the simple



program UNRAVEL to put data into this form. Chapter IV also discusses the differences between ALLOC VI and the other ALLOC programs. Chapter V contains specific directions for preparing other input data required by ALLOC VI. Finally, Chapter VI gives directions for using the program RETRENCH to prepare data so that ALLOC VI can solve very large problems.

### Conventions

In the interest of brevity, the monograph uses several conventions. First, with the exception of the discussion of execution times in Chapter II, everything written about ALLOC V in the monograph applies equally to ALLOC IV. The two programs require identical input data, use virtually identical amounts of core storage, and give identical answers. ALLOC V requires substantially less computer time than ALLOC IV and is, therefore, generally to be preferred. Greater speed required greater complexity in the program code, however, and this makes ALLOC V difficult to modify for special analyses (e.g., Meneley, 1973). For this reason, both programs are presented here, but the discussion will refer only to ALLOC V.

Second, the discussion will refer to punched card input as a matter of convenience. The ALLOC programs are able to read much of their input data from other storage media such as tapes and disks, however, and the sections on preparing the input data indicate how to supply data from these media.

Third, the main body of the monograph refers only to the p-median problem, with and without maximum distance constraints, as the type of location problem that the ALLOC programs can solve. This convention greatly understates the power of the programs as tools of locational analysis. Recent work by Church (1974), Church and ReVelle (1976), and Hillsman (1979) has shown that many location-allocation problems can be solved as though they were p-median problems. Thus, the ability to solve a p-median problem implies the ability to solve these other problems as well.

Although the use of the p-median problem as a tool for solving other problems is an important development, the monograph considers it only within an appendix. It has been written this way for two reasons. First, with few exceptions--notably the p-median problem with maximum distance constraints--the recognition that the ALLOC programs could solve other location-allocation problems came after most of the program codes had been written: ALLOC IV and ALLOC V were substantially complete by the summer of 1974,



and ALLOC VI was substantially complete a year later. Although the programs can solve other location-allocation problems as p-median problems, they are less flexible in this regard than they are for the p-median. Second, most of the research on solving other problems as p-median problems remains in unpublished dissertations. It will therefore be unfamiliar to most users, and particularly to those outside of academia.

To deal with the awkwardness or the unfamiliarity in the main body of the monograph would make it unwieldy. Accordingly, directions for using the programs to solve problems other than the p-median, and directions for interpreting the output of the program in these cases, have been placed in an appendix. The appendix provides only a brief introduction to the topic, however, and analysts interested in greater detail should consult the original research sources cited above. It is anticipated that the results of that research will be published, that the programs will be modified to be more flexible, and that a revised monograph will be issued within a few years.

#### Program Listings

Persons familiar with Monograph Number 6 will notice that this monograph does not contain listings of the computer programs. The monograph has been written as an introduction and guide for the general user of the programs, and most users have little or no need for program listings in their day-to-day work. Moreover, the listings run in excess of 100 pages and including them would require a noticeable increase in the cost of the monograph. It has therefore seemed desirable to omit the listings, with their associated costs, and to let persons who want listings request them at cost by writing to the address inside the front cover. Persons who have purchased the tape that contains the programs will be able to make their own listings as they desire.

#### Disclaimer

Although the programs described in this monograph have been tested and used by many people at The University of Iowa and elsewhere, no warranty, expressed or implied, is made by The University of Iowa or the author as to the accuracy and functioning of the programs and related programs and the related program descriptions; no responsibility is assumed in connection therewith.



## ACKNOWLEDGEMENTS

The computer programs discussed in this monograph were written in large part while I was a research assistant for Dr. Gerard Rushton at The University of Iowa. My first work with Dr. Rushton involved his interest in solving large location-allocation problems, and it was he who first suggested some of the methods for solving large problems that have been used in the programs. Dr. Rushton has requested for several years now that I prepare this monograph, and his persistent interest has finally borne fruit.

A number of persons besides Dr. Rushton and I have used the programs, for a wide range of studies. This use has improved the quality of the programs, by leading both to suggestions for improving them and to the discovery and correction of errors. Among the persons who have contributed in this way have been Jim Kohler, Joe Meneley, Sally McLafferty, Avijit Ghosh, Rick Rhoda, Girish Misra, Rex Honey, Boye Agunbiade, Tom Eagle, Ken Rosing, Mike Goodchild, Nigel Waters, Chuck Hindes, and numerous students in the classes that some of these people have taught. Their assistance is greatly appreciated. I would also like to thank Debra Schottman for typing the final manuscript, and Tom Eagle for preparing the sample input and output.

Finally, this monograph has been drawn from my Ph.D. dissertation. Although the research, software, and much of the writing of this material was done while I was in residence at The University of Iowa, the Energy Division of Oak Ridge National Laboratory supported the remainder of the writing and the preparation of most of the illustrations that appear here. The Laboratory is operated by Union Carbide Corporation for the United States Department of Energy.



and ALLOC VI was substantially complete a year later. Although the programs can solve other location-allocation problems as p-median problems, they are less flexible in this regard than they are for the p-median. Second, most of the research on solving other problems as p-median problems remains in unpublished dissertations. It will therefore be unfamiliar to most users, and particularly to those outside of academia.

To deal with the awkwardness or the unfamiliarity in the main body of the monograph would make it unwieldy. Accordingly, directions for using the programs to solve problems other than the p-median, and directions for interpreting the output of the program in these cases, have been placed in an appendix. The appendix provides only a brief introduction to the topic, however, and analysts interested in greater detail should consult the original research sources cited above. It is anticipated that the results of that research will be published, that the programs will be modified to be more flexible, and that a revised monograph will be issued within a few years.

#### Program Listings

Persons familiar with Monograph Number 6 will notice that this monograph does not contain listings of the computer programs. The monograph has been written as an introduction and guide for the general user of the programs, and most users have little or no need for program listings in their day-to-day work. Moreover, the listings run in excess of 100 pages and including them would require a noticeable increase in the cost of the monograph. It has therefore seemed desirable to omit the listings, with their associated costs, and to let persons who want listings request them at cost by writing to the address inside the front cover. Persons who have purchased the tape that contains the programs will be able to make their own listings as they desire.

#### Disclaimer

Although the programs described in this monograph have been tested and used by many people at The University of Iowa and elsewhere, no warranty, expressed or implied, is made by The University of Iowa or the author as to the accuracy and functioning of the programs and related programs and the related program descriptions; no responsibility is assumed in connection therewith.



## ACKNOWLEDGEMENTS

The computer programs discussed in this monograph were written in large part while I was a research assistant for Dr. Gerard Rushton at The University of Iowa. My first work with Dr. Rushton involved his interest in solving large location-allocation problems, and it was he who first suggested some of the methods for solving large problems that have been used in the programs. Dr. Rushton has requested for several years now that I prepare this monograph, and his persistent interest has finally borne fruit.

A number of persons besides Dr. Rushton and I have used the programs, for a wide range of studies. This use has improved the quality of the programs, by leading both to suggestions for improving them and to the discovery and correction of errors. Among the persons who have contributed in this way have been Jim Kohler, Joe Meneley, Sally McLafferty, Avijit Ghosh, Rick Rhoda, Girish Misra, Rex Honey, Boye Agunbiade, Tom Eagle, Ken Rosing, Mike Goodchild, Nigel Waters, Chuck Hindes, and numerous students in the classes that some of these people have taught. Their assistance is greatly appreciated. I would also like to thank Debra Schottman for typing the final manuscript, and Tom Eagle for preparing the sample input and output.

Finally, this monograph has been drawn from my Ph.D. dissertation. Although the research, software, and much of the writing of this material was done while I was in residence at The University of Iowa, the Energy Division of Oak Ridge National Laboratory supported the remainder of the writing and the preparation of most of the illustrations that appear here. The Laboratory is operated by Union Carbide Corporation for the United States Department of Energy.



## CHAPTER I

### CONCEPTS AND TERMINOLOGY FOR THE ALLOC PROGRAMS

This chapter describes the purpose of the ALLOC programs, the types of input data they require, and the use of some of the program features.

#### Purpose of Programs

The p-median or central facilities location problem (ReVelle and Swain, 1970) is to find locations from which to serve members of a population, so that the average distance (or, equivalently, the total or aggregate distance) traveled by the population to its nearest center is as small as possible. The population to be served is assumed to occur at the vertices or nodes of a network. Distances are measured along the network, and the service centers are to be located somewhere on the network.

Hakimi (1964) proved that an optimum solution to the p-median problem can be found by locating all service centers at the nodes of the network. Most algorithms for solving the p-median problem rely upon this proof, and examine only the nodes when seeking locations for the centers. Equally good solutions may be found, occasionally, by locating centers on the links of the network. These equally good solutions are rare and most algorithms, including those in the ALLOC programs, make no effort to find them.

The ALLOC programs use heuristic algorithms to solve the p-median problem. These algorithms may obtain very good solutions to the problem, but their solutions are not necessarily the best possible, or optimum, solutions to the problem. The algorithms of Teitz and Bart (1968) and of Hillsman and Rushton frequently find the optimum solution for small problems, however. Chapter II will compare the performance of these algorithms with that of the Maranzana algorithm, and suggest some things to consider when choosing an algorithm to solve a specific problem.

The ALLOC programs also accept certain variations in the standard form of the p-median problem, to widen the applicability of the problem as a tool for locational analysis. The programs will accept location constraints to force centers to locate at or avoid specific nodes on the network. Maximum distance constraints may be imposed on a problem, to ensure that every node is within a specified distance of its nearest center.



ALLOC V and ALLOC VI each have some features not found in the other program. ALLOC V can be made to construct spatial hierarchies automatically, with centers at higher levels of the hierarchy chosen from among the centers in the lower levels. The program will accept a partial, rectangular distance matrix to reduce computer storage for moderate-sized problems (150-500 nodes). ALLOC V contains the algorithm of Maranzana (1964). ALLOC VI lacks the Maranzana algorithm, but it contains an algorithm to add centers to a p-median solution. This algorithm is equivalent to the greedy adding heuristic of Church and ReVelle (1976) or the delta method of Khumawala (1973). ALLOC VI also contains an unpublished algorithm to trade off the average distance measured in the p-median model against an index of characteristics of the nodes that have centers. This permits the selection of locations that are both accessible and suitable for centers in other ways. ALLOC VI can fix centers at designated locations during an entire analysis, and it can also provide punched or other machine-readable output. ALLOC VI uses a special method of storing data that usually reduces both computer storage and execution time from those required by ALLOC V. This has permitted ALLOC VI to be used in analyses containing as many as 2990 nodes.

#### Types of Data Needed

The following paragraphs describe the data base required for solving the p-median problem, and some of the features of the ALLOC programs that may be used to analyze it. The discussion will emphasize features common to all of the ALLOC programs, although occasional mention will be made of features specific to one program or another. Specific program features, and the directions for implementing the common features, will be discussed in greater detail in later chapters.

#### The p-Median Data Base

The ALLOC programs require a data base containing a list of nodes and a matrix of coefficients. For p-median problems, the programs will compute the coefficients using a matrix of distances measured between the nodes, and the populations of the nodes. For other types of problems, discussed in Appendix A, the programs read the complete set of coefficients and analyze it without further computation.

The number of nodes in the data base may be denoted by  $N$  for convenience. The data base for an  $N$ -node,  $p$ -median problem may be represented in the form shown in Figure I-1a.



## List of Nodes

The list of nodes consists of a list of N identification (ID) numbers, representing nodes to be served by service centers. Each node must have a unique ID number, and each ID number must be greater than zero and less than 100,000 (10,000,000 for ALLOC VI). Otherwise, the nodes may be numbered arbitrarily in any fashion. Node ID numbers are used to link information to the rest of the data base, and to interpret the solutions to p-median problems. For ease of interpreting solutions, it is recommended but not required that node ID numbers be placed in the list in ascending order and, where possible, grouped by regions. For example, all nodes in one county or state might be numbered from 1-99, those in another from 100-199, and so forth.

## Distance Matrix

Distances between nodes are recorded in a square matrix of N rows and N columns.<sup>1</sup> Each row of the matrix corresponds to one of the N nodes to be served, and each column of the matrix corresponds to a potential center location, or candidate node. The columns of the matrix must appear in the same order as the rows.

Distances between the nodes may be measured in a variety of ways. The p-median problem was stated originally in terms of a network, and distances obviously can be measured along the shortest paths between nodes of the network. Ostresh (1973) has described a shortest path algorithm, SPA, that can be used for this purpose. Road distances are often approximated adequately by straight-line (Euclidean) or by city-block (Manhattan) measurements, however. The program DISTANCE, described at the end of this chapter, can compute distances in either of these ways. Transportation costs or travel times may also be used instead of direct distance measures. It will be convenient simply to refer to distances.

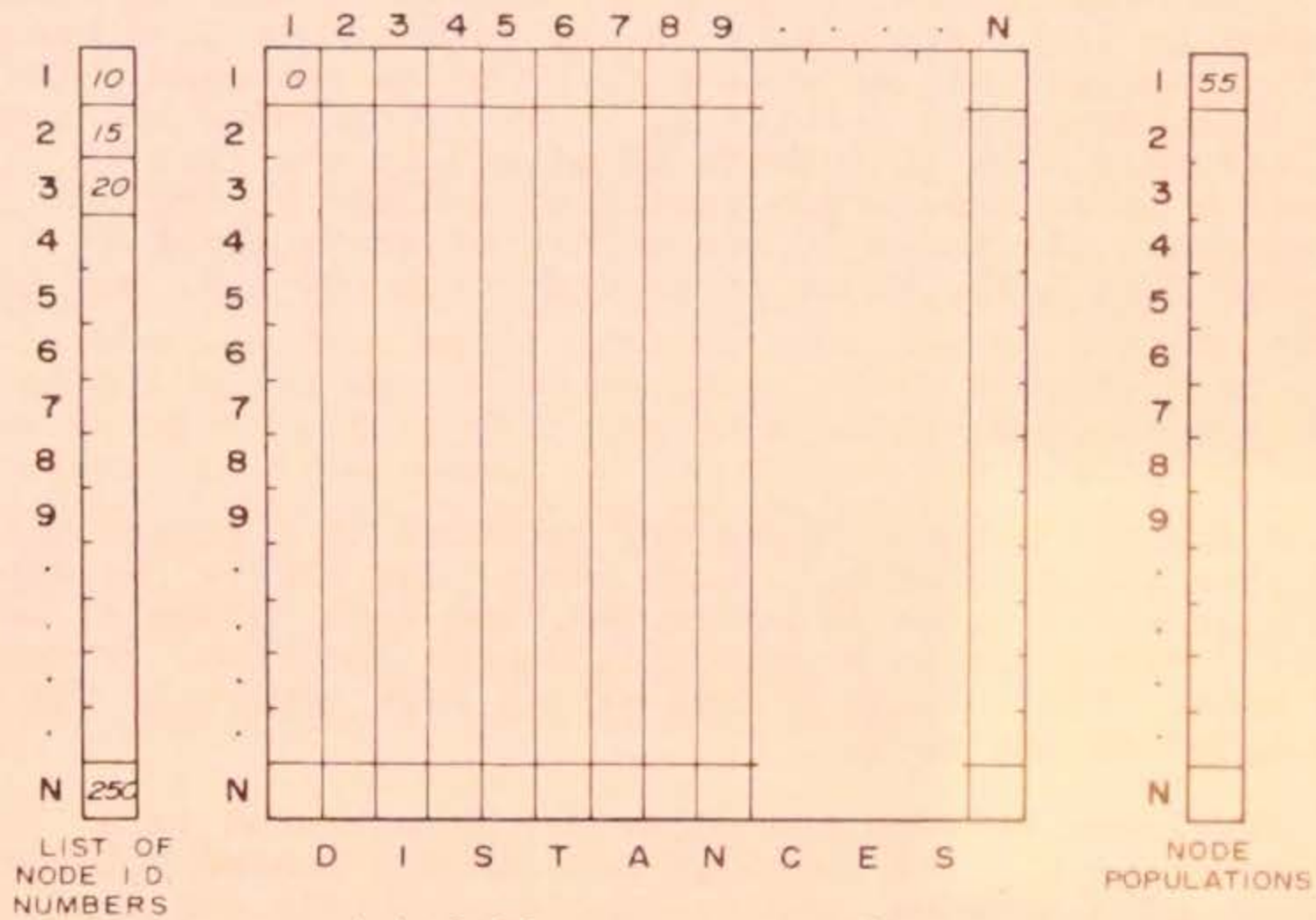
The ALLOC programs place three restrictions on the distance matrix. First, all distances used in the programs must be punched as integers. Decimal fractions may be used by punching them without the decimal point, provided that the resulting integers are in constant units. Second, the programs can use very large distances in their calculations, subject to a limit described in the section on node populations below. Several parts of the programs cannot print distances greater than 99,999, however. If an

---

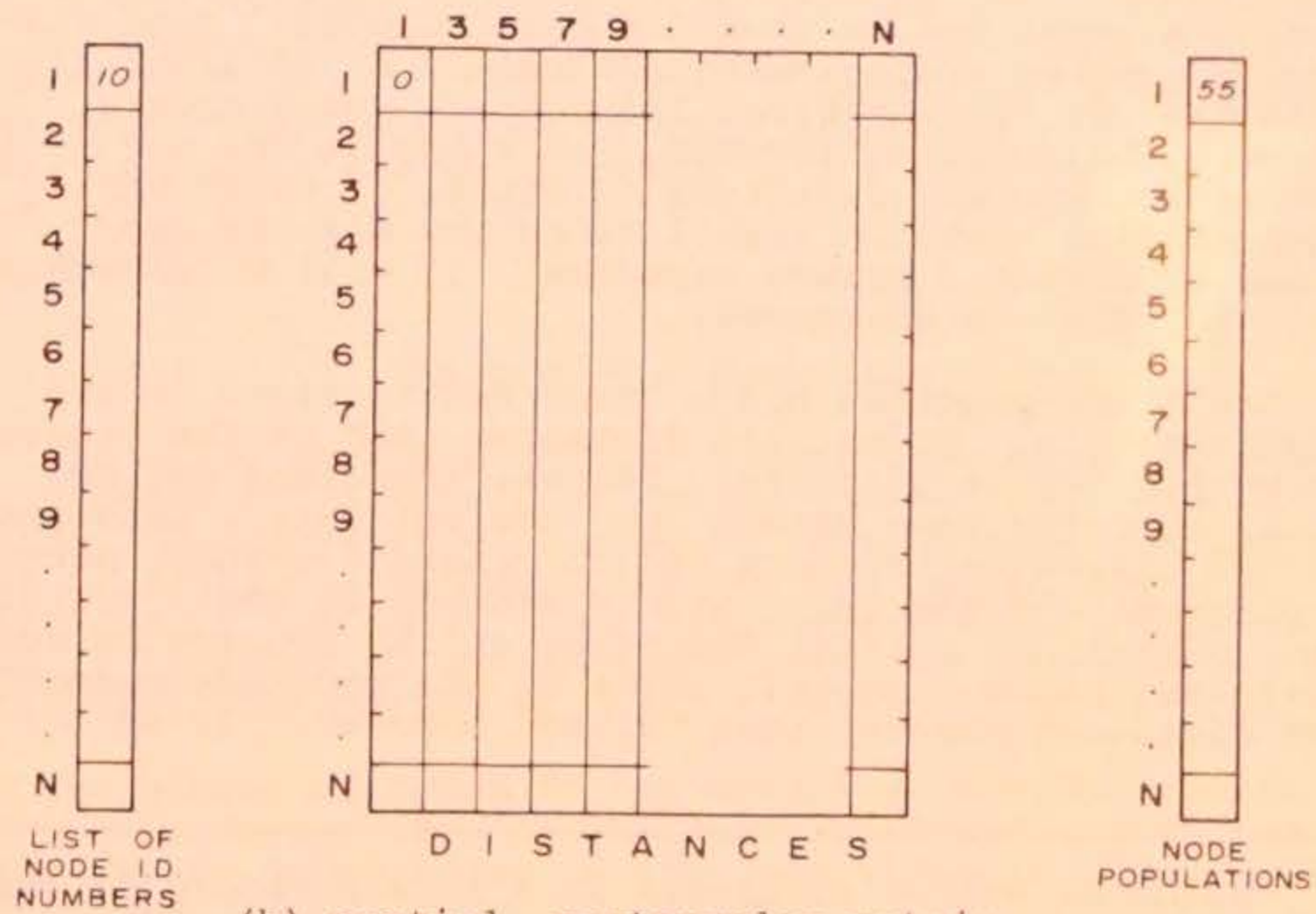
<sup>1</sup>ALLOC VI stores distances in a form markedly different from a matrix, but the form is easily derived from a matrix. For now it will be convenient to assume that ALLOC VI uses a matrix.



FIGURE I-1 p-Median Data Base



(a) full, square matrix.



(b) partial, rectangular matrix.



analysis requires the use of larger distances but does not require that they be printed, then the ALLOC programs may still be of use. Third, the minimum element in each row of the distance matrix (or of a complete coefficient matrix) must fall on the principal diagonal of the matrix. If it does not, the program may compute aggregate distance incorrectly and give invalid results.<sup>2</sup> The responsibility for meeting this requirement rests entirely with the user of the programs. This requirement should not be a problem unless the programs are used to solve location-allocation problems other than the p-median, as discussed in Appendix A.

Distances need not be symmetrical between nodes. One-way streets and intermediate stopping or shipping points often introduce directional differences in the distances between two nodes. If the distances are asymmetrical about the diagonal of the matrix, then the placement of the distances in the matrix rows and columns must reflect the direction of movement when a center provides service to the population. When the important distance is the one that the population must travel to reach the center, as it would be if the population were to walk to the center, then each row of the matrix should contain the distances from a particular node to each of the candidates (Figure I-2b). If the distance to be traveled by the service center staff to the population is more important, as it would be for ambulance or fire stations, then each row of the matrix should contain the distances to be traveled to a particular node from each of the candidate nodes (Figure I-2c). If total round-trip distances from node to center and back are more important, then the distance matrix should be symmetrical, and it should contain the round-trip distance between each pair of nodes (Figure I-2d).

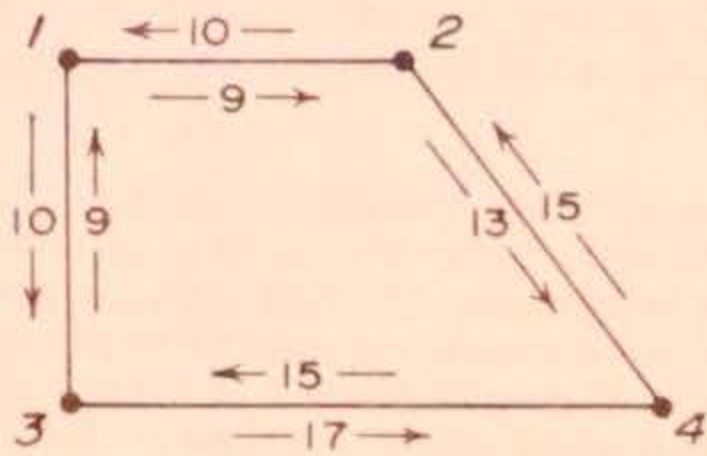
When working with a moderate or large number of nodes, it may be desirable to use less than a full, square distance matrix to reduce core storage requirements. In the standard form of the p-median problem (Figure I-1a), each node to be served is also a potential center location. In many large problems, however, some of the nodes to be served would make poor candidates for centers. For example, in rural areas it may be unrealistic to locate some types of service centers in towns with fewer than 500 inhabitants, even though several centers in such locations might greatly reduce average distance for the region as a whole. In urban areas, zoning restrictions or the need for access to arterial streets may restrict candidates to a small number of the nodes that require service. The use of a full, square distance matrix in these cases would require the

---

<sup>2</sup>If the minimum value in a row falls on the diagonal but is not unique, the programs will compute aggregate distance correctly, but some centers may not serve the nodes where they are located. These errors do not affect the analysis and they may be corrected manually.



FIGURE I-2 Use of Asymmetrical Distances



a

0	9	10	22
10	0	20	13
9	18	0	17
24	15	15	0

b

0	10	9	24
9	0	18	15
10	20	0	15
22	13	17	0

c

0	19	19	46
19	0	38	28
19	38	0	32
46	28	32	0

d



computer to store a large amount of information that would never be used, since the programs need only the distances between each node and each candidate. When a node is deemed to be unsuitable as a candidate, its column of the matrix may be deleted, and the remaining matrix may be compressed into a rectangular form (Figure I-1b). The ALLOC programs can work with a rectangular matrix of this form as long as each column of the matrix corresponds to one of the matrix rows.

### Populations

The populations or weights of the network nodes are recorded in a list or vector corresponding to the list of node ID numbers and the rows of the distance matrix (Figure I-1). The general form of the p-median problem permits each node to have a different population, and the input data for the ALLOC programs will normally contain a deck of node ID numbers and populations. The program uses the node ID numbers to match each population to the proper node, and the cards in the population deck thus may be in any order. The programs will arrange them into the form shown in Figure I-1.

The ALLOC programs have an option to give each node the same population without preparing the full population deck. This feature is useful for solving some standard test problems that assume equal populations (Jarvinen et al., 1972) or that give only the weighted distance matrix (ReVelle and Swain, 1970). This feature is also useful for solving other location-allocation problems as p-median problems, as discussed in Appendix A. In computing coefficients, the ALLOC programs weight the distance matrix by multiplying the distances in each row of the matrix by the population of the node to which the row corresponds.

Although the discussion here has used the term "population," the nodes may be given other attributes for the p-median problem. For example, Schilling et al. (1976) have used property values, and Kohler and Rushton (1977) have used planned expenditures for highway construction.

The weighting operation places an upper limit on the size of population that the ALLOC programs can accept. If the product of  $N$  (the number of nodes<sup>3</sup>) and the largest weighted distance from the weighting operation exceeds 2,147,483,647,<sup>4</sup> the algorithms may fail to find an answer and may fail to stop. In the event that the data would

---

<sup>3</sup>The limit in ALLOC VI involves a value that is usually smaller than  $N$ . This will be explained in more detail in the section on ALLOC VI.

<sup>4</sup>This number is machine dependent and may be larger or smaller on machines than the IBM 360/65 on which the programs were developed.



exceed this limit, the limit may be met by dividing each population by some constant value, such as 10 or 100, before using it in the program. This scaling of the populations may introduce a small amount of rounding error, but it will not otherwise affect the answers obtained with the programs.

Hakimi's theorem (1964) assumes that every intersection in the network of a p-median is a candidate to receive a center, even if some of these intersections or nodes have no population. Many analyses ignore these assumptions, legitimately, by stating that an acceptable solution must have all centers at nodes with populations. On occasion, however, it may be desirable to use nodes without populations in the data base of an analysis. For example, highway intersections near urbanizing areas may be considered in anticipation of future growth. Alternatively, as Church and Meadows (1977) have shown, adding selected nodes to a problem with maximum distance constraints may make it possible to find better solutions to the problem. The ALLOC programs will accept such "dummy" candidate nodes as long as they also appear as rows in the distance matrix and as long as they appear in the population deck for the data base. Analyses involving "dummy" nodes must be interpreted with greater care than other analyses, however, because zeroes in the population deck affect the ability of some algorithms to solve the problem. The chapter on algorithms and algorithm choice will consider the effect of "dummy" nodes in greater detail.

#### Defining and Solving a p-Median Problem

After an ALLOC program has read a p-median data base, it will accept definitions for as many as 100 p-median problems on a single run of the program. Some information, such as the number of centers in the problem, must be provided as a part of every problem definition. Other information, such as the form of printed output or the use of location or maximum distance constraints, is optional.

#### Required Information

A problem definition must include the number of centers in the problem. In addition, the problem definition must contain a list of centers that the heuristic algorithms can move around or add to. Centers are listed by the ID numbers of the candidate nodes where they are located, and the list may be in any order. This list will be referred to as a starting solution.



A problem definition will also normally contain the choice of an algorithm to solve the problem. If no algorithm is specified in the definition, the program will assign each node to its nearest center, compute and print information about the starting solution, and proceed to the next problem without trying to improve the solution.

### Location Constraints

In defining a problem it may be necessary to take account of existing centers, force them into a solution, and locate additional centers around them. In other cases it may be desirable to prevent centers from locating at certain candidate nodes. The ALLOC programs permit a problem definition to contain location constraints on candidate nodes to enforce either or both of these requirements for a problem. The definition of a problem may constrain as many candidate nodes into or out of solution as necessary. Each problem can use a new set of location constraints, or the programs can be made to repeat a set of constraints automatically through consecutive problem definitions. This last feature is useful when locating different numbers of centers around a set of existing centers.

Location constraints may also be used to force the construction of spatial hierarchies, either starting with the lowest level and working up or starting at the highest level and working down. For example, after locating the centers in the lowest level, the analyst may impose location constraints to prevent the algorithm from placing centers anywhere except at candidate nodes with lower level centers. A second level of centers, fewer in number, could then be located. The second level could be used to define the eligible candidates for a third level, and so forth. To work from the top down, constraints may be used to force the centers located at higher levels of the hierarchy to remain in the solutions for lower levels. The top-down approach usually gives a different pattern of centers in the hierarchy than does the bottom-up approach. ALLOC V permits the automated development of hierarchies from the bottom up, but both programs can be used to develop hierarchies, from either direction, with multiple runs of the programs.

Setting a location constraint, in and of itself, does not ensure that the final solution will meet it. To force a center to locate at a particular candidate node, a constraint must be set for that candidate and the starting solution for the problem must also contain a center at the candidate. To prevent a center from locating at a candidate, the constraint must be set for the candidate and the candidate must not appear in the starting solution.



## Maximum Distance Constraints

Maximum distance constraints may be used for three different purposes in an analysis. The most direct of these is to ensure that every node is within some specified distance of its nearest center. Such a constraint is common in planning locations for fire stations or ambulances, and it might be desirable when the population is going to walk to the service centers, as in the case of public schools or bus stops. Because increasing distance is frequently considered a disadvantage, a maximum distance constraint limits the level of disadvantage incurred by any member of the population.

A second use of maximum distance constraints is to solve the minimax location problem, or to minimize the longest distance that any member of the population must travel to reach one of the centers. The minimax problem may be solved by solving a p-median problem first, imposing a maximum distance constraint on the solution, solving the problem again to meet the maximum distance constraint, imposing a tighter constraint or shorter maximum distance, and repeating the sequence until a solution fails to meet the most recent constraint. Although this can be a time-consuming procedure, it is much more effective than the minimax algorithm used in the earlier ALLOC program (Rushton and Kohler, 1973). A problem definition in the ALLOC programs may specify a series of successively tighter maximum distance constraints to carry out this procedure automatically.

A third use of maximum distance constraints is to try to improve the solution obtained by one of the algorithms. When a heuristic algorithm finds a good, nearly optimum solution to a p-median problem, it is occasionally possible to improve the solution by finding the longest distance traveled from any node to its nearest center, imposing a maximum distance constraint just less than this distance, and then solving the problem again under the new constraint. A problem definition may request the program to compute and impose such a constraint after finding a solution, to try to "bump" a good solution into the optimum one. This procedure may fail to improve a solution that is not the optimum. Whether it improves the solution or not, the procedure is still a heuristic one, and nothing should be concluded about the optimality of the solution.

The ALLOC programs discussed here are able to define and solve several problems with maximum distance constraints during a single run of the program. This is an improvement over the original ALLOC program (Rushton and Kohler, 1973), which could only work with one such problem



during a run. When a maximum distance constraint is imposed on a problem, all of the ALLOC programs search the weighted distance matrix, identify distances longer than the maximum distance, and replace these distances with a very large number. This forces the algorithms to meet the maximum distance constraint (Rushton and Kohler, 1973), but it also removes the original values from the weighted distance matrix. The original ALLOC program had no way of recovering these values at the end of the problem. The ALLOC programs in this monograph can recover the original values, but only at the end of a problem, not in the midst of one. For this reason, when a problem definition imposes several maximum distance constraints the constraints must be imposed in decreasing length, e.g., 70km, 68km, 67km, etc. This permits smaller values to be removed from the weighted matrix after the larger constraints have been imposed. Imposing the smaller values first would require the programs to recover lost information before imposing the larger ones, and the programs lack this ability.

When a problem definition contains several maximum distance constraints, the ALLOC programs check the final solution for each constraint to determine whether it meets the next distance to be imposed. Thus, if no node is more than 68km from its nearest center when a constraint of 70km is imposed, the solution would also meet constraints of 69km or 68km. This feature can save large amounts of computer time. The earlier ALLOC program of Rushton and Kohler (1973) did not check solutions in this fashion before imposing new constraints.

The use of maximum distance constraints in a problem definition has two ramifications for the problem being analyzed. First, Hakimi's theorem (1964) does not hold when a problem must meet a maximum distance constraint, and an optimum solution will almost always require centers to be located on the links of the network rather than just at the nodes. Many analyses may avoid this difficulty by defining an acceptable solution as one with all centers at network nodes. Alternatively, Church and Meadows (1977) indicate that they have developed a method of generating additional nodes on the network links so that an optimum solution to the original problem may be found by locating centers at the new and original nodes. Either approach permits the use of conventional p-median algorithms to solve a problem with maximum distance constraints.

The second effect of maximum distance constraints is to impair the ability of the Maranzana and Hillsman-Rushton algorithms to solve p-median problems. The discussion on choosing an algorithm in Chapter II will discuss this in greater detail.



## Output Form

The ALLOC programs compute and print information about the starting solution and the solution at the end of each algorithm. In addition, ALLOC V computes and prints this information at the end of each cycle of an algorithm. This information consists of three parts, one concerning the centers in solution, one concerning the nodes served, and one containing summary information for the problem as a whole.

Information about each center consists of the ID number of the center's location; the population it serves; the aggregate and average distance from the population center; and the increase in aggregate distance that would result if that center were removed from the solution and not replaced by another. Information about the list of nodes consists of each node's population, nearest center, and distance to that center. Summary information contains the aggregate and average distances traveled for the entire solution, the longest distance between any node and its nearest center, and the percentage change in aggregate distance since the last time the summary information was printed.

The information printed about the list of nodes is optional, and it is available in two forms. In one form the information is printed for nodes in the order of ID numbers in the list of nodes. This form is convenient when the list of ID numbers is in ascending order or is organized in some other way. In the second form, information is printed for the nodes in each center's service area. Thus, the information for all of the nodes served by one center will be printed first, followed by that for nodes served by a second center, and so forth. This form is convenient for focusing on the service area of particular centers, but it is inconvenient for finding which center serves a particular node. A problem definition may request node information printed in either of the two forms, or neither.

The information printed by the programs must be interpreted with care if the data base contains "dummy" nodes or if the solution fails to meet a maximum distance constraint. Solutions in these cases are valid. That is, given the definition of the problem, they are the best that the algorithm could find. Unfortunately, the way that they are derived does not permit certain information to be computed. Both programs assign "dummy" nodes to arbitrary centers, although ALLOC VI usually matches them with centers nearby. When a "dummy" node does not have a center in the solution, the node and all information printed about it should be ignored. When a "dummy" node has a center it may fail to serve itself, but all information printed about its service to real nodes will be valid. ALLOC V assumes that all "dummy" nodes meet all maximum distance constraints.



ALLOC VI treats "dummy" nodes no differently than real ones in this regard.

If a real node cannot be served within a maximum distance, ALLOC V will assign it arbitrarily to one of the centers in solution at a very large cost. ALLOC VI will assign it to a dummy center with an ID number of zero. These procedures make it easy to inspect the printed output to determine which nodes cannot be served within the maximum distance, but they make meaningless the aggregate and average distance measures printed for the solution as a whole. When a node cannot be served within the maximum distance, the distance to its center should be ignored. In ALLOC V, if a center serves one of these "unservable" nodes, the statistics for that center should be ignored. In ALLOC VI these statistics are valid but they do not include the effect of nearby nodes that cannot be served within the distance constraint.

#### Running DISTANCE

This section describes DISTANCE, a program to compute distances between nodes using their coordinates, and it gives directions for running the program.

DISTANCE computes a symmetrical matrix of distances between nodes, using either straight-line (Euclidean or Pythagorean) or city-block (Manhattan or right-angle) distance formulas. The program writes the matrix in machine-readable form, for use as input directly to ALLOC V or directly to UNRAVEL, for later use by ALLOC VI. To increase compatibility between DISTANCE and other programs, DISTANCE also creates the variable format card that ALLOC V and UNRAVEL use to read the matrix (Figure I-3) and it creates the matrix using the same format as the shortest path algorithm SPA (Ostresh, 1973).

All data for DISTANCE are read from punched cards. Input consists of a control card, a variable format card, and a deck of cards containing node ID numbers and x-y coordinates (Figure I-3). With the exception of the variable format card, all input data are to be punched as integer variables and are to be right-justified in their allotted fields. The variable format should be punched left justified, for convenience.

1.0 CONTROL CARD (required).

1.1 In columns 1-5, punch the number of nodes.



- 1.2 Column 10 controls the method of computing distances. To use the city-block or Manhattan method, punch a 1 in column 10, to use the straight-line method, punch a 2 in column 10.
  - 1.3 If you want the distance matrix punched on cards, skip to 1.4 below. Otherwise, in columns 11-15 punch the unit number for the tape, disk, or other medium on which the distance matrix is to be written.
  14. If you want the program to print a copy of your distance matrix as well as making a machine-readable copy, punch a 1 in column 20. Otherwise, the program will produce only the machine-readable version.
- 2.0 FORMAT TO READ COORDINATES (required).
  - 2.1 On the next card, punch a Fortran format to read the ID number and x- and y-coordinates for one node. This format must specify three integer fields, and each node must appear on a separate card in the deck.

For example, the ID number of each node might be punched in columns 1-5 of a card, with the x- and y-coordinates punched in columns 16-20 and 21-25. The format for this deck would be (I5,10X,2I5).

- 3.0 NODE ID NUMBERS AND COORDINATES (required).
- 3.1 Each of the remaining cards contains a node ID number, an x-coordinate, and a y-coordinate, punched in that order in fields to match the format in 2.0 above. The nodes may be placed in any order within the deck. The order in which the nodes are placed in the deck determines the order of rows and columns in the distance matrix. Thus, if node ID numbers appear in the deck in the sequence 1,2,3,5,4, the rows and columns will be computed and written in this same order.

#### Punched Output from DISTANCE

Punched output from DISTANCE consists of a format card bearing the format (14I5), followed by the distance matrix punched in this format. Each row of the matrix begins on a new card, with the ID number punched in columns 1-5. The first distance in columns 6-10, and so forth. If more than 13 nodes are used, the distance to the 14th node appears in columns 1-5 of the second card, the 15th in columns 6-10, and so forth. A format of (14I5) has been used to allow for sequential numbering of the cards in columns 71-80.



FIGURE I-3b Sample Input Deck for DISTANCE

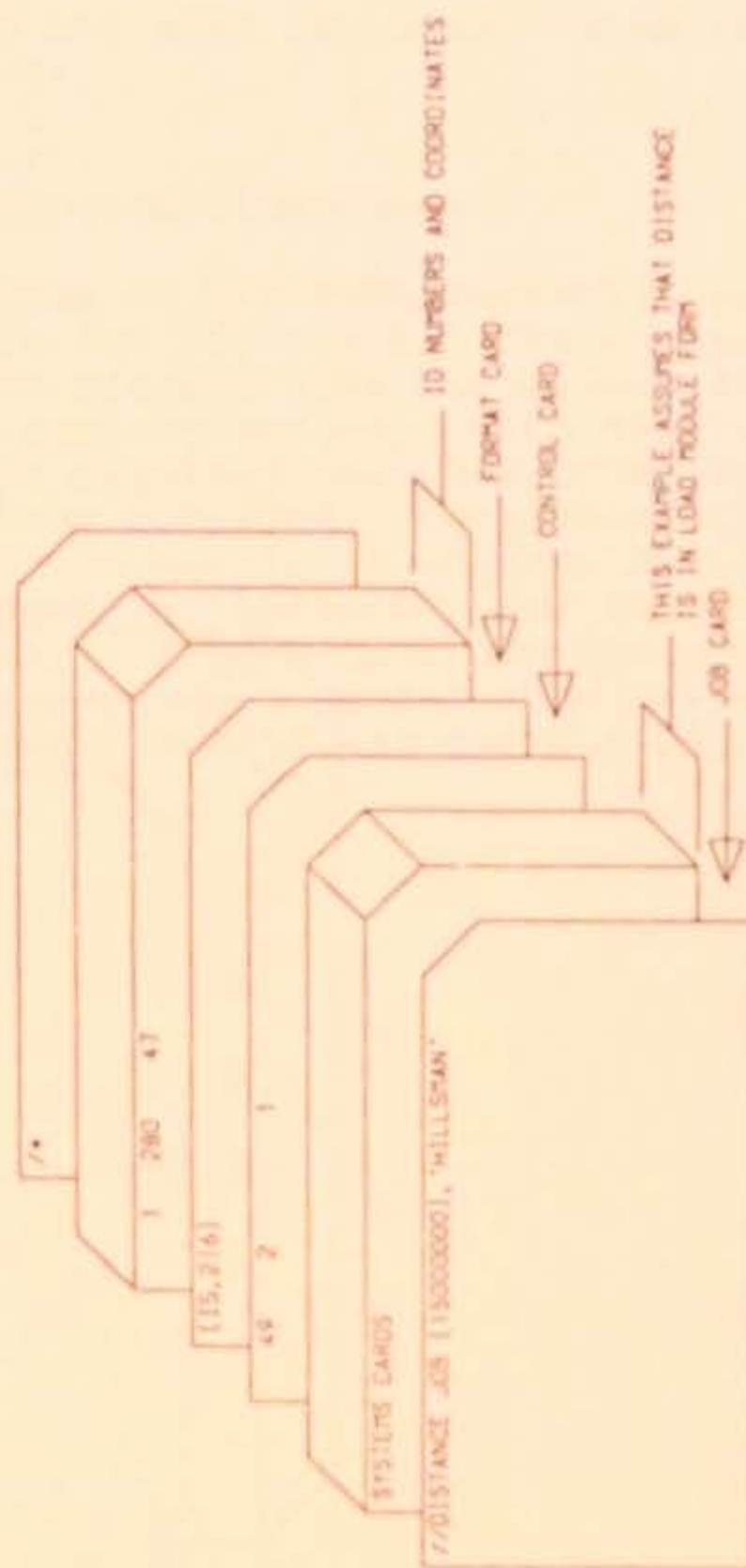
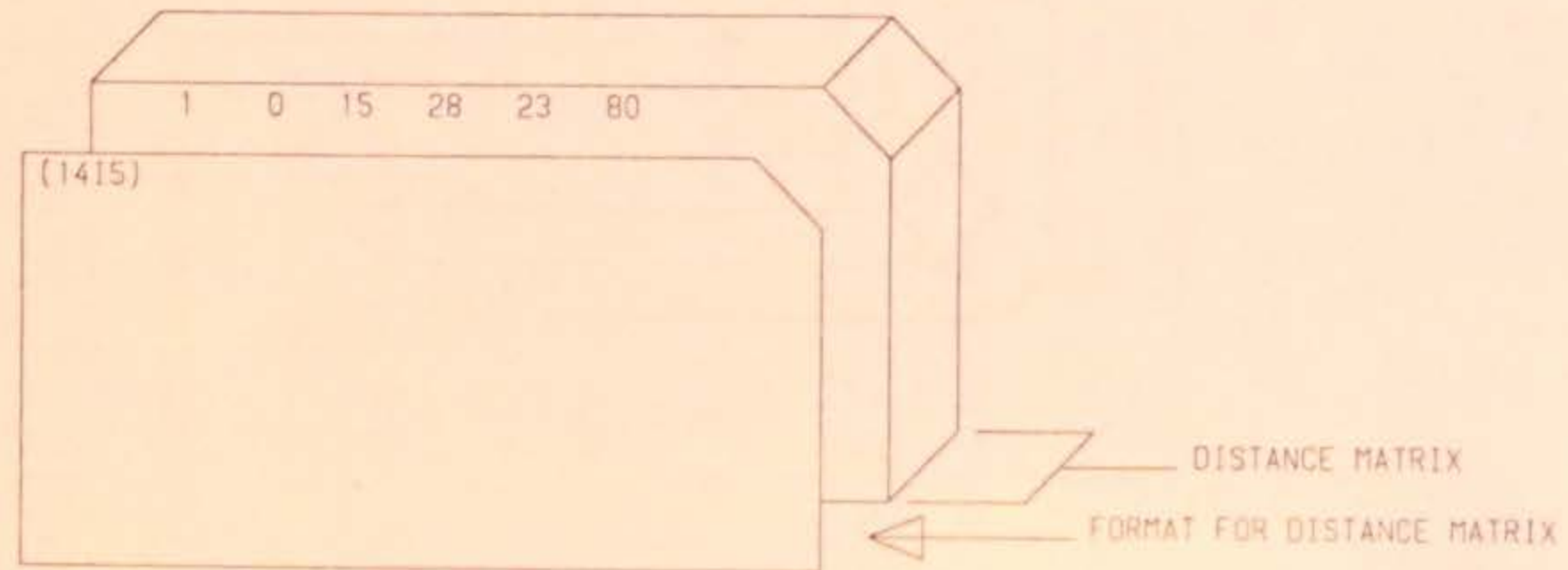




FIGURE I-3b Sample Punched Output  
from DISTANCE

16





## Error Conditions

The following conditions will produce an error message and stop the program.

1. End-of-file while reading the deck of node ID numbers and coordinates (3.0). Caused by miscounting the number of nodes (1.1) or by omitting a node from the deck.
2. Appearance of the same ID number twice in the node deck (3.0).

## Program Dimensions

The array dimensions of DISTANCE currently permit its use to compute a matrix for 150 nodes. The comment cards at the beginning of the program give directions for changing the dimensions of the program arrays. With its current dimensions, the program requires only 38K bytes of core storage after it has been compiled. Core requirements will be somewhat greater if the matrix is written on tape or disk, because of the need for buffers. For its present dimensions of 150 nodes, DISTANCE uses 38K bytes of core if all input and output use punched cards. The use of disks or tapes may require small additional amounts of storage.



FIGURE I-5 Sample Printed Output from DISTANCE

---

PROGRAM DISTANCE

WRITTEN BY EDWARD L. HILLSMAN

DEPARTMENT OF GEOGRAPHY

THE UNIVERSITY OF IOWA

FALL, 1975

---

COMPUTE A SQUARE DISTANCE MATRIX FOR 10 NODES

WRITE THIS MATRIX ON OUTPUT UNIT 7

LIST THE MATRIX

COMPUTE DISTANCES USING EUCLIDEAN OR STRAIGHT-LINE METHOD

READ NODE DATA USING FORMAT OF: (3I5)

ECHO CHECK OF INPUT DATA FOLLOWS

ID	X	Y
1	10	20
2	16	35
3	28	14
4	27	3
5	20	15
6	11	8
7	5	12
8	22	11
9	25	30
10	26	25

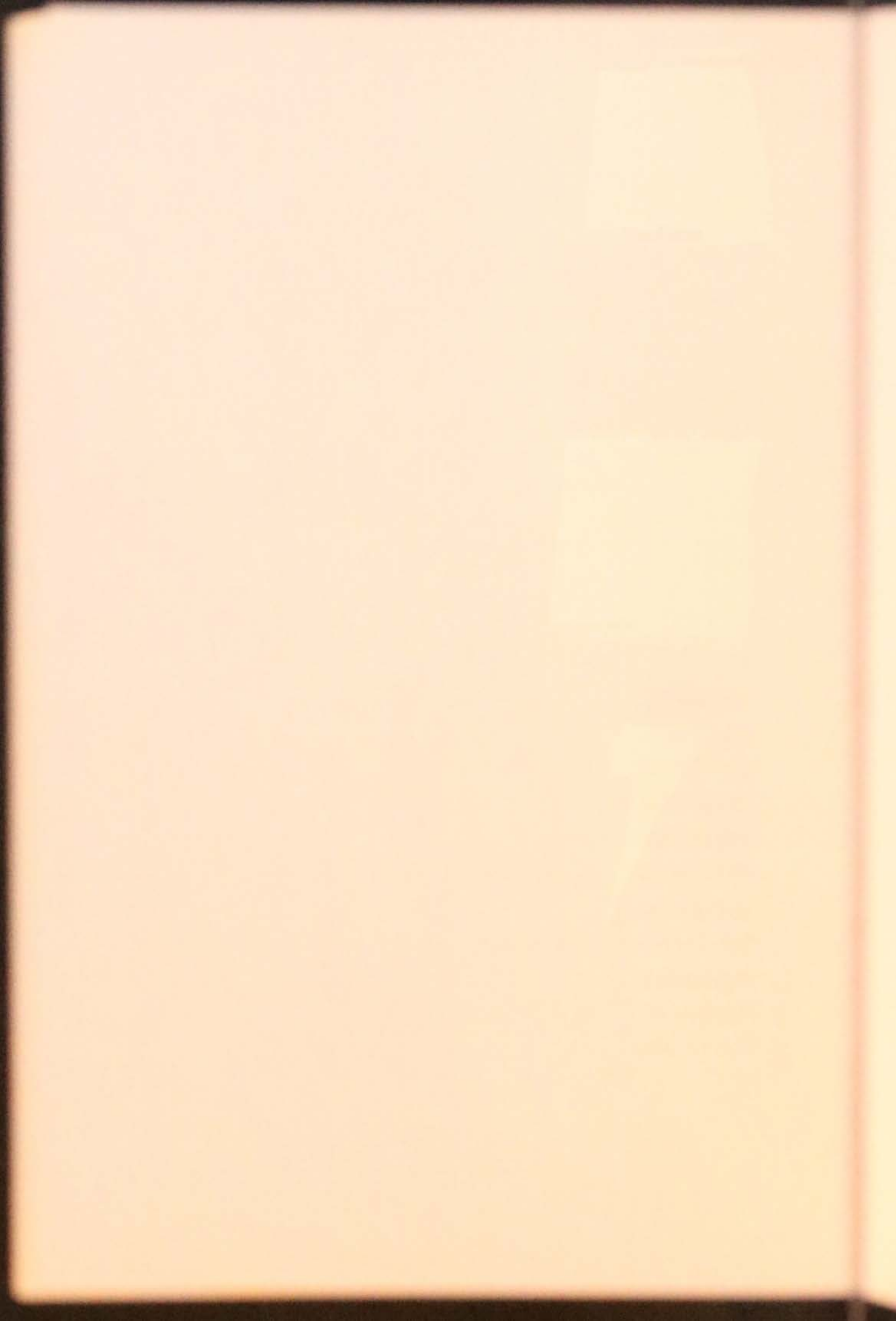


LISTING OF DISTANCE MATRIX

1	0	16	19	24	11	12	9	13	18	17
2	16	0	24	34	20	27	23	25	10	14
3	19	24	0	11	8	18	23	7	16	11
4	24	34	11	0	14	17	24	9	27	22
5	11	20	8	14	0	11	13	4	16	12
6	12	27	18	17	11	0	7	11	26	23
7	9	23	23	24	15	7	0	17	27	25
8	15	25	7	4	4	11	17	0	19	15
9	16	10	16	27	16	26	27	19	0	5
10	17	14	11	22	12	23	23	15	5	0

DISTANCE MATRIX IS COMPLETE







## CHAPTER II

### ALGORITHMS

The ALLOC programs contain five heuristic algorithms. The Maranzana algorithm, available in ALLOC V; the add algorithm, available in ALLOC VI; and the Teitz and Bart algorithm, available in both programs, have appeared in the literature before (Maranzana, 1964; Khumawala, 1973; Teitz and Bart, 1968).

The two unpublished algorithms were developed by Gerard Rushton and Edward Hillsman at The University of Iowa. One of these, to be termed the Hillsman-Rushton algorithm, was developed in 1973 to solve large p-median problems with data stored in a conventional matrix. The Hillsman-Rushton algorithm appears in both ALLOC programs. The second unpublished algorithm was developed in 1975 to solve a multiobjective location planning problem in rural Iowa. This algorithm, termed the trade-off algorithm, was described in an unpublished paper (Hillsman and Rushton, 1976). The algorithm appears only in ALLOC VI.

The first part of this chapter contains brief descriptions of the unpublished algorithms and more lengthy descriptions of the two unpublished ones. The second part presents execution times for the algorithms on different test problems. The third part discusses factors that should be considered when choosing an algorithm to solve a problem. These factors include execution times and known strengths and weaknesses of the algorithms. The third part of this chapter also discusses using a second algorithm to try to improve the results obtained with a first.

#### Descriptions

Descriptions of the three published algorithms will be quite brief. Users interested in greater detail should consult the original articles.

#### Maranzana Algorithm

The algorithm proposed by Maranzana (1964) works on the p-median data base in the following manner:

1. Read the starting solution.
2. Assign each node to its nearest center.
3. For each center:
  - a. Identify the nodes in its service area.



- b. Compute the aggregate distance from the nodes in 3a to each candidate node in the service area.
- c. Move the center to the candidate for which the aggregate distance in 3b is smallest.
4. If no centers were moved in step 3, stop. Otherwise return to step 2.

Steps 2-4 constitute one cycle of the algorithm. The algorithm converges, and usually stops within three or four cycles.

#### Teitz and Bart Algorithm

The algorithm proposed by Teitz and Bart (1968) operates on the p-median data base in the following manner:

1. Read the starting solution.
2. Assign each node to its nearest center.
3. For each candidate that does not have a center:
  - a. Substitute it for each of the existing center locations.
  - b. Compute the aggregate distance that would result for each of the substitutions in step 3a.
  - c. If moving any of the centers to the candidate would reduce aggregate distance, move the center that would yield the largest reduction for the problem.
  - d. If a center was moved in step 3c, reassign each node to its nearest center.
4. If no centers were moved in step 3, stop. Otherwise, repeat step 3.

Steps 3-4 constitute one cycle of the algorithm. Like the Maranzana algorithm, the Teitz and Bart algorithm usually stops within three or four cycles.

It should be noted that most of the notation in the original article involves calculating the effect of substituting a candidate for each of the centers in step 3b without actually moving each of the centers to the candidate. The corrections to the algorithm by Rushton and Kohler (1973) involve these same calculations.

#### The Add Algorithm (or Greedy Heuristic)

The add algorithm has been proposed for several location-allocation problems by Kuehn and Hamburger (1963), Church and ReVelle (1974), and Khumawala (1973). As programmed for the p-median problem in ALLOC VI, the algorithm



requires a starting solution of at least one center. It operates in the following manner:

1. Read the starting solution and the number of additional centers to be located.
2. Assign each node to its nearest center.
3. For each candidate, compute the reduction in aggregate distance that would result if a center were to be added at that location.
4. Add a center at the candidate yielding the greatest reduction in step 3.
5. If more centers are to be located, return to step 2. Otherwise, stop.

Users of ALLOC VI may also invoke the add algorithm to add centers at specified locations, in order to evaluate the performance and service areas of proposed additions to a pattern of centers. The user may retain these centers in the pattern for further analysis or drop them once they have been evaluated. More details appear in the discussion of ALLOC VI.

#### Hillsman-Rushton Algorithm

The Hillsman-Rushton algorithm was designed to solve large p-median problems--those with more than 100-200 nodes. As the number of nodes in a p-median problem increases, the computer storage and computer time required for solving it both increase. Increasing the number of centers in a problem will also increase computer time. This is particularly true of the Teitz and Bart algorithm, which requires a great deal of storage and a minute or more of time to solve large problems in ALLOC V. The Maranzana algorithm, on the other hand, can solve the same problem in a few seconds. Because it needs only a small amount of the data for the problem at one time, the Maranzana algorithm could be programmed to store most of the weighted distance matrix out of core to reduce core storage requirements. The Maranzana algorithm rarely finds very good answers to large p-median problems, however.

The Hillsman-Rushton algorithm is a compromise between these two algorithms. It is nearly as effective as the Teitz and Bart algorithm in finding good answers to p-median problems. While the algorithm requires more computer time than the Maranzana algorithm, it requires substantially less time than the Teitz and Bart. Like the Maranzana algorithm, the Hillsman-Rushton algorithm could be programmed to work efficiently from a weighted distance



matrix in external storage to reduce core storage requirements, but this has not been done.

The Hillman-Bushon algorithm begins with a starting solution and assigns each node to its nearest center. The algorithm then begins the first of two phases. In its first phase, the algorithm identifies the most expendable center in the solution and moves that center to the candidate where it will lead to the greatest reduction in aggregate distance. The expendability of a center is the amount that aggregate distance for the problem would increase if the center were removed, or dropped, from the solution without being replaced by a center at some other candidate node. The center with the lowest expendability value is the most expendable center, because dropping it would have the least effect on aggregate distance if a center actually had to be dropped.

The first phase of the Hillman-Bushon algorithm identifies the most expendable center, finds the best new location for it, moves the center to the new location, and reassigns each node to its new nearest center after the move. Moving a center changes the expendability of many centers in the solution, increasing the expendability values of centers near its old location, and reducing the expendability values of centers near its new location. The usual result is to make a different center the most expendable one. The first phase thus repeats the procedure for the new pattern of centers and continues until moving the most expendable center will not reduce the aggregate distance further.

The first phase is similar to the Teltz and Bart algorithms, in that it can move centers great distances across the network to correct region-wide imbalances in the distribution of centers. The first phase is efficient in this effort, because it concentrates on trying to move the centers that, intuitively, are the ones most likely to be poorly located. It may miss some good moves involving the less expendable centers, however. The Teltz and Bart algorithm considers all of the moves that the first phase does, and more besides. Many of these additional moves are unlikely to yield much reduction in aggregate distance, however, and each one requires computer time to consider.

Having tried to establish a rough balance of centers throughout the network, the algorithm enters a second phase that resembles the Maranzana algorithm. The second phase examines each center in turn and tries to move it to one of the candidate nodes within its service area. The algorithm moves the center to the candidate where it will most reduce the aggregate distance for the problem, reassigns



each node to its new nearest center, and then tries to move another center in the same manner. This phase of the algorithm continues until no center can be relocated within its service area to reduce the aggregate distance. Then, on the chance that the second phase may have exposed or developed another regional imbalance in the pattern of centers, the algorithm begins its first phase again. The algorithm alternates between its two phases until neither one can improve the solution to the problem.

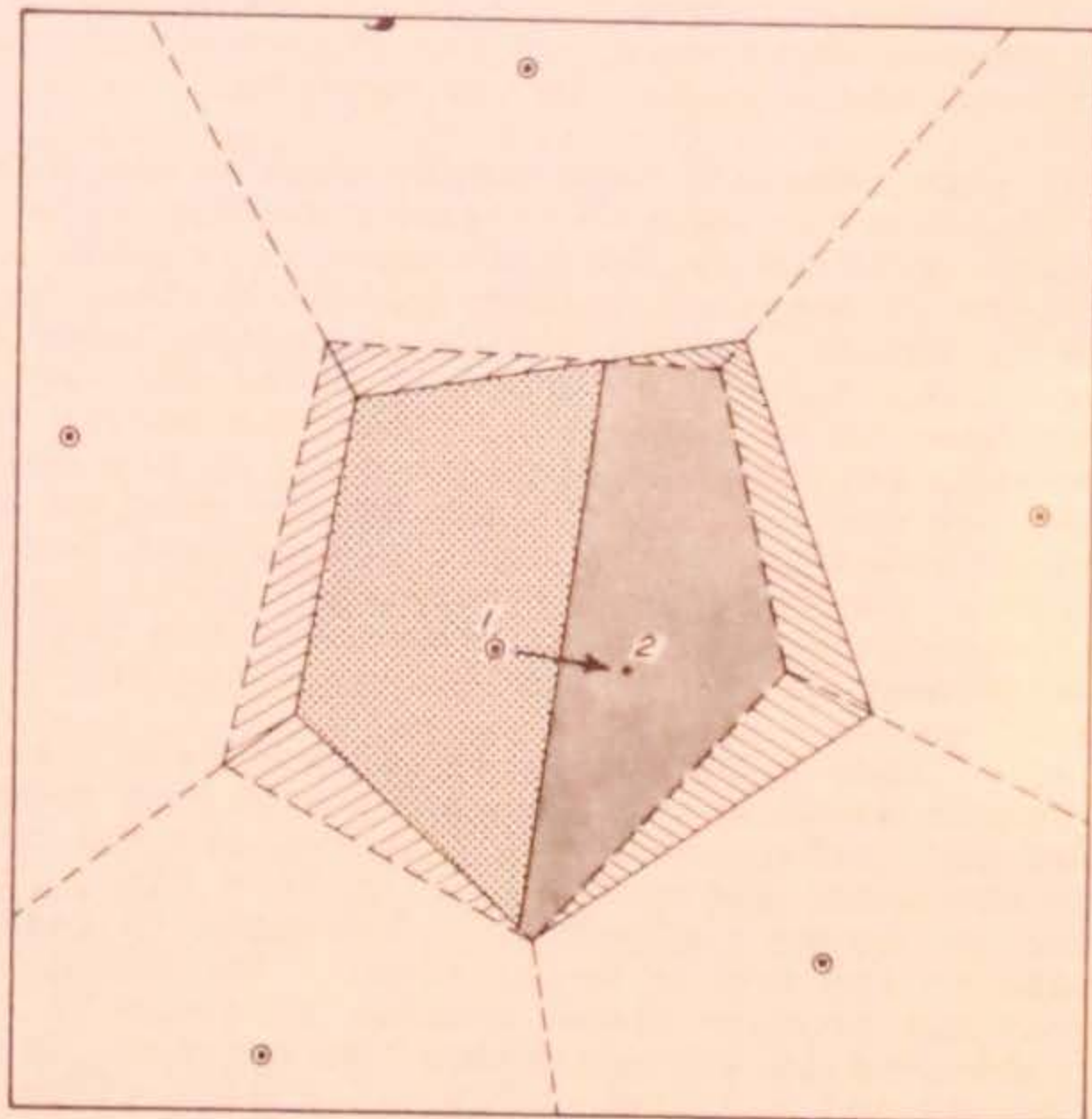
In trying to relocate each center within its service area, the second phase of the Hillsman-Rushton algorithm asks the same question as the Maranzana algorithm, but it considers more information in determining whether or not a center can be moved. The Maranzana algorithm looks for reductions in aggregate distance only among the nodes in the service area to be relocated. When the center is moved, however, it becomes nearer to some of the nodes served by other centers, and it may become near enough to some of these nodes to become their new nearest center (Figure II-1). The Maranzana algorithm ignores these "external" savings, and thus it underestimates the reductions in aggregate distance from moving a center.

Similarly, when a center is moved away from a particular node in its service area, the center may be moved far enough that some other center will become the new nearest center for the node (Figure II-1). In this case, the increase in aggregate distance from the node is less than the increase to the center being moved. The Maranzana algorithm does not measure these smaller increase in aggregate distance, and thus it overestimates the increase that results from moving a center. The net effect of overestimating these increases, and underestimating reductions in aggregate distance, is to consider many potential relocations of a center as bad decisions, or to prefer a fair relocation to a much better one.

The second phase of the Hillsman-Rushton algorithm calculates the true reductions and increases in aggregate distance for each potential relocation. It thus should find a better solution to the problem than does the Maranzana algorithm when the two procedures begin with the same starting solution. Comparison of the second phase of the algorithm with the Maranzana, using matched starting solutions in three test problems, confirms the superiority of the second phase over the Maranzana algorithm. Table II-1 contains the results of these tests. The second phase usually found a pattern with lower aggregate distance than did the Maranzana.



FIGURE II-1 Comparison of Calculations Performed by the Maranzana Algorithm and the Second Phase of the Hillsman-Rushton Algorithm







- ⊙ Center
- Service Area boundaries with center at 1
- Service Area boundaries with center at 2
-  Zone where aggregate distance will increase and nodes will remain in this center's service area. (Correctly counted by Maranzana and the second phase.)
-  Zone where aggregate distance will increase and nodes will become nearer to other centers (Increase overestimated by Maranzana)
-  Zone where aggregate distance will decrease and nodes will remain in the same service area. (Correctly counted by Maranzana and the second phase.)
-  Zone where aggregate distance will decrease because nodes change service areas. (Ignored by Maranzana)



TABLE II-1

Comparison of Hillsman-Rushton Algorithm, Second Phase, with the Maranzana Algorithm

Test Problem	Phase Two outperformed Maranzana	Phase Two equaled Maranzana	Phase Two outperformed by Maranzana
49 nodes 5 centers 75 runs	68	3	4
49 nodes 10 centers 75 runs	70	1	4
150 nodes 21 centers 25 runs	25	0	0

Source: Compiled by author.

The effectiveness of the complete Hillsman-Rushton algorithm is discussed in the section on choosing an algorithm.

#### The Trade-off Algorithm

The trade-off algorithm is quite different from the other algorithms in the ALLOC programs. The algorithm was designed to make trade-offs between the accessibility of a pattern of centers, as measured by aggregate distance, and an index of the characteristics for the candidates where centers are located. If the candidates are towns in a rural area, a town and index score could include the size of the town, the presence of activities that could support the center being located, and the cost of opening a center in the town. If the candidates are land parcels in an urban area, each parcel's index score might include the size of the parcel and the availability of utility services to it. For convenience, the value of the index score will be termed a site characteristic or the suitability of the candidate. The algorithm requires that this score be scaled from zero to one for each candidate. The trade-off algorithm locates centers to minimize the aggregate distance to them and to maximize the aggregate suitability of



their locations. Rushton, et al. (1976), discuss an example of this type of problem that was encountered in planning the locations for primary health care in rural Iowa.

A second difference between the trade-off algorithm and the other algorithms is the philosophy behind it. The steps in the algorithm mimic a process of making decisions. In this process, it is assumed that decisionmakers are concerned with aggregate distance and suitability for the entire pattern of centers, and with the local changes in these measures that result as centers are moved around to produce a better location pattern. It is not assumed that calculations made for the aggregate measures are entirely consistent with the calculations made in the vicinity of the centers being moved. The two types of calculations will be termed global and local calculations, respectively. Hillsman and Rushton (1976) have discussed the relationship between the local and global calculations in greater detail, and it is hoped that their discussion will eventually be published.

A description of the decisionmaking scenario and the trade-off algorithm follows. It borrows heavily from the Hillsman and Rushton (1976) paper, but it does not attempt to justify their assumption of the decisionmaking process or of any of the specific steps in the algorithm's operation.

The scenario simulated by the trade-off algorithm begins with a group of decisionmakers, for whom an outside consultant has just recommended an optimum solution to the location problem. The consultant lacks the detailed information which the decisionmakers have about their region, and he has developed the global solution without this knowledge. Accordingly, some aspects of the plan seem counter-intuitive to some of the decisionmakers. Other aspects, either because of political reasons or for other reasons not considered by the consultant, are simply unacceptable. The decisionmakers use their local knowledge and preferences, and begin to propose small changes to the plan, either to check their intuition or to correct obvious faults. Some of these changes, when evaluated, are found to improve the plan and are incorporated into it. The process of proposing, evaluating, and accepting small changes continues until no small change can improve the plan. This kind of process seems both common and reasonable. When the decisionmakers have access to an information system to help in the evaluation of changes, this process combines both optimization behavior and an unstructured sensitivity analysis. The trade-off algorithm is an approximate, more structured simulation of this kind of process.



The scenario assumes that the consultant has completely ignored the accessibility of the centers, as measured by aggregate distance, and has suggested locating the centers at the locations with the highest aggregate site characteristics. Accordingly, the overriding concern of the decisionmakers is to meet the obvious accessibility needs of the plan without sacrificing too much of its high site characteristics. In the scenario, the decisionmakers agree to take turns around the table, with each one proposing changes that are important to him. In the following discussion, consider a problem of locating five centers, and focus on the turn of one of the decisionmakers.

The decisionmaker, representing a part of the region that lacks any center in the current version of the plan, selects a location X that she believes should have a center (Figure II-2). Moving any of the existing centers to X will obviously improve aggregate distance in the immediate vicinity of X. Moving the center from either D or E will increase the total distance for the region as a whole (row 3 of Table II-2), because the increase in total distance near those centers would outweigh the decrease near X. The rest of the decisionmakers, because they want to improve the accessibility of the whole plan, will not permit either of these two centers to be moved to X. Thus, the centers at D and E may be ignored for the rest of the example. Moving a center from A, B, or C to X would reduce total distance, however. If the loss in site characteristics does not outweigh the gain in accessibility, the other decisionmakers would allow such a move. To this point, the scenario has involved only global calculations. The remaining calculations in the process are local ones.

To determine whether the gain outweighs the loss, the algorithm standardizes the accessibility gain that would accrue from moving each of the centers. For the center at A, this involves computing a ratio between the increase in total distance that results near A when the center is moved, (row 1 of Table II-2), and the decrease that would occur near X when the center is moved there (row 2 of Table II-2). Subtracting this ratio from one yields a measure of relative accessibility change, ranging from zero to one with small relative changes receiving smaller values than larger ones. This range of values is the same as that assumed earlier for the site characteristic values. The algorithm computes this measure of relative accessibility change for all three centers to determine which moves are better than others (row 4 of Table II-2).

To model the local calculations of the group of decisionmakers, the algorithm uses a single, elliptical



Table II-2

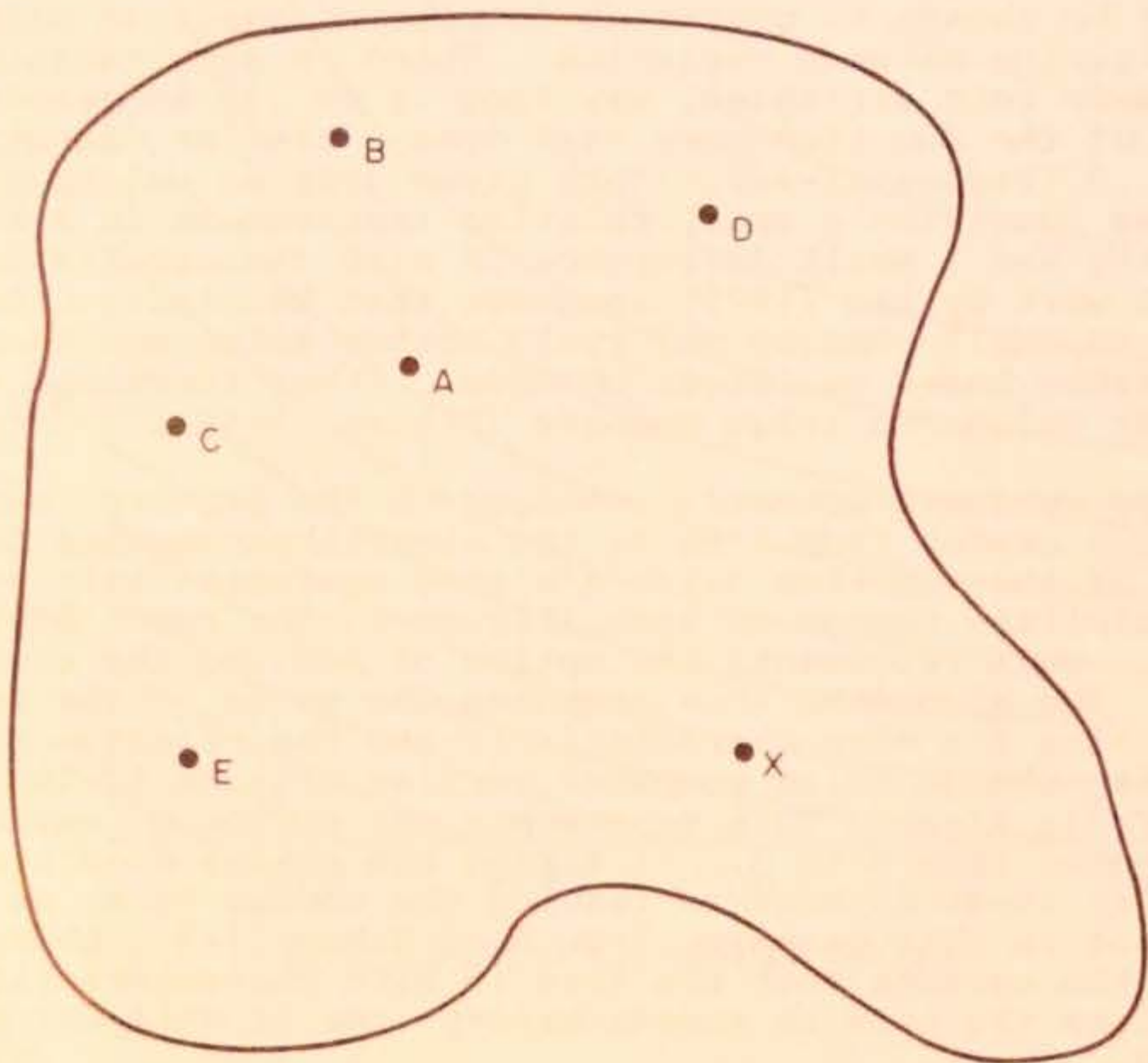
## Sample Calculations for the Trade-off Algorithm

Row	Interpretation	Center Location				
		A	B	C	D	E
1.	Accessibility loss if column center were removed, instead of moved	720	400	425	700	750
2.	Accessibility gain near X given absence of column center	800	500	500	400	400
3.	Change in total distance moving column center to X (row 1 - row 2)	-80	-100	-75	+300	+350
4.	Standardized accessibility gain 1 - row 1/row 2	.1	.2	.15		
5.	Site characteristic of column center	.8	.8	.7		
6.	Value of $T(x,y)$ for $x = 0, y = \text{row 5}$ (i.e., no change)	-.5400	-.5400	-.5900		
7.	Value of $T(x,y)$ for $x = \text{row 4}, y = .6$ (i.e., move to X)	-.5650	-.4800	-.5213		
8.	Difference, row 7 - row 6	-.0250	.0600	.0687		
9.	Greatest positive value, row 8			.0687		

Source: Hypothetical data for region in Fig. C-4. Potential location X is assumed to have a site characteristic value of .6 for the computation in row 7. Rows 1-3 are global calculations. The remaining rows are local calculations.



FIGURE II-2 Hypothetical Region for Trade-off  
Algorithm Calculations





function to evaluate and rank potential local changes to the pattern of centers. The vertical axis in Figure II-3 measures the site characteristics of places involved in changes. The horizontal axis measures the relative accessibility change. An ellipse has three qualities that make it attractive for the simulation.

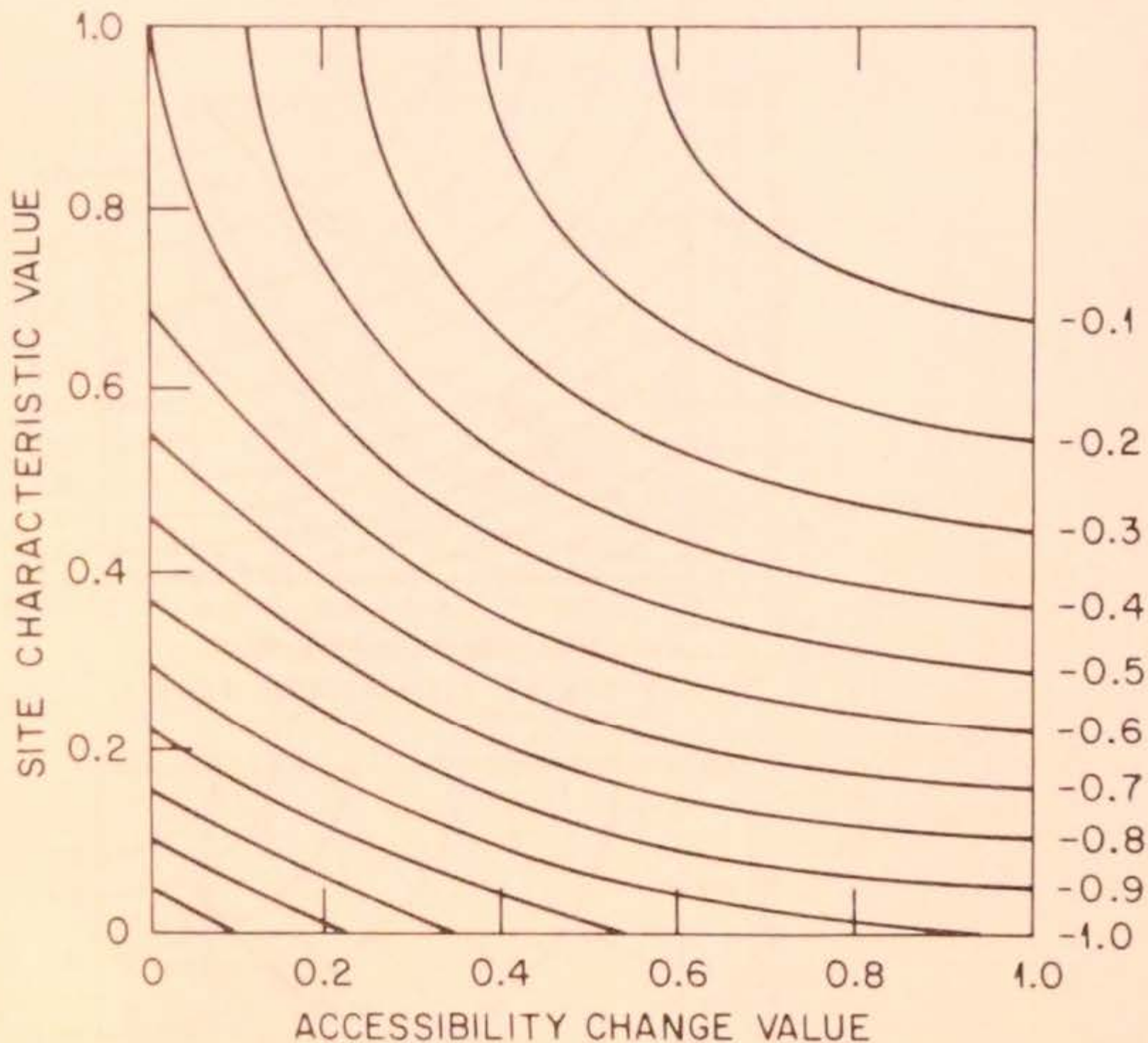
First, the standard form of the function has four parameters that may be varied to give different relative importance to the two variables. Most of the work with the function (Rushton et al., 1976) has used the parameter values given in Figure II-3. Second, because the two variables range only from zero to one, the parameters may easily be chosen to guarantee decreasing marginal rates of substitution between variables. Third, a simultaneous unit change in both variables, say from .1 to .3, increases the value of the function more than does a similar change from .6 to .8 (Figure II-4a). This gives greater weight to changes involving a small relative improvement in accessibility and a small difference in site characteristics. Recent work by Lin (1975) suggests that heuristic algorithms based on small changes may yield better solutions than algorithms based on larger changes. Other functions give greater weight to large changes (Figure II-4b).

To evaluate potential changes to the pattern, such as moving a center from A to X, the algorithm computes the value of the function using A's site characteristic and an accessibility change of zero (Figure II-5a; row 6 of Table II-2). This represents the option of leaving the center at A. The algorithm then computes the value of the function using X's site characteristic and the relative accessibility change value computed earlier (Figure II-5b; row 7 of Table II-2). This represents the option of moving the center from A to X. If making the change does not score at least as high as leaving the center at A, as it does not in this example, (row 8 of Table II-2), then the algorithm assumes that the loss in site characteristics outweighs the gain in accessibility, and it will not move A to X.

The same type of calculation is made to evaluate moving a center from B or C to X (Figure II-6; Table II-2). Moving either of these centers scores higher than leaving them where they are. The algorithm chooses the larger difference between the values of moving a center and not moving it (rows 8 and 9 of Table II-2), and moves C to provide the better change to the pattern. It then proceeds to let another decisionmaker take a turn in proposing a change to the new pattern of centers.



FIGURE II-3 Preference Function Used to Evaluate Local Changes to a Pattern of Centers



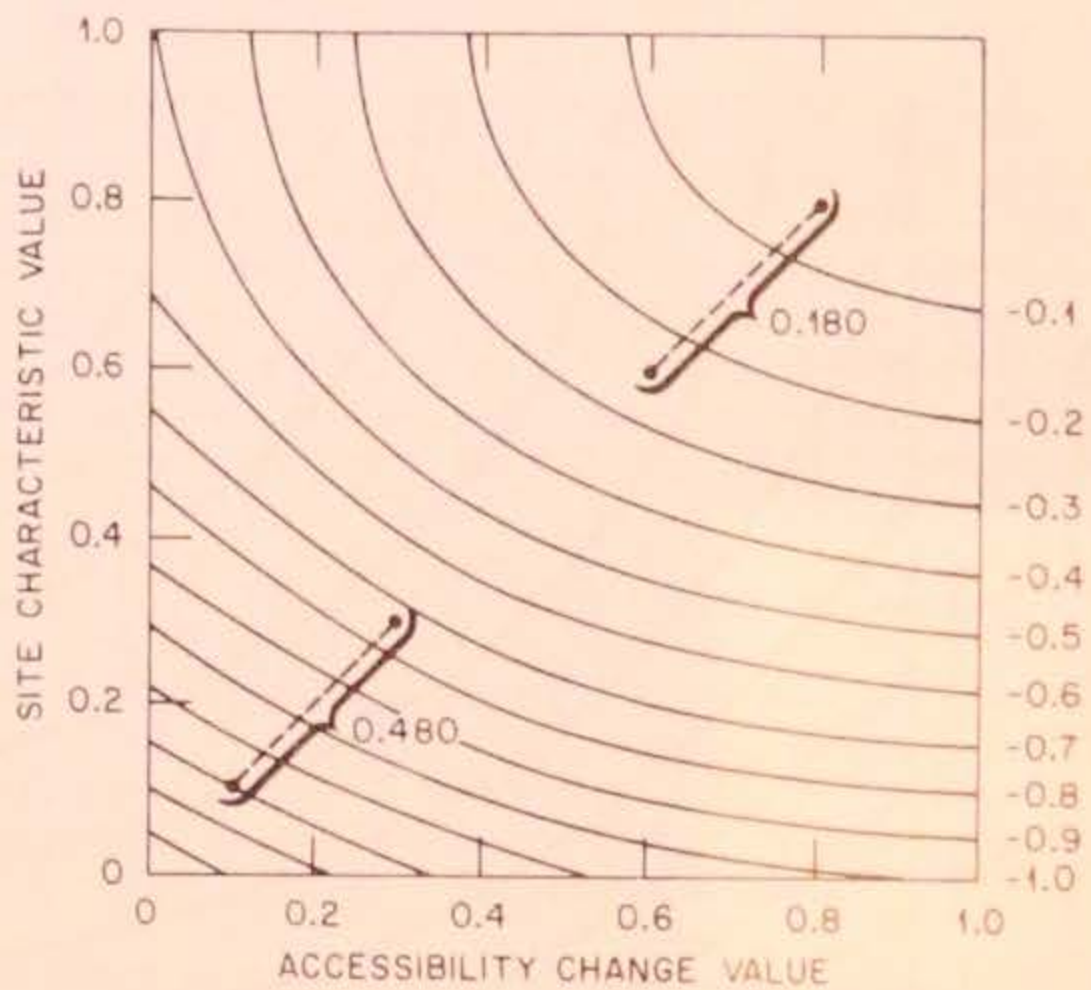
$$T(x, y) = - \left[ \frac{(x-a)^2}{c} + \frac{(y-b)^2}{d} \right]$$

with:  $a=b=d=1$   
 $c=2$

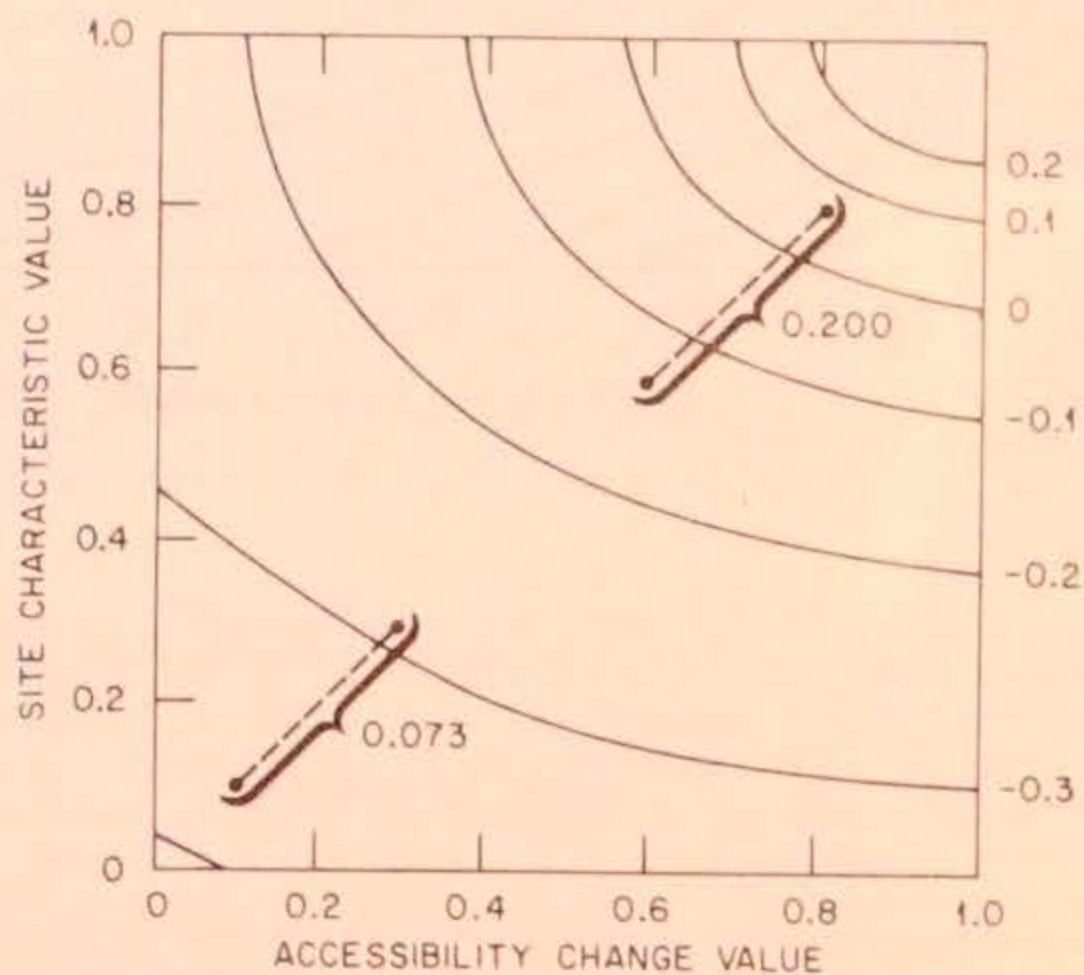
$y$  is the site characteristic value, and  $x$  is a measure of accessibility gain defined as 1--increase in total distance if a center is removed from solution--decrease in total distance if the center removed is replaced.



FIGURE II-4 Comparison of Unit Changes for Two Local Preference Functions



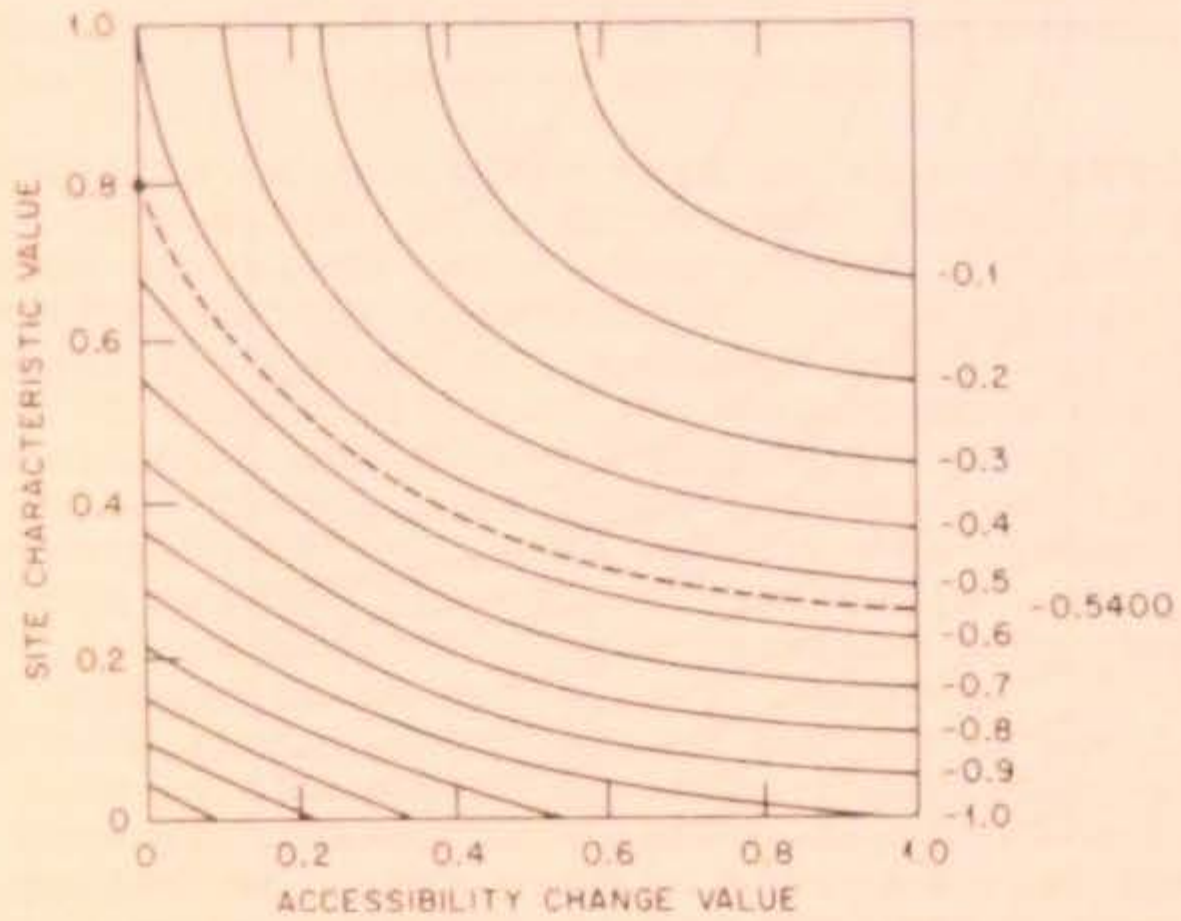
(a)  $T(x,y)$  from FIGURE II-3



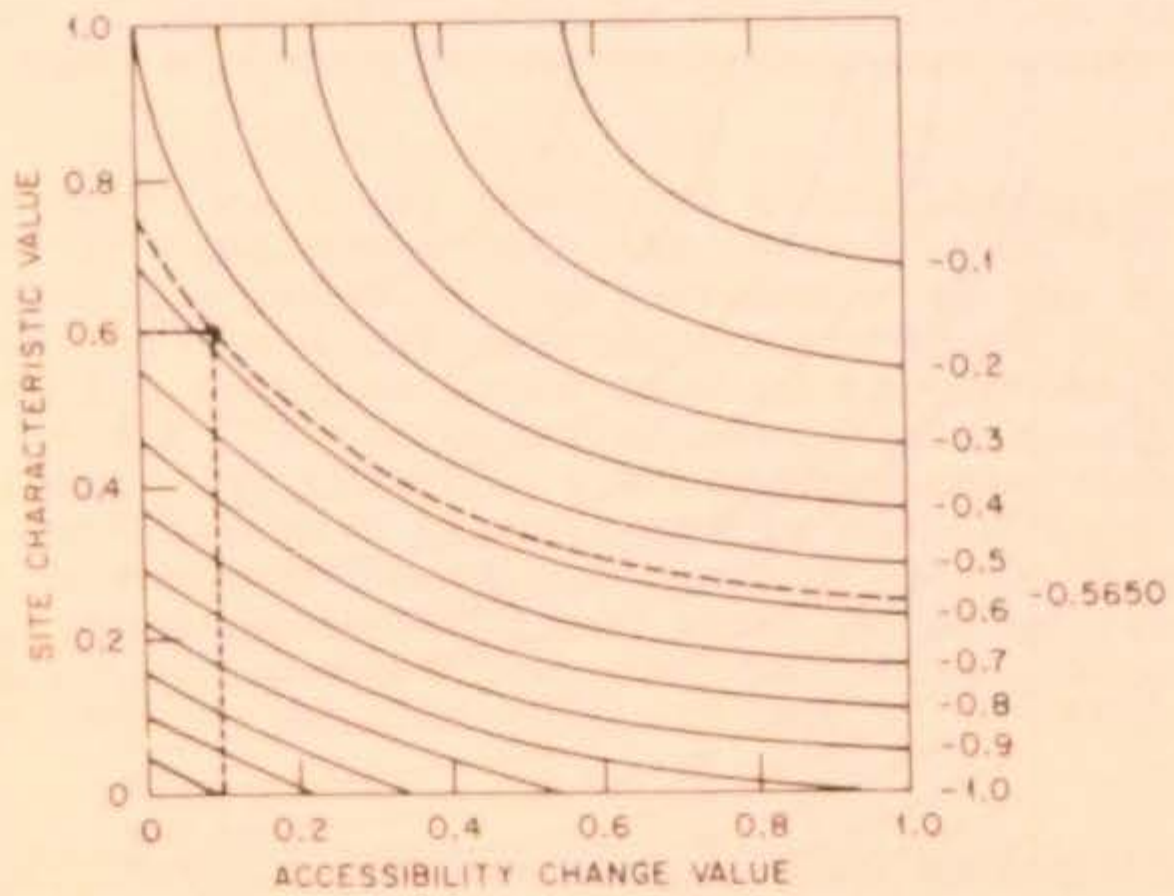
(b)  $S(x,y) = -\frac{\log[10T(x,y)]}{10 \log(2)}$



FIGURE II-5 Local Preferences for Center at A



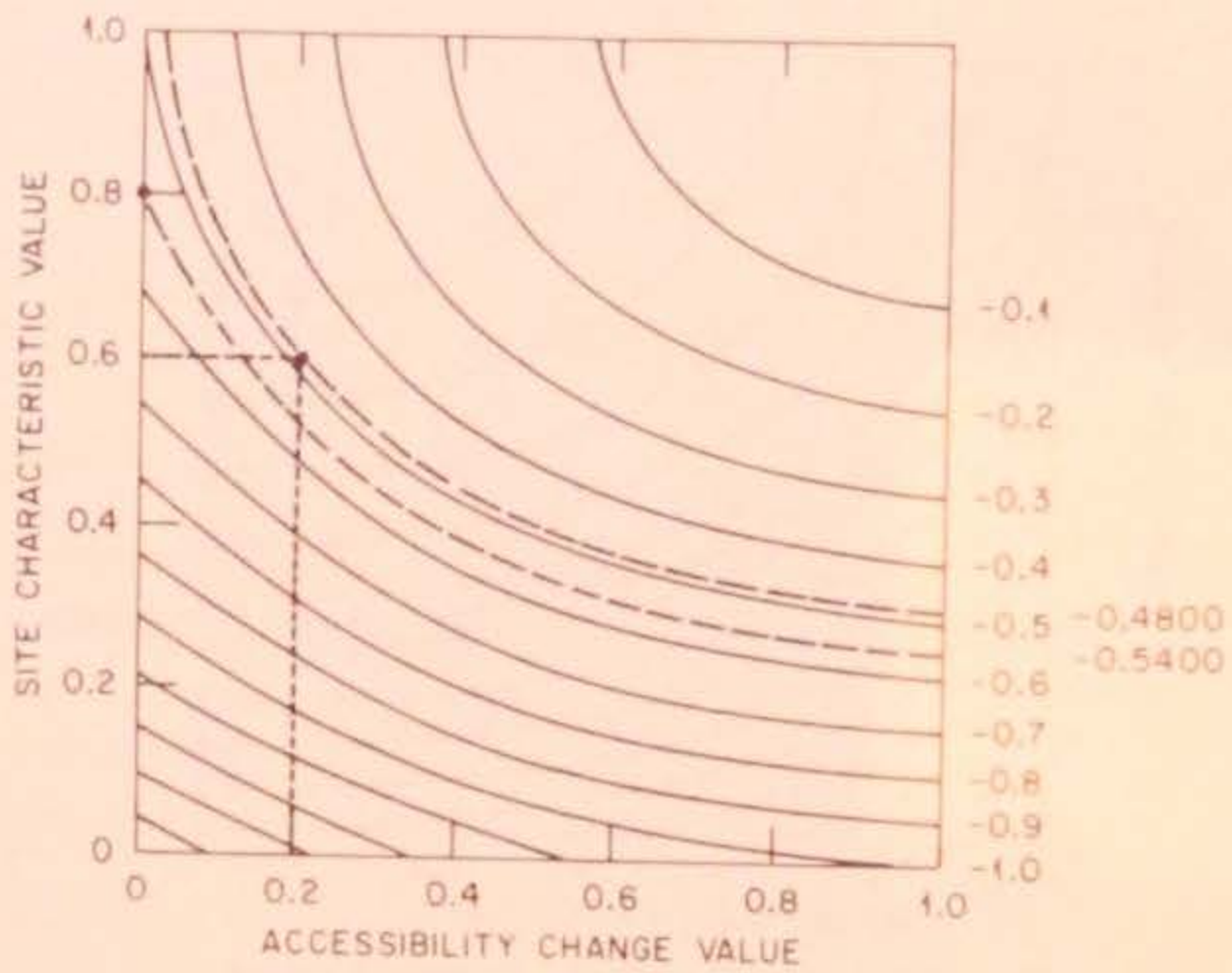
(a)  $T(x,y)$  leaving center at A



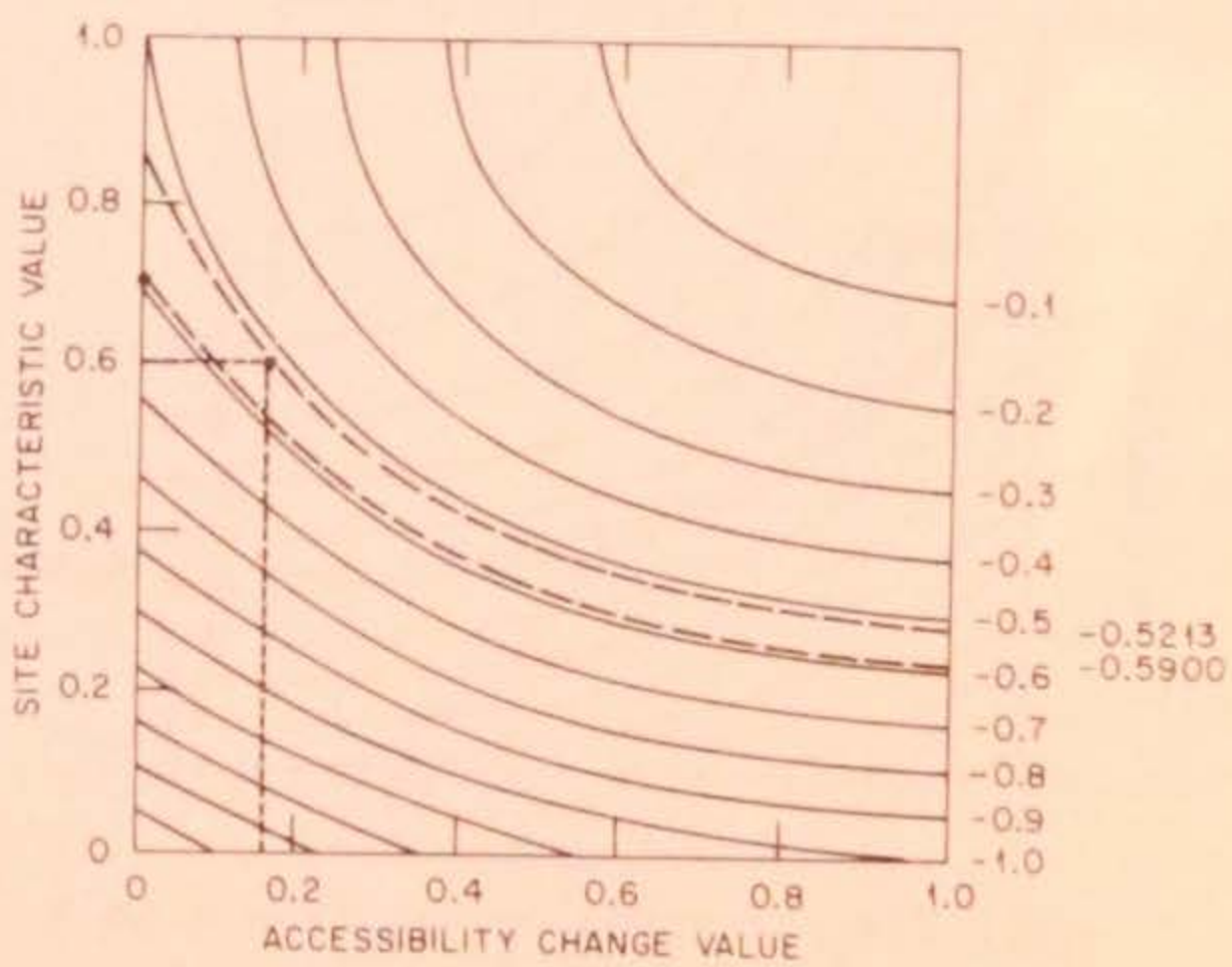
(b)  $T(x,y)$  moving center from A to X



FIGURE II-6 Local Preferences for Centers at B and C



(a) Calculations for center at B



(b) Calculations for Center at C



The algorithm considers every potential center location once. If it moves one or more centers during this sequence of examinations it will begin the sequence again, and it will stop only when it cannot move any center. This sequence for considering moves is identical to that of the Teitz and Bart algorithm. The difference between the two algorithms is in the way that they evaluate potential moves. The resulting pattern of centers is a compromise between the plan with the best site characteristics and the plan with the greatest accessibility.

There is no guarantee that an algorithm would ever stop if it operated on a truly local basis, without any reference at all to the global pattern. However, the trade-off algorithm does make a global calculation when it determines whether or not a change would improve the global measure of accessibility. The requirement that global accessibility must improve, when applied to a finite number of potential center locations, ensures that the algorithm will eventually stop. This requirement prevents the algorithm from trading the two attributes in both directions. That is, the algorithm cannot increase site characteristics at the expense of accessibility. However, it can increase the site characteristics of the pattern. The first few moves made by the algorithm, from the pattern with the best site characteristics, necessarily degrade the overall site characteristics of the pattern. Frequently, however, the algorithm moves some centers back to the high-valued places later in its operation. It thus can improve the global measures of both attributes at once, although it need not prefer such a change to one that improves accessibility alone.

As noted earlier, the local calculations are not necessarily consistent with the global ones. For example, increasing the value of the parameter  $c$  in the trade-off function for the local calculations (Figure II-3) will increase the relative importance of the accessibility change measure in the calculations. This usually leads to a final solution that has a lower aggregate distance and a lower aggregate site characteristic than before. In some cases, however, increasing  $c$  has led to solutions with higher aggregate distance and site characteristic values. As yet, no pattern has been detected in the inconsistency between the two sets of calculations.

#### Execution Times and Core Storage Requirements

Table II-3 contains execution times for the Maranzana, Teitz and Bart, and Hillsman-Rushton algorithms in ALLOC



IV, ALLOC V, and ALLOC VI. Seventy-five random starting solutions were generated for a 49-node, 5-center problem, and each algorithm in ALLOC IV and ALLOC V was run from these solutions. The same procedure was used for a 10-center problem using the same 49-node data base. The distances and weights for this data base are shown in the sample output for ALLOC V. Ten random starting solutions were generated and used to compare execution times for a 150-node, 21-center problem. The algorithms from all three programs were on the second two test problems. Each entry in Table II-3 is the average time needed to read the data base once and then solve the seventy-five or ten trial problems. The time needed to solve a single problem, including input, has differed from these averages by as much as thirty percent. No maximum distance constraints or location constraints were used with ALLOC IV or ALLOC V. The execution times and core storage requirements for ALLOC VI depend significantly upon the use of a maximum distance constraint, and loose maximum distance constraints were used when that program was run. The effect of maximum distance constraints on ALLOC VI will be discussed in Chapter IV.

All times were obtained using The University of Iowa's IBM 360-65 computer and, with two exceptions noted in the table, all are based on the IBM Fortran G compiler. The two exceptions used the IBM Fortran H compiler, with the parameter OPT = 2. When it is available for use with ALLOC IV and ALLOC V, the H compiler is clearly preferable to the G because of its markedly lower execution times. The H compiler is particularly recommended if the programs can be stored and run as load modules. Although no times are reported using the H compiler with ALLOC VI, tests made during the development of that program indicated that the two compilers give virtually identical execution times. ALLOC VI does not use a matrix to store its data base, and this lack of a matrix appears to be the main reason that the H compiler makes little improvement in its running time.

The execution times of ALLOC V are markedly lower than those of ALLOC IV except for the Maranzana algorithm, whose times are very slightly higher. The source code for ALLOC IV was deliberately kept simple to make easier the development and testing of new algorithms. The source code for ALLOC V was written to make the Teitz and Bart and Hillsman-Rushton algorithms run faster, but this was accomplished at the expense of greater program complexity. The time differences for the Maranzana algorithm cannot be accounted for with any precision, and they are small enough that they may be ignored for most analyses. For normal use, then, ALLOC V's faster execution makes it preferable to ALLOC IV.

Just as ALLOC V is faster than ALLOC IV, so ALLOC VI is faster than ALLOC V. Again, speed was gained at the



Table II-3

Execution Times in Seconds for Different Algorithms in the ALLOC Programs

Problem	Algorithm	ALLOC IV	ALLOC V	S	ALLOC VI
49 nodes 5 centers (75 trials, averaged)	Maranzana	.28	.30		--
	Teitz and Bart	1.83	1.33		--
	Hillsman-Rushton	1.51	1.15		--
49 nodes 10 centers (75 trials, averaged)	Maranzana	.34	.38		--
	Teitz and Bart	2.38	2.04	100	.90**
	Hillsman-Rushton	1.96	1.52	100	.57
150 nodes 21 centers (10 trials, averaged)	Maranzana	2.05	2.06		--
	Teitz and Bart	61.67	46.48	100	19.93
		40.88*	25.76*	70	11.10
	Hillsman-Rushton	24.88	18.73	100	5.61
70				3.60	

Source: Compiled by author. All times from The University of Iowa's IBM 360/65 computer. S is the maximum distance used when running ALLOC VI.

\*Fortran H compiler.

\*\*Job terminated after 69 solutions because of page limit.



expense of greater complexity. Even when ALLOC V was run with the H compiler, the Teitz and Bart algorithm in ALLOC VI saved from 22 to 57 percent of the execution time required by the one in ALLOC V. When ALLOC V was run with the G compiler, some of the savings from ALLOC VI were even greater. As a later section of the appendix will show, however, substantially more time and effort are needed to prepare a data base for ALLOC VI than for ALLOC V. Since the absolute time savings for ALLOC VI are small, even on the 150-node test problem, the extra time and effort needed to prepare small data bases for that program may make ALLOC V a more attractive choice if the Fortran H compiler is available.

ALLOC V solved the two 49-node test problems in 70 K bytes of core storage on the IBM 360. Using the G compiler, it solved the 150-node problem in 148 K bytes. The H compiler required only 144 K bytes. The corresponding figures for ALLOC IV were 70 K, 144 K, and 140 K bytes. Using the G compiler, ALLOC VI solved the 49-node problem in 74 K bytes, and the 150-node problem in 134 K. The core storage requirements depend in part on the use of halfword storage, as permitted by IBM Fortran. Halfword storage is not part of ANSI Standard Fortran, however, and the programs may require more core storage on non-IBM computers. In addition, core storage requirements depend in part upon the number of input units from which the data are read. In the cases cited here, all input was from punched cards. Reading data from tape or disk requires additional space for input buffers.

As noted earlier, ALLOC V and ALLOC VI were developed to solve large p-median problems. No systematic comparison has been made between the programs and the algorithms for large problems, because of the cost, but a few running times and space requirements will be reported here. The largest problem yet solved with ALLOC V was a 521-node, 100-candidate, 50-center problem. The data base was a partial, rectangular distance matrix, and the program required 320 K bytes of core storage. The Teitz and Bart algorithm used 3 minutes, 39 seconds of computer time under the G compiler. The algorithm required 2 minutes, 54 seconds of time when the number of centers was reduced from 50 to 25. A third problem reduced core storage needs from 320 K bytes to 190 K bytes by reducing the number of candidates from 100 to 25. The number of centers in this problem and the time needed to solve it were not preserved for comparison.

An earlier version of ALLOC VI solved a problem with 2990 nodes, 180 candidates, and 102 centers. The analysis required that 23 of the centers be constrained to specific candidates. The Teitz and Bart algorithm required 1 minute, 57 seconds for the problem, and the program required 396 K bytes of core storage. Approximately 25 seconds of this



time was needed just to read the data base for the problem. The trade-off algorithm required 1 minute, 24 seconds for this problem, again including the time needed to read the data. The current version of the program, reported in Table II-3, would require from five to seven percent more time than the earlier version, but it would need only 305 K bytes of storage. The Teitz and Bart algorithm in the current version solved a 2446-node, 418-candidate, 189-center problem in 5 minutes, 7 seconds, including input time. The analysis constrained the locations of 50 of the centers. The program required 301 K bytes of storage.

The add algorithm and the trade-off algorithm in ALLOC VI have never been timed in a systematic way. Because of the way that they were programmed, however, the add algorithm should require about the same time as the Hillsman-Rushton algorithm and the trade-off algorithm should be comparable in time to the Teitz and Bart. These expected times may vary on individual problems.

#### Choosing an Algorithm

The following section discusses factors to consider in choosing an algorithm to solve a problem. Following this discussion, another section considers the possibility of using one algorithm to improve the solution obtained from another.

##### Using a Single Algorithm to Solve a Problem

In a few cases, the choice of an algorithm to solve a particular problem is clear. When a problem contains only one center, all algorithms except the trade-off algorithm will find the same, optimum solution. In ALLOC V, the Maranzana algorithm is preferred because of its speed. The Maranzana algorithm also seems to do as well as the other algorithms in solving problems with only two centers (Rosing et al., 1979b). The algorithm should be run several times on such problems, but this will still take less time than running the other algorithms. For larger numbers of centers the robustness of the Maranzana algorithm--its ability to find good solutions--diminishes rapidly, and other algorithms should be considered. The Maranzana algorithm is not available in ALLOC VI, and the first phase of the Hillsman-Rushton algorithm is the fastest way to locate a single center in that program. The two phases of the Hillsman-Rushton algorithm may be run as independent algorithms in either of the programs, if desired.



When a p-median data base contains "dummy" nodes, or when a problem definition includes maximum distance constraints, the Teitz and Bart algorithm and the first phase of the Hillsman-Rushton algorithm are generally more effective than the Maranzana algorithm and the second phase of the Hillsman-Rushton. The last two algorithms relocate centers within their service areas, and they thus require that the service area of each center be spatially accurate. That is, each node must be assigned to receive service from its nearest center. Neither program meets this requirement for "dummy" nodes. ALLOC V assigns "dummy" nodes to centers arbitrarily. ALLOC VI, because of its method of storing data, is able to assign "dummy" nodes to nearby centers, but it still may not assign all of them to the nearest. ALLOC V also assigns nodes arbitrarily when they cannot be served within a maximum distance constraint, and ALLOC VI assigns them to a "dummy" center. The above procedures violate spatial accuracy for these nodes as well as for the "dummy" nodes. In either case, the Maranzana and second phase of the Hillsman-Rushton algorithm will be unreliable and should be avoided.

The Teitz and Bart algorithm and the first phase of the Hillsman-Rushton algorithm do not require this form of spatial accuracy for their calculations, and they may be used with more confidence than the other two algorithms in these cases. The complete Hillsman-Rushton algorithm is less likely to be affected by a maximum distance constraint than is the second phase alone, because the first phase often ends with all nodes meeting the constraint. The second phase will not violate the maximum distance constraint once the constraint has been met, and it thus will not encounter the spatial inaccuracies that would cause it to be unreliable.

The Teitz and Bart algorithm is relatively unaffected by loose or moderately tight distance constraints. That is, it tends to move centers in the same way that it does when the problem has no distance constraint at all. The add algorithm and the first phase of the Hillsman-Rushton algorithm are much more sensitive to maximum distance constraints. These algorithms will move centers to meet the maximum distance constraint first, and minimizing aggregate distance is of much less importance until the constraint has been met. Most work with the trade-off algorithm has been on problems with maximum distance constraints, and the algorithm seems to have been robust on these problems.

Location constraints in a problem definition will probably reduce the robustness of all of the algorithms to some degree. The effect of these constraints on the



Table II-4

Comparative Efficiency of Solutions to  
Three Test Problems

Test Problem		Random Start	Maranzana	Teitz and Bart	Hillsman-Rushton
49 nodes 5 centers 75 runs	mean	.6346	.9080	1.000	.9952
	st. dev.	.0802	.0475	.000	.0073
	high	.7709	.9961	1.000	1.000
	low	.3519	.7930	1.000	.9592
49 nodes 10 centers 75 runs	mean	.5510	.8231	1.000	1.000
	st. dev.	.0887	.0826	.000	.000
	high	.8334	.9702	1.000	1.000
	low	.3692	.6342	1.000	1.000
150 nodes 21 centers 25 runs	mean	.5108	.8661	.9962	.9834
	st. dev.	.0734	.0389	.0035	.0100
	high	.7132	.9213	1.000	.9971
	low	.4039	.7704	.9910	.9617

Source: Compiled by author. Efficiency of a solution = lowest average distance found for a test problem/average distance for the solution (after Jarvinen et al., 1972). The lowest average distance found for the two 49-node test problems is known to be the optimum.



Table II-5

## Frequency of Efficiencies in Three Test Problems

Efficiency	49 nodes 5 centers			49 nodes 10 centers			150 nodes 21 centers		
	Random	Maranzana	Teitz and Bart Hillsman- Rushton	Random	Maranzana	Teitz and Bart Hillsman- Rushton	Random	Maranzana	Teitz and Bart Hillsman- Rushton
.34 - .40	1								
.40 - .45				2					
.45 - .50	3			10					
.50 - .55	7			7			5		
.55 - .60	11			17			6		
.60 - .65	26			19			9		
.65 - .70	6			15	2		2		
.70 - .75	15			2	6		1		
.75 - .80	6			1	5		1		
.80 - .85		1			12				
.85 - .90		4		2	19			2	
.90 - .95		27			16			5	
.95 - .9999		27			13			14	
1.000		16			13			4	
			75	36	2				15
				39		75	75		10
									25

Source: Compiled by author. Efficiency is defined as in Table II-4.



algorithms needs further research, but the following effects are likely. Prohibiting centers from locating at certain nodes--that is, reducing the number of candidates --probably will have little effect on the Teitz and Bart and Hillsman-Rushton algorithms. Of course, excluding a candidate that should be in the optimum solution necessarily requires the algorithm to find a worse one, but these two algorithms should remain fairly robust when working with the remaining candidates. Unless the ratio of centers to candidates is relatively high, excluding candidates may have little effect on the Maranzana algorithm. The effect of excluding candidates on the add and trade-off algorithms is harder to predict, but it is probably greater than for the other algorithms. Excluding a candidate that the add algorithm would otherwise choose early in its operation may lead to a quite different pattern of centers. The same may be said for excluding very suitable candidates for the trade-off algorithm. The use of a partial, rectangular distance matrix would have an effect similar to the use of this type of location constraint.

Constraining one or more centers to remain at certain nodes will drastically reduce the robustness of the Maranzana algorithm, and it will probably cause a noticeable reduction in the robustness of the other algorithms as well. Increasing the number of immovable centers in a problem decreases the flexibility with which the algorithms can move the other centers around, and this reduces their robustness. An even distribution of immobile centers will probably degrade robustness more than an uneven one, although more research needs to be done to confirm this hypothesis. On the basis of its overall performance, the Teitz and Bart algorithm will probably be more robust than the others in solving problems with immovable centers. It should probably be run several times on such problems however, even small ones.

The choice of an algorithm is less clear in other situations. Tables II-4 and II-5 present results from running the Maranzana, Teitz and Bart, and Hillsman-Rushton algorithms in ALLOC V on the three test problems discussed earlier. In this case, however, twenty-five randomly generated starting solutions were used in the 150-node, 21-center problem. As before, no location or maximum distance constraints were used on the problems. Table II-4 summarizes the mean and extreme values of aggregate distance for the random starting solutions and for the solutions obtained by the three algorithms on the problems. Table II-5 presents the frequency with which the algorithms found different values of aggregate distance. The values in these tables may be used with the execution times from Table II-3 when choosing an algorithm.



On a small problem, such as the two 49-node test problems, the Teitz and Bart and Hillsman-Rushton algorithms require very little computer time. The Teitz and Bart algorithm is more likely to find the optimum solution to a problem of this size,<sup>5</sup> and it seems almost certain to do so at least once if used with several starting solutions to the problem. By the same token however, the Hillsman-Rushton algorithm is more likely to find several good solutions to a particular problem of this size. The ability to obtain a range of good alternative solutions to a problem is one argument made for using heuristic algorithms. By this standard, the Teitz and Bart algorithm is a poor heuristic because it is too consistent. The Marnazana algorithm may not be consistent enough, however.

For larger problems, such as the 150-node, 21-center problem, both the Teitz and Bart and Hillsman-Rushton algorithms provide a range of solutions, and those from the Teitz and Bart algorithm tend to be slightly better on the average. The Teitz and Bart algorithm requires an average of nearly three times as much computer time to find a solution, however, and it is an expensive method of generating a range of good alternative location patterns for centers. As mentioned in the discussion of execution times, one run of the Teitz and Bart algorithm may require several minutes of computer time for larger problems. The Hillsman-Rushton algorithm would probably require from one-third to two-thirds as much time, depending upon the problem. The absolute difference in the times required by the two algorithms will increase for larger problems. As it increases, the Hillsman-Rushton algorithm becomes an ever more attractive choice.

#### Using a Second Algorithm to Improve the Results of the First

Rushton and Church and ReVelle (1976) have discussed the possibility of using the p-median solution found by one heuristic algorithm as the starting solution for a second algorithm. The hope is that the second algorithm can improve upon the solution from the first. The ALLOC programs permit any problem definition, except one involving maximum distance constraints, to use this "piggybacking" approach automatically. The algorithms in a program may be "piggybacked" in any combination, and the two phases of the Hillsman-Rushton algorithm may also be requested as separate algorithms and "piggybacked" with the others. The Hillsman-Rushton algorithm is itself a "piggybacking" of these two phases, which alternate automatically until neither one can

---

<sup>5</sup>Rosing et al., (1979b) report that the best solutions shown in the tables for these two problems are in fact the optimum ones.



improve a solution. The more general "piggybacking" option does not repeat the first algorithm after the second has finished, however.

Table II-6 lists the possible "piggyback" combinations for the algorithms in the ALLOC programs, and it indicates the possibility that the second will improve the solution from the first. Obviously, a second algorithm cannot improve the solution from the first if that solution is the optimum, and it may fail to improve many good solutions. Nor can an algorithm improve upon its own solution, unless it has been stopped prematurely by limits on computer time or pages of output. Thus, neither phase of the Hillsman-Rushton algorithm can improve a solution from the complete algorithm. The Hillsman-Rushton algorithm can improve a solution obtained by its first phase, but only through the action of its second phase. In some cases, an algorithm has been observed to improve the solution from another. These observed improvements are noted in the table, as a guide to selecting algorithm combinations.

The combinations labeled "impossible" or simply "possible" require further explanation. When the Teitz and Bart algorithm stops, it has determined that the aggregate distance for the problem cannot be reduced by relocating any one of the centers to any of the other candidate nodes. When each phase of the Hillsman-Rushton algorithm begins, it examines only a subset of the possible moves that the Teitz and Bart algorithm has considered and rejected. The first phase will only try to relocate the most expendable center, while the second phase will try to move each center within its service area but not beyond. Thus, neither phase alone or in combination can improve a solution from the Teitz and Bart algorithm. Similarly, the Maranzana algorithm cannot improve a solution from the Teitz and Bart algorithm. The Maranzana algorithm can move a center only if the center is not optimally located within its service area, and the Teitz and Bart algorithm will not stop unless each center is optimally located within its service area.

Similar reasoning shows that the Teitz and Bart algorithm can improve a solution obtained by any of the other algorithms, although the robustness of the Hillsman-Rushton makes such improvements unlikely in its case. The combinations labeled "possible" in the table, without further comment, are possible under these types of arguments. The possible improvement was never observed in the development of the algorithms and, in some cases, the combination was never tried.

The add algorithm, contained only in ALLOC VI, is listed as a first algorithm, because it can be used to



Table II-6

## Possibility of Improvement in Different Algorithm Combinations

Source of Starting Solution	Maranzana	Teitz and Bart	Hillsman-Rushton Phase 1	Hillsman-Rushton Phase 2	Hillsman-Rushton Complete	Trade-off
Maranzana	impossible	possible; observed frequently	possible; observed frequently	possible; observed	possible; observed frequently	possible
48 Teitz and Bart	impossible	impossible	impossible	impossible	impossible	impossible
Hillsman-Rushton Phase 1	possible; observed frequently	possible	impossible	possible; observed frequently	possible; observed frequently	possible
Hillsman-Rushton Phase 2	impossible	possible	possible; observed	impossible	possible	possible
Hillsman-Rushton Complete	impossible	possible	impossible	impossible	impossible	possible
Add	possible	possible	possible	possible	possible	possible
Trade-off	possible	possible	possible	possible	possible	impossible

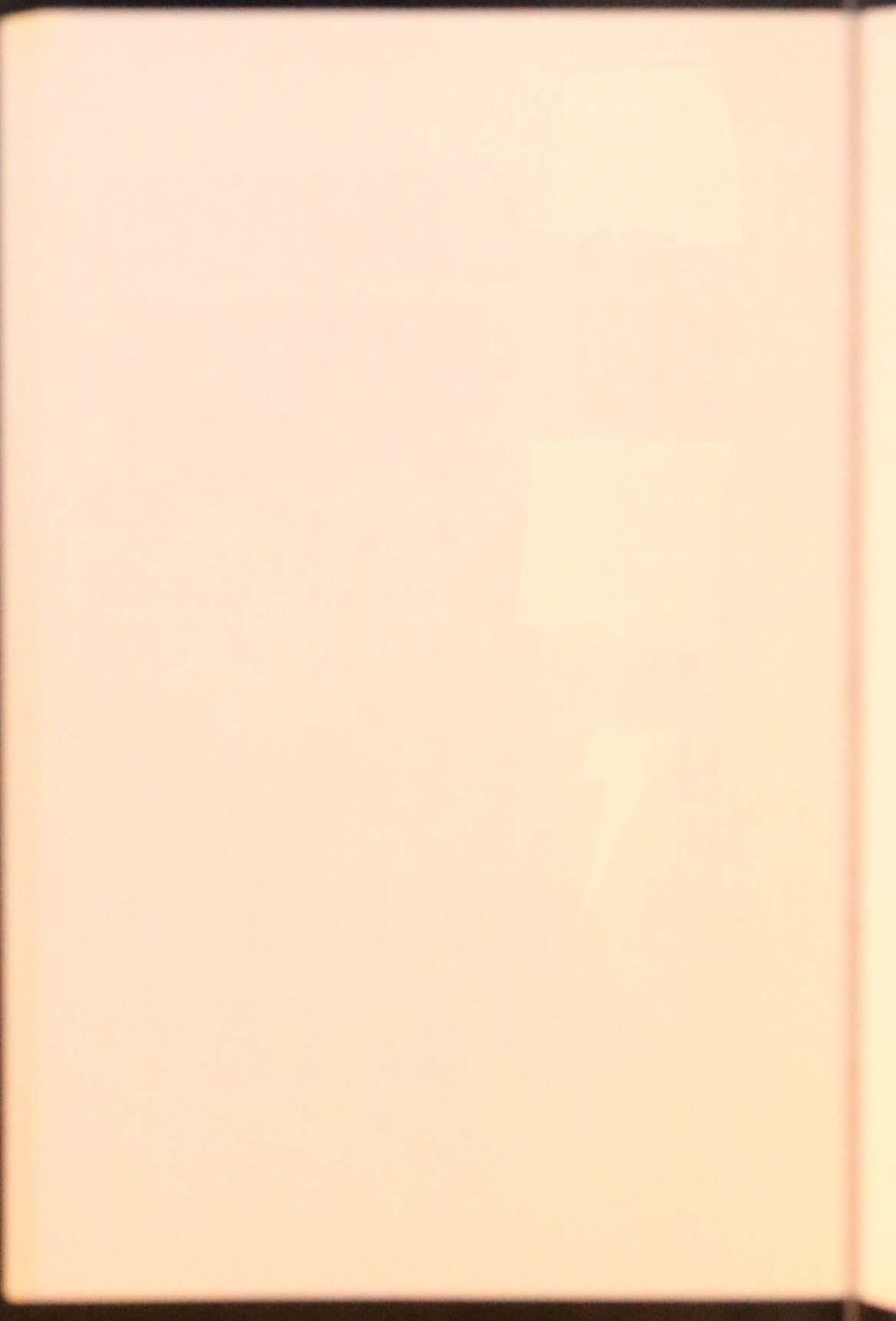
Source: Compiled by author.



produce a solution for use by a second algorithm. The add algorithm is not listed as a second algorithm, because the type of comparison in the table requires the same number of centers at the end of the second algorithm as at the end of the first. Obviously, adding a new center to any solution will reduce the aggregate distance for the problem.

The possible improvements noted in the table for the trade-off algorithm reflect improvements in aggregate distance only. Since the algorithm moves centers only if the movements will reduce aggregate distance, it may make a few moves if it is started with the solution from the more robust algorithms. It cannot make any moves if started with a solution from the Teitz and Bart algorithm, because it considers the same changes. The trade-off algorithm considers the suitability of each candidate, and this may prevent it from locating each center optimally within its service area. Accordingly, any of the other algorithms may be able to move a center and reduce aggregate distance. Such moves may make aggregate suitability worse. The note that the trade-off algorithm cannot improve its own solution assumes that the algorithm uses the same parameters in the trade-off function in both cases. ALLOC VI does not permit the algorithm to be "piggybacked" automatically with different sets of parameters.







## CHAPTER III

### RUNNING ALLOC V

This chapter contains directions for punching the data base cards and problem definition cards needed to run ALLOC V. Chapter I has described the features of the program and the type of data needed to run it. The directions that follow assume familiarity with the earlier descriptions. A few features of the program require additional explanation, and the directions provide examples when needed.

The chapter contains three parts. The first gives directions for punching the p-median data base, and the second gives directions for punching cards to define and solve problems. The third part lists conditions that will stop the program or cause error messages to be printed.

For convenience the directions assume that all data are to be read from punched cards, but only the control card actually must be a punched card. The control card indicates to the program whether the remaining data are to be read from cards or from other storage media.

With the exception of the three variable format cards in the data base, all input data for ALLOC V are to be punched as integer variables and are to be right-justified in their allotted fields. The three variable formats, when used, should be punched left-justified for convenience.

Before the directions are given, two program features need to be mentioned. They differ enough from the features in ALLOC VI that they have not been mentioned in the earlier, more general discussion of program features. First, ALLOC V has a feature that will automatically check the symmetry of an unweighted, square distance matrix. This feature is particularly useful if a symmetrical matrix is to be read from punched cards. It can stop the program if it finds the matrix to be asymmetrical, as might occur if a card gets out of order or is read improperly by the card reader. The symmetry check is optional, since it would be of no use when the matrix should be asymmetrical.

The second feature involves the way that ALLOC V prints information about a solution. When the program prints information about the list of nodes in the order of that list, it does so at the rate of three nodes per printed line. When the program groups the nodes in each center's service area before printing them, each line of printed output will contain information for only one node. This requires roughly three times as much paper as printing this information in the order of the list of nodes. Printing the information in the order of the list of nodes is



considered standard, but the two forms of output are optional, and a problem definition may request either, neither, or both. One form or the other must be requested in order to have the service area population and other information about each center printed.

### Punching the p-Median Data Base

The p-median data base consists of a control card and three decks of cards. The three decks of cards contain: the list of nodes and the distance matrix; the node populations; and, if the distance matrix is not square, the ID numbers of the columns of the matrix that represent candidate nodes. Figure III-1 illustrates a sample input deck for ALLOC V.

- 1.0 CONTROL CARD (required).
- 1.1 In columns 1-5, punch the number of nodes in the list of nodes (number of rows in the distance matrix). This number will be referred to as N.
- 1.2 If the distance matrix is square, skip to 1.3. Otherwise, in columns 6-10, punch the number of candidate nodes (columns of the distance matrix). This number will be referred to as NS.
- 1.3 If the nodes are to have unequal populations, skip to 1.4. Otherwise, in columns 11-15, punch the population to be given to each of the N nodes in the data base.
- 1.4 If you want the program to print the unweighted distance matrix, punch 1 in column 20. Otherwise, leave column 20 blank.
- 1.5 This field controls the symmetry check feature. If you are not using a square distance matrix, or if you do not want your square matrix checked for symmetry, skip to 1.6.

During the symmetry check, the program looks for and prints every symmetry error in the distance matrix. To make the program attempt to correct any symmetry errors which it finds, and then solve p-median problems using the "corrected" data base, punch 1 in column 25. If the program finds a symmetry error, it will replace the distance below the matrix diagonal with the distance above the diagonal. If you have directed the program to print the unweighted distance matrix (1.4), it will print the "corrected" version, not the original.



If you would prefer to have the program stop when it finds symmetry errors, instead of trying to correct the errors in this manner, punch -1 in columns 24-25. The program will print all errors in the matrix before stopping, and it will not print a complete unweighted distance matrix afterward.

- 1.6 Column 30 controls the program's ability to recover the weighted distance matrix after it solves a problem with maximum distance constraints. If this run of the program does not contain any problems with maximum distance constraints, or if it contains only one such problem and that problem is the last one to be solved on the run, skip to 1.7. Otherwise, punch 1 in column 30.

If you are running this program as part of the Geography Program Library at The University of Iowa Computer Center, skip to 2.0. Otherwise, you must supply job-control language (JCL) statements to define enough scratch space to store NxN (or NxNS if your matrix is not square) unformatted 4-byte integers on unit 1. The program writes the integers onto the unit N at a time (NS at a time if the distance matrix is not square) and rereads them into the weighted distance matrix each time it completes a problem with maximum distance constraints. If ALLOC V were run from a job library load module at The University of Iowa Computer Center, the necessary JCL would be

```
//GO.FT01F001 DD UNIT=2314,SPACE=(CYL,(1,1),RLSE),
DCB=(RECFM=VS)
```

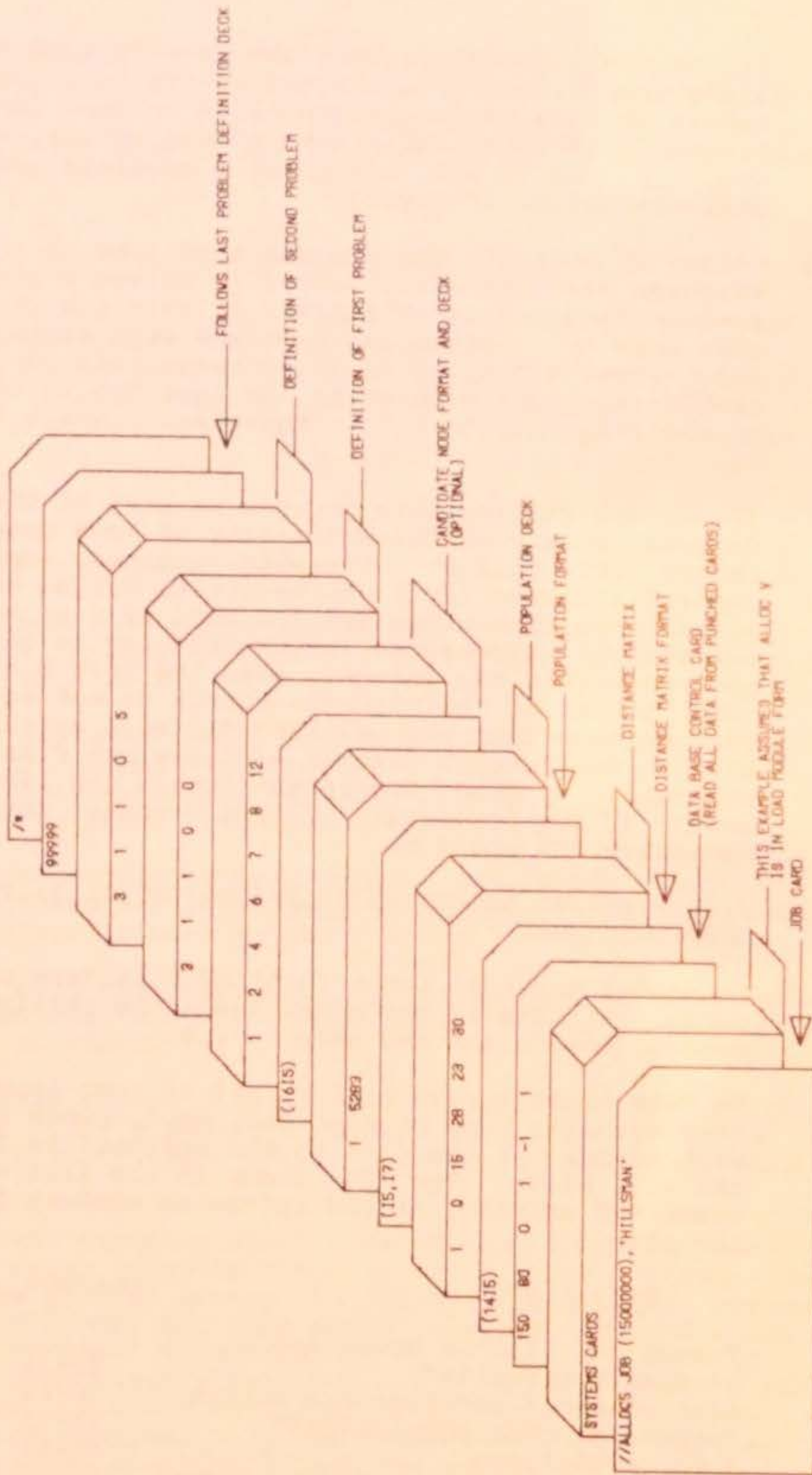
- 1.7 In normal use, all input to ALLOC V is from punched cards. If this is the case, leave the remainder of the control card blank and skip to 2.0.

To have ALLOC V read part or all of your input data from magnetic tape or disk data sets, punch the input unit number for the data in the appropriate field from the list below. The data items in the list will be described shortly, at the reference numbers given in the list.

<u>Data Item</u>	Columns	Reference
Format for list of nodes and distance matrix*	31-35	2.1
List of nodes and distance matrix	36-40	2.2
Format for node populations*	41-45	3.1
Node populations	46-50	3.2
Format for list of candidate nodes*	51-55	4.1
List of candidate nodes	56-60	4.2
Problem definitions*	61-65	5.0- 9.0



FIGURE III-1 Sample Input Deck for ALLOC V





The starred data items must be standard, 80-column card images, but the remaining items may have longer or shorter card or record lengths. All problem definition data for a run of the program must be read from a single input unit during the run. If a data item is to be read from cards, leave the field for that item blank. If your nodes have equal populations (1.3), the program will ignore fields 41-45 and 46-50.

## 2.0 LIST OF NODES AND DISTANCE MATRIX (required).

- 2.1 On the next card, punch a Fortran format to read one node ID number and the row of the distance matrix which corresponds to that node. The format must specify integer fields.

As an example, assume that a distance matrix for the 99 counties in Iowa has been punched one row at a time, with the county ID number punched at the start of each row, and with each ID number and distance punched in a field of five columns. Each ID number and row of the matrix might require 6 cards of 15 numbers each and one card with only 10 numbers. This ID number and row could be read using a format of (15I5) or a format of (6(15I5/),10I5). If your matrix has been prepared by SPA (Ostresh, 1973), your format is (14I5). If your matrix has been prepared by DISTANCE (Chapter I), your format is also (14I5), and you should use the format card punched by that program. If you are using a square distance matrix, skip to 2.2.

The format for a row of the distance matrix must not read more than NS distances (1.2) from the row. If your cards for one row contain more than NS distances (for example, if you were reading from the cards of a square matrix but not using all of the columns), your format must skip distances to places that are not to be candidates.

- 2.2 The remainder of this deck contains the node ID numbers and distances to be read with the format. The ID number for each row of the matrix must be punched in a field preceding the distances for the row. If your matrix has been prepared by SPA or by DISTANCE, it meets all requirements of ALLOC V and may be used without changes.
- 2.9 If your nodes have equal populations (1.3), or if the distance matrix contains coefficients for a problem other than the p-median, skip to 3.9.



3.0 NODE POPULATIONS (optional).

- 3.1 On the next card, punch a Fortran format to read one card of the population deck. The format must specify integer fields.

For example, the population deck for the 99 counties in Iowa might be punched one population to a card, with the ID number punched in columns 1-5 and the population in columns 6-15. The format for this deck would be (I5,I10). If the populations were punched five counties to a card, with five columns for the ID number and ten columns for the population, the format for the deck would be (5(I5,I10)).

- 3.2 The rest of the deck contains the node ID numbers and populations, punched together for each node, with the ID number preceding the node population. You may punch more than one ID and population per card, as long as you do not split a node ID and population between two cards, and as long as each card (except possibly the last in the deck) has the same number of node ID numbers and populations. Node ID numbers and populations need not be in the same order as the ID numbers in the list of nodes (2.2). The program will place them in the proper order after reading them.

- 3.9 If you are using a square distance matrix, skip to 4.9.

4.0 LIST OF CANDIDATE NODES (optional).

- 4.1 On the next card, punch a Fortran format to read one card of the candidate node deck. The format must specify integer fields.

- 4.2 The rest of the deck contains the ID numbers of the candidate nodes, punched to be read by the format. You may punch more than one ID number per card, as long as each card (except possibly the last) has the same number of candidate ID numbers. The candidate ID numbers correspond to columns of the distance matrix, and they must be punched in the same order as column elements were punched to be read from the distance matrix (2.1-2.2). Thus, if the first distance in each row of the matrix was measured from a node to a candidate with ID number 25, and the second distance in each row was the distance to candidate number 30, the first two candidate ID numbers in the candidate list must be 25 and 30, in that order.

- 4.9 This completes the p-median data base for a series of problems.



## Punching the Problem Definition Cards

Each problem definition consists of one control card and three decks of cards. The three decks contain: location constraints; a list of centers to serve as a starting solution; and maximum distance constraints. Only the control card and the list of centers are necessary to define a problem. Each deck begins on a new card. Figure III-2 illustrates a sample problem definition.

As many as one hundred problems may be defined on each run of ALLOC V, by adding as many problem definitions to the input deck as desired.

The directions that follow (5.0-8.9) assume that none of the problem definitions involve the automatic construction of a hierarchy. Special instructions for constructing a hierarchy appear in section 9.0 of the directions.

5.0 PROBLEM DEFINITION CONTROL CARD (required).

5.1 In columns 1-5, punch the number of centers in the problem.

5.2 Column 10 controls the algorithm to be used in solving the problem, and its value appears as MALG on the printed output. If you want the program to compute and print information about the starting solution, but do not want an algorithm to try to improve it, skip to 5.3. Otherwise, in column 10, punch the number from the following list that corresponds to the algorithm you wish to use.

- 1 Maranzana algorithm
- 2 Teitz and Bart algorithm
- 3 Hillsman-Rushton algorithm, phase 1 only
- 4 Hillsman-Rushton algorithm, phase 2 only
- 5 Hillsman-Rushton algorithm, complete

5.3 The next field controls the use of location constraints, and its value appears as ICON on the printed output. If you do not want to use any location constraints on this problem, skip to 5.4.

If you did not use any location constraints on the problem immediately preceding this one, or if you did but want to change them for this problem, punch 1 in column 15 and skip to 5.4.

If you used location constraints on the problem immediately preceding this one, and want to use the



same set of constraints on this problem, punch -1 in columns 14-15. This method may be used to repeat one set of constraints in as many consecutive problems as desired.

- 5.4 The next field controls the use of a second algorithm on the problem, and its value appears as MALG2 on the printed output. If you do not want to use a second algorithm to try to improve the solution obtained by the first, skip to 5.5. In addition, if the problem definition will include maximum distance constraints, the program will not use a second algorithm on the problem, and you should also skip to 5.5.

In column 20, punch the number corresponding to the second algorithm that you wish to use on the problem, using the list at 5.2. The second algorithm will use as its starting solution the solution obtained by the first algorithm.

- 5.5 The next field controls the use of maximum distance constraints, and its value appears as FRIT on the printed output. If you do not want to use any maximum distance constraints on this problem, skip to 5.6.

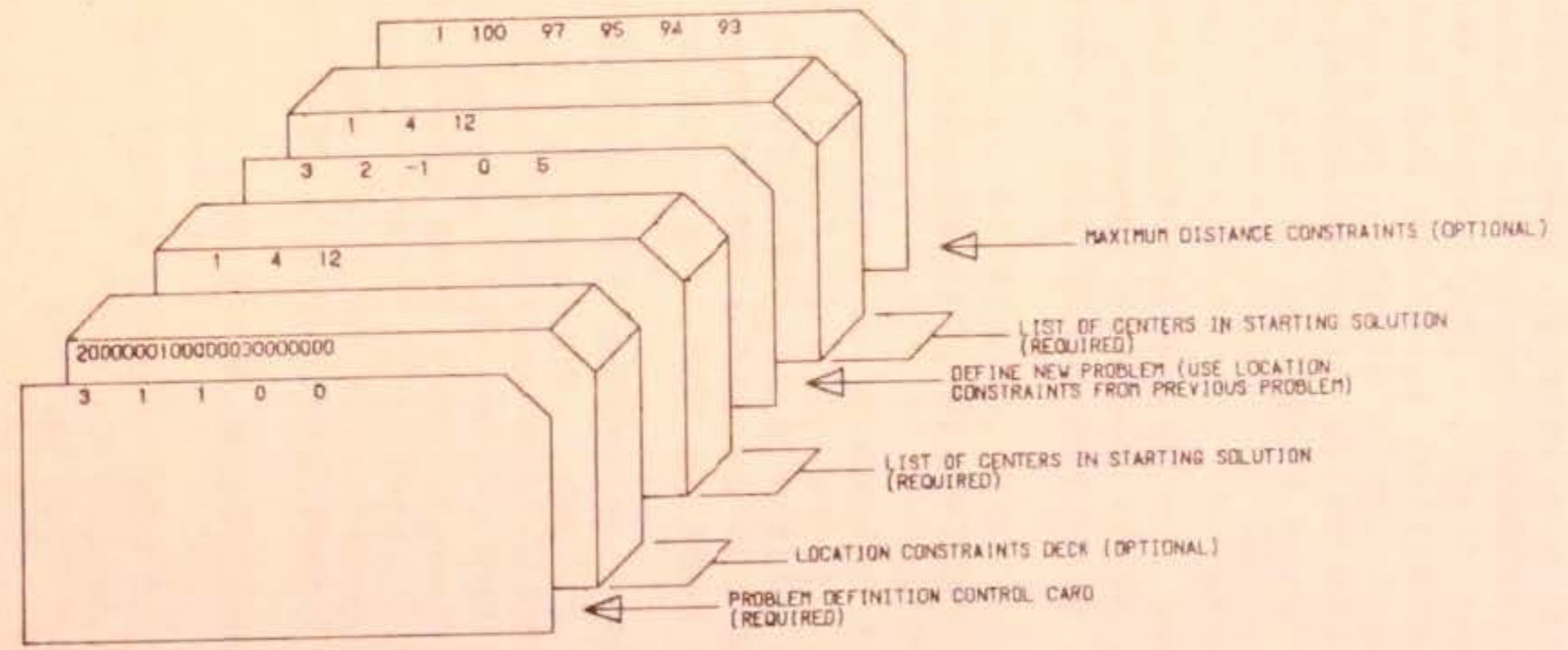
If you want to impose a list of specific maximum distance constraints on this problem, punch in columns 24-25 the number of constraints to be imposed. No more than 30 maximum distance constraints may be imposed on one problem.

If you want to solve the problem, find the longest distance from any node to its nearest center, and impose a maximum distance constraint on the problem just less than this longest distance, punch the change in the longest distance in columns 21-25. For example, if you want the longest distance reduced by 1 distance unit, punch -1 in columns 24-25. To reduce the longest distance by 2 units, punch -2, etc. After computing and imposing this constraint, the algorithm will try to meet it. Only one distance constraint may be computed in this manner for a problem.

- 5.6 Column 30, with column 35 below (5.7), controls the way that information is printed about the list of nodes and each node's nearest center. If you want this information printed in the order of the list of nodes, leave column 30 blank. Otherwise punch a 1 in column 30 to suppress this form of the printed output. The value punched in column 30 appears as NMAP on the printed output.



FIGURE III-2 Sample Problem Definition for ALLOC V





- 5.7 If you want information about each node and its nearest center grouped and printed in the form of a service area for each center, punch a 1 in column 35 to request this form of printed output. Otherwise leave column 35 blank. The value punched in column 35 appears as MMAP on the printed output.
- 5.8 If you have a value of aggregate (not average) distance with which you wish to compare the aggregate distance for this problem, punch the value you wish to compare in columns 41-50. The program will make the comparison for the starting solution and for the solution at the end of each cycle of an algorithm. The comparison is made as the ratio of your aggregate distance value to the aggregate distance of the current solution to the problem.
- 5.9 If you are not using any location constraints on this problem, or if you are repeating the location constraints from the preceding problem (5.3), skip to 7.0.

#### 6.0 LOCATION CONSTRAINTS (optional).

Each card column of a location constraints deck corresponds to one node in the list of nodes. If the list of nodes contains more than 80 nodes, then the first column of the second card of the deck corresponds to the 81st node in the list, and so forth.

A location constraint is imposed by punching 1 or 2 in a node's column. Punching 1 in the column will prevent an algorithm from moving a center to the node. Punching 2 in the column will prevent an algorithm from moving a center away from the node. Punching 0 in the column, or leaving it blank, does not impose any location constraint on the node. For example, the location constraints for a 26-node network might be:

```
00000100000210000200110000
```

This set of constraints would prevent an algorithm from moving a center to nodes 6, 13, 21, and 22, and would prevent it from moving any center located at nodes 12 and 18. The other twenty nodes would have no location constraints, and the algorithm would be free to move centers to or from any of these nodes at any time.

The program will print the location constraints at the beginning of the problem.



If your distance matrix is square, skip to 7.0. Otherwise, you should note that location constraints must be supplied for each node in the list of nodes, even though many of these may not be candidate nodes. If a node is not a candidate, the program will ignore any location constraints punched for it. When the program prints the location constraints at the beginning of a problem, it will print only the constraints for the candidate nodes, in the order that those nodes appeared in the list of candidates (4.0).

7.0 STARTING SOLUTION (required).

- 7.1 In fields of five columns, punch the ID numbers of your initial center locations. If the problem has more than 16 centers, punch the 17th ID number in the first five columns of a second card, etc. The ID numbers may be punched in any order.

If your distance matrix is square, skip to 7.9. Otherwise, note that the ID numbers of your initial center locations must appear somewhere in the list of candidate nodes (4.0).

- 7.9 If you are not imposing a set of specific distance constraints on this problem (5.5), skip to 8.8.

8.0 MAXIMUM DISTANCE CONSTRAINTS (optional).

- 8.1 If you want to impose the first maximum distance constraint before the algorithm tries to solve the problem, leave the first five columns of this deck blank. Otherwise, punch 1 in column 5 to have the algorithm solve the problem without any maximum distance constraint before it imposes the first one.
- 8.2 In columns 6-10, punch the first maximum distance constraint to be imposed on this problem. In succeeding fields of five columns, punch any additional maximum distance constraints, in order of decreasing length. If more than fifteen constraints are to be imposed, punch the sixteenth constraint in columns 1-5 of a second card and continue the rest of the constraints on that card.
- 8.8 This completes the definition of a problem. The control card to define an additional problem would be placed immediately behind the last card for this problem.



8.9 Following the last card for the last problem definition, place a card containing 99999 punched in columns 1-5.

9.0 AUTOMATIC HIERARCHY CONSTRUCTION (optional).

Construction of a spatial hierarchy is requested by defining the lowest level of the hierarchy as though it were a typical p-median problem, and by punching the total number of levels in the hierarchy in columns 36-40 of the problem definition for that level. The value punched in these columns appears as MUP on the printed output. Each of the remaining levels is defined as though it were a separate, typical p-median problem, except that no initial list of centers is supplied as a starting solution for the upper levels. For the upper levels, the program selects a starting solution from the first centers in the list of centers located at the preceding level. The number of levels in the hierarchy need not be punched in the definitions of the higher levels.

Although no maximum distance constraints may be imposed on any level of hierarchy, location constraints may be used if desired and may be repeated from level to level. At the completion of the highest level, however, the program removes the location constraint from every candidate in the data base. If the first problem defined after the construction of a hierarchy is to have location constraints, it cannot automatically repeat the location constraints imposed on the hierarchy.

The following example will construct a three-level hierarchy of ten, six, and two centers, using the Teitz and Bart algorithm for the bottom two levels and the Maranzana on the highest. The node with ID number 49 is to be forced into the lower levels of the hierarchy, and out of the third.

10      2      1                      3

[A deck of location constraints goes here, with a 2 corresponding to the node with ID number 49.]

1      8      26      49      7      11      79      103      52      61

[ID number 49 could be punched in any of the first six fields to be started in the second level of the hierarchy.]



6 2 -1

[This control card for the second level will locate 6 centers and repeat the location constraints used on the first level, to force node 49 to have a center.]

2 1 1

[This control card defines the third level problem. A new deck of location constraints would go here, with a 1 for the node having ID number 49, to force it out of solution. Because ID number 49 was not punched in the first two fields for the lowest level, it will not start in the third level.]

#### Error Conditions

The following conditions will produce an error message and stop the program.

1. Symmetry errors in the distance matrix, if this form of symmetry check is requested (1.5).
2. Appearance of the same ID number twice in the populations deck (3.0) or in the candidate node deck (4.0).
3. Failure to find an ID number from the populations deck (3.0) or candidate node deck (4.0) in the list of nodes (2.0). This may result from an erroneous ID number in the list of nodes, or from an error in the populations or candidate decks.
4. Failure to find an ID number from the starting solution deck (7.0) in the list of candidate nodes. The most common cause of this error is a misplaced card in the problem definition cards (5.0-8.9).
5. Maximum distance constraints (8.2) that are not in decreasing order.
6. Specification of a distance matrix with more columns than rows (1.1-1.2).
7. Attempt to impose maximum distance constraints on a hierarchy.
8. Values greater than 2 in the location constraints deck (6.0). The most common cause of this error is a misplaced card.



9. Attempt to locate more centers in the upper levels of a hierarchy than in the lower levels.
10. End-of-file while trying to read location constraints, starting solutions, maximum distance constraints, or an upper level of a hierarchy (6.0-9.0).
11. The product of the number of nodes and the largest weighted distance exceeds the largest integer that can be stored.

The following error conditions are less serious. They will produce an error message and the following actions.

1. Request for a symmetry check (1.5) when the distance matrix is not square (1.2). The program ignores the request for the symmetry check.
2. Attempt to define a problem after using maximum distance constraints when no external storage has been provided for recovering the weighted distance matrix (1.6). The program ignores the remaining problem definitions.

#### Program Dimensions

The array dimensions in the listings of ALLOC IV and ALLOC V permit a data base of 150 nodes and 150 candidates, and a problem of 150 centers. The comment cards at the beginning of the programs give directions for changing the dimensions of the program arrays.



FIGURE III-3 Sample Output from ALLOC V Including TALALA  
Distance Matrix (Rosing, Hillsman et al. 1979)

PROGRAM ALLOC V  
WRITTEN BY EDWARD L. HILLSMAN  
DEPARTMENT OF GEOGRAPHY  
THE UNIVERSITY OF IOWA  
IOWA CITY, IOWA 52242  
JUNE 1974-JUNE 1977

SQUARE DISTANCE MATRIX

SYMMETRY CHECK OF DISTANCE MATRIX

NO SYMMETRY ERRORS ENCOUNTERED

ECHO CHECK OF DISTANCE MATRIX

1	0	54	135	149	171	222	84	181	130	131	192	93	166	167	113	133	81	137	173	169	183	40	134
	219	127	163	140	51	178	172	116	175	44	160	87	132	145	100	172	214	180	188	87	75	196	182
	158	95	203																				
2	54	0	143	203	179	254	34	189	145	185	224	101	220	221	167	187	111	117	232	223	130	14	142
	176	181	217	148	59	232	180	124	125	98	110	141	82	153	50	205	222	130	242	95	129	250	132
	166	149	153																				
3	135	143	0	242	36	227	138	162	158	211	197	114	259	260	206	226	136	55	233	249	128	129	85
	84	220	256	121	84	271	37	67	133	173	108	180	90	54	122	217	195	128	291	48	168	289	130
	25	188	151																				
4	149	203	242	0	278	164	233	176	120	67	141	147	140	104	112	118	106	244	114	105	327	189	212
	326	36	17	163	158	73	279	194	322	105	307	62	279	223	249	121	196	327	81	194	74	55	329
	236	94	350																				
5	171	179	36	278	0	263	174	198	194	247	233	150	295	296	242	262	172	91	269	285	149	165	121
	63	256	292	157	120	307	73	103	163	209	129	216	126	90	158	253	231	149	317	84	204	325	151
	61	224	172																				
6	222	254	227	164	263	0	284	65	110	97	30	153	238	239	185	205	160	239	50	82	322	240	178
	311	128	164	106	195	229	264	160	317	178	302	141	274	189	300	50	40	322	237	189	147	211	324
	202	145	345																				
7	84	34	139	233	174	284	0	219	175	215	254	131	250	251	197	217	141	83	262	253	96	44	172
	142	211	247	178	89	262	147	154	91	128	76	171	48	183	16	235	252	96	272	125	159	280	98
	163	179	119																				
8	181	189	162	176	198	65	219	0	63	109	35	88	209	210	156	176	113	174	71	103	257	175	113
	246	140	176	41	130	221	199	95	252	149	237	130	209	124	235	55	33	257	231	124	118	223	259
	137	98	280																				



9	130	145	158	120	194	110	175	63	0	53	80	44	146	147	93	113	50	170	93	91	253	131	109
	242	84	123	43	86	158	195	91	248	86	233	67	205	120	191	60	96	253	168	120	55	167	255
	133	35	276																				
10	131	185	211	67	247	97	215	109	53	0	74	97	147	148	94	114	88	223	87	38	336	171	162
	295	31	67	96	139	132	248	144	301	87	286	44	258	173	231	54	129	306	140	173	56	114	308
	186	76	329																				
11	192	224	197	141	233	30	254	35	60	74	0	123	208	209	155	175	130	202	36	68	292	210	148
	281	105	141	76	165	206	234	130	287	143	272	118	244	159	270	20	55	292	214	159	117	188	294
	172	115	315																				
12	93	101	114	147	152	153	131	88	44	97	123	0	164	165	111	131	41	126	134	135	209	87	65
	198	125	161	47	42	176	151	47	204	78	189	85	161	76	147	104	121	209	135	76	73	194	211
	69	79	232																				
13	166	220	259	140	295	238	250	209	146	147	208	154	0	36	89	33	123	261	194	185	344	206	229
	343	143	132	189	175	67	296	211	339	122	324	103	296	240	266	188	242	344	33	211	91	85	346
	253	111	367																				
14	167	221	260	104	296	239	251	210	147	148	209	165	36	0	90	34	124	262	195	186	365	207	230
	344	132	96	190	176	31	297	212	340	123	325	104	297	241	267	189	243	345	53	212	92	49	347
	254	112	368																				
15	113	167	206	112	242	185	197	156	93	94	155	111	89	90	0	55	70	208	141	132	291	153	176
	290	90	126	136	122	131	243	158	286	69	271	50	243	187	213	135	189	291	111	158	38	119	293
	200	58	314																				
16	133	187	226	118	262	205	217	176	113	114	175	131	33	34	56	0	90	228	151	152	311	173	196
	310	110	110	156	142	45	263	178	306	89	291	70	263	207	233	155	209	311	55	178	58	63	313
	220	78	334																				
17	81	111	136	106	172	150	141	113	50	88	130	41	123	124	70	90	0	138	135	126	221	97	106
	220	84	120	88	52	135	173	88	216	37	201	44	173	117	157	110	146	221	145	88	32	153	223
	130	52	244																				
18	137	117	55	244	91	239	83	174	170	223	209	126	261	262	208	228	138	0	245	261	83	127	97
	129	222	258	133	86	273	92	79	78	175	63	182	35	108	67	229	207	83	283	50	170	291	85
	80	190	106																				
19	178	232	233	114	269	50	262	71	90	47	36	134	194	195	141	161	135	245	0	32	328	218	184
	317	78	114	112	176	179	270	166	323	134	308	91	280	195	278	33	90	328	187	195	103	161	330
	208	123	351																				
20	169	223	249	105	285	82	253	103	91	38	68	135	185	186	132	152	126	261	32	0	344	209	200
	333	69	105	134	177	170	286	182	339	125	324	82	296	211	269	58	122	344	179	211	94	152	346
	224	114	367																				
21	180	130	128	327	149	322	96	257	253	306	292	209	344	145	291	311	221	83	328	344	0	140	180
	66	305	341	216	169	356	91	162	54	224	20	265	48	182	80	312	290	40	366	133	253	374	42
	153	273	23																				
22	40	14	129	189	165	240	44	175	131	171	210	87	206	207	153	173	97	127	218	209	140	0	128
	186	167	203	134	45	218	166	110	135	84	120	127	92	139	60	191	208	140	228	81	115	236	142
	152	135	163																				



23	134	142	85	212	121	178	172	113	109	162	148	65	229	230	176	196	106	97	184	200	190	126	0
	169	190	226	72	83	241	122	18	175	143	160	150	132	47	164	164	146	180	251	47	138	259	182
	60	144	203																				
24	219	176	84	326	63	311	142	246	242	295	281	198	343	344	290	310	220	129	317	333	86	186	169
	0	304	340	205	168	355	47	151	100	257	66	264	94	138	126	301	279	86	365	132	252	373	88
	109	272	109																				
25	127	181	220	36	256	128	211	140	84	31	105	125	143	132	90	110	64	222	78	69	305	167	190
	304	0	36	127	136	101	257	172	300	83	285	40	257	201	227	85	160	305	109	172	52	83	307
	214	72	328																				
26	163	217	256	17	292	164	247	176	120	67	141	161	132	96	126	110	120	258	114	105	341	203	226
	340	36	0	163	172	65	293	208	336	119	321	76	293	237	263	121	196	341	73	208	88	47	343
	250	108	364																				
27	140	148	171	163	157	106	178	41	43	96	75	47	189	190	136	155	88	133	112	134	216	134	72
	205	127	163	0	89	201	158	54	211	125	196	110	168	83	194	96	74	216	211	83	98	210	218
	96	78	239																				
28	51	59	84	158	120	195	89	130	86	139	165	42	175	176	122	142	52	86	175	177	169	45	83
	168	136	172	89	0	187	121	65	164	89	149	96	121	94	105	146	163	169	197	36	84	205	171
	107	104	192																				
29	178	232	271	73	307	229	262	221	158	132	206	176	67	31	101	45	135	273	179	170	356	218	241
	355	101	65	201	187	0	308	223	351	134	336	115	308	252	278	185	254	356	29	223	103	18	358
	265	123	379																				
30	172	160	37	279	73	264	147	199	195	248	234	151	296	297	243	263	173	92	270	286	91	166	122
	47	257	293	158	121	308	0	104	105	210	71	217	99	91	131	254	232	91	318	85	205	326	93
	62	225	114																				
31	116	124	67	194	103	160	154	95	91	144	130	47	211	212	158	178	88	79	166	182	162	110	18
	151	172	208	54	65	223	104	0	157	125	142	132	114	29	146	150	128	162	233	29	120	241	164
	42	126	185																				
32	175	125	133	322	163	317	91	252	248	301	287	204	339	340	286	306	216	78	323	339	54	135	175
	100	303	336	211	164	351	105	157	0	219	34	260	43	186	75	307	285	14	361	128	248	369	56
	158	268	77																				
33	44	98	173	105	209	178	128	149	86	87	148	78	122	123	69	84	37	175	134	125	224	84	143
	257	83	119	125	89	134	210	125	219	0	204	43	176	154	144	128	182	224	144	125	31	152	226
	167	51	247																				
34	160	110	108	307	129	302	76	237	233	286	272	189	324	325	271	291	201	63	308	324	20	120	160
	66	285	321	196	149	336	71	142	34	204	0	245	28	162	60	292	270	20	346	113	233	354	22
	133	253	43																				
35	87	141	180	62	216	141	171	130	67	44	118	85	103	104	50	70	44	182	91	82	265	127	150
	264	40	76	110	96	115	217	132	260	43	245	0	217	161	187	98	163	265	125	132	12	117	267
	174	32	288																				
35	132	82	90	279	126	274	49	209	205	258	244	161	296	297	243	263	173	35	280	296	48	92	132
	94	257	293	164	121	308	97	114	43	176	28	217	0	143	32	264	242	48	318	85	205	326	50
	115	225	71																				



37	145	153	54	223	90	189	183	124	120	173	157	76	240	241	187	207	117	108	195	211	182	139	47
	138	201	237	83	94	252	91	29	186	154	162	161	143	0	175	179	157	162	252	58	149	270	184
	34	155	205																				
38	100	50	122	249	158	300	16	235	191	231	270	147	266	267	213	233	157	67	278	269	80	60	164
	126	227	263	194	105	278	131	146	75	144	60	187	32	175	0	251	268	80	298	117	175	296	82
	147	195	103																				
39	172	205	217	121	253	50	235	55	60	54	20	104	188	189	135	155	110	229	30	58	312	191	168
	301	85	121	96	146	186	254	150	307	128	292	98	264	179	251	0	75	312	194	179	97	168	314
	192	95	335																				
40	214	222	195	196	231	40	252	33	96	129	55	121	242	243	189	209	146	207	93	122	290	208	146
	279	160	196	74	163	254	232	128	285	182	270	163	242	157	268	75	0	290	264	157	151	243	292
	170	131	313																				
41	180	130	128	327	149	322	96	257	253	306	292	209	344	345	291	311	221	83	328	344	40	140	180
	86	305	341	216	169	356	91	162	14	224	20	265	48	182	80	312	290	0	368	133	253	374	42
	153	273	63																				
42	188	242	281	81	317	237	272	231	168	140	214	186	88	60	111	55	145	283	187	178	366	228	251
	365	109	73	211	197	29	318	233	361	144	346	125	318	262	288	194	264	366	0	233	113	26	368
	275	133	389																				
43	87	95	48	194	84	189	125	124	120	173	159	76	211	212	158	178	88	50	195	211	133	81	47
	132	172	208	83	36	223	85	29	128	125	113	132	85	58	117	179	157	133	233	0	120	241	135
	71	140	156																				
44	75	129	168	74	204	147	159	118	55	56	117	73	91	92	38	58	32	170	103	94	253	115	138
	252	52	88	98	84	103	205	120	248	31	233	12	205	149	175	97	151	253	113	120	0	121	255
	162	20	276																				
45	196	250	287	55	325	211	280	223	167	114	188	194	85	49	119	63	153	291	161	152	374	236	259
	373	83	47	210	205	18	326	241	369	152	354	117	326	270	296	168	243	374	26	241	121	0	376
	283	141	397																				
46	182	132	130	329	151	324	98	259	255	308	294	211	346	347	293	313	273	85	330	346	42	142	182
	88	307	343	218	171	358	93	164	56	226	22	267	50	104	82	314	292	42	359	135	255	376	0
	155	275	38																				
47	158	166	25	236	61	202	163	137	133	186	172	89	253	254	200	273	130	80	208	224	153	152	60
	109	214	250	96	107	265	62	42	158	167	133	174	115	34	147	192	170	153	275	71	162	283	155
	0	168	176																				
48	95	149	189	94	224	145	179	98	35	76	115	79	111	112	58	78	52	190	123	114	273	135	144
	272	72	108	78	104	123	225	126	268	51	253	32	225	155	195	95	131	273	133	140	20	141	275
	168	0	296																				
49	203	153	151	350	172	345	119	280	276	329	315	232	367	368	314	354	244	106	351	367	23	163	203
	109	328	364	239	192	379	114	185	77	247	43	268	71	205	103	335	313	63	389	156	276	397	38
	176	296	0																				



NUMBER OF PLACES IS 49	
1	2811
5	1241
9	2981
13	767
17	1337
21	133
25	813
29	1119
33	1070
37	764
41	973
45	1086
49	529
2	592
6	845
10	2767
14	904
18	1905
22	857
26	755
30	1755
34	393
38	739
42	1803
46	1144
3	4027
7	422
11	933
15	734
19	1198
23	1087
27	365
31	1835
35	1607
39	932
43	1736
47	733
4	1536
8	819
12	3356
16	2882
20	546
24	1245
28	2328
32	1205
36	1095
40	1320
44	6740
48	1198

TOTAL WEIGHT IS 69962

ALL 49 NODES ARE ELIGIBLE FOR SELECTION AS CENTERS

WEIGHTED DISTANCE MATRIX STORED ON DATA SET 1 FOR RECOVERY FOLLOWING WORK WITH DISTANCE CONSTRAINTS

69

PROBLEM NUMBER 1

LOCATE 10 CENTERS

MALG ICON MALG2 KRIT NMAP MMAP MUP  
 2 0 0 1 0 0 0  
 EFFICIENCY WILL BE MEASURED AGAINST A VALUE OF 1561823

LIST BY DEMAND

CENTER	WEIGHT	DISTANCE * WEIGHT	AVERAGE DISTANCE	COST IF DROPPED
44	12686	147090	12	518440
34	7878	300247	38	585107
3	9661	232711	24	440327
28	5684	140952	25	125440
1	4260	66248	16	150606
42	8561	340933	40	421988
31	5422	92066	17	221182
8	4282	146105	34	198914
9	2981	0	0	157993
10	8547	306082	36	237628

FOR THE ASSIGNMENT WHICH FOLLOWS:  
 TOTAL WEIGHTED DISTANCE IS 1772434  
 AVERAGE DISTANCE TO NEAREST CENTER IS 25



DESTINATION	CENTER	WEIGHT	DIST.	DESTINATION	CENTER	WEIGHT	DIST.	DESTINATION	CENTER	WEIGHT	DIST.				
11	1	1	2811	0	11	2	1	592	54	11	3	3	4027	0	11
11	4	10	1536	67	11	5	3	1241	36	11	6	8	845	65	11
11	7	34	422	76	11	8	8	819	0	11	9	9	2981	0	11
11	10	10	2767	0	11	11	8	933	35	11	12	28	3356	42	11
11	13	42	767	88	11	14	42	904	60	11	15	44	734	38	11
11	16	42	2882	55	11	17	44	1337	32	11	18	3	1905	55	11
11	19	10	1198	47	11	20	10	546	38	11	21	34	133	20	11
11	22	1	857	40	11	23	31	1087	18	11	24	34	1245	66	11
11	25	10	913	31	11	26	10	755	67	11	27	8	365	41	11
11	28	28	2328	0	11	29	42	1119	29	11	30	3	1755	37	11
11	31	31	1835	0	11	32	34	1205	34	11	33	44	1370	31	11
11	34	34	393	0	11	35	44	1607	12	11	36	34	1095	28	11
11	37	31	764	29	11	38	34	739	60	11	39	10	932	54	11
11	40	8	1320	33	11	41	34	973	20	11	42	42	1803	0	11
11	43	31	1736	29	11	44	44	6740	0	11	45	42	1086	26	11
11	46	34	1144	22	11	47	3	733	25	11	48	44	1198	20	11
11	49	34	529	43	11										

EFFICIENCY STATISTICS

MOST EXPENDABLE CENTER IS 28, WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY 125440 IF DROPPED WITHOUT REPLACEMENT

MAXIMUM DISTANCE TRAVELED IS 88, FROM NODE 13 TO CENTER 42

TOTAL WEIGHTED DISTANCE IS 1772434  
 AVERAGE DISTANCE TO NEAREST CENTER IS 25

RATIO OF REFERENCE TO CURRENT VALUE OF OBJECTIVE FUNCTION IS 0.8811741  
 PERCENT CHANGE IN OBJECTIVE FUNCTION FROM STARTING SOLUTION IS 0.0  
 PERCENT CHANGE IN OBJECTIVE FUNCTION FROM PRECEDING CYCLE IS 0.0

WEIGHTED DISTANCE MATRIX TO BE TRANSFORMED BY 1860240 IN DISTANCE CONSTRAINT CALCULATIONS  
 SOLVE PURE DISTANCE MINIMIZATION PROBLEM BEFORE IMPOSING DISTANCE CONSTRAINTS

TEITZ AND BART ALGORITHM CYCLE 1

OLD CENTER	NEW CENTER														
28	4	1757212	44	34	3	4	1	42	31	8	9	10			
8	6	1755461	44	34	3	4	1	42	31	6	9	10			
6	11	1708551	44	34	3	4	1	42	31	11	9	10			
9	12	1672559	44	34	3	4	1	42	31	11	12	10			
4	13	1660625	44	34	3	13	1	42	31	11	12	10			
13	14	1658575	44	34	3	14	1	42	31	11	12	10			
14	16	1587022	44	34	3	16	1	42	31	11	12	10			
42	45	1561823	44	34	3	16	1	45	31	11	12	10			

70



END OF TETZ AND BART CYCLE 1  
 THERE WERE 8 CHANGES DURING THIS CYCLE

CENTER	WEIGHT	DISTANCE * WEIGHT	AVERAGE DISTANCE	COST IF DROPPED
44	12686	147090	12	535107
34	7878	300247	38	585107
3	9661	232711	24	440327
16	4553	56047	12	220600
1	4260	66248	16	278928
45	6299	186985	30	184450
31	5422	92066	17	169418
11	6047	188383	31	301050
12	9030	246095	27	208068
10	4126	45951	11	188405

FOR THE ASSIGNMENT WHICH FOLLOWS:

TOTAL WEIGHTED DISTANCE IS 1561923  
 AVERAGE DISTANCE TO NEAREST CENTER IS 22

DESTINATION	CENTER	WEIGHT	DIST.	DESTINATION	CENTER	WEIGHT	DIST.	DESTINATION	CENTER	WEIGHT	DIST.
11	1	2811	0	11	2	592	54	11	3	4027	0
11	4	1536	55	11	5	1241	36	11	6	845	30
11	7	422	76	11	8	819	35	11	9	7981	44
11	10	2767	0	11	11	933	0	11	12	3356	0
11	13	767	33	11	14	904	34	11	15	754	38
11	16	2892	0	11	17	1337	32	11	18	1725	55
11	19	1198	36	11	20	546	38	11	21	133	20
11	22	857	40	11	23	1087	18	11	24	1245	66
11	25	813	31	11	26	755	47	11	27	365	47
11	28	2328	42	11	29	1119	18	11	30	1755	37
11	31	1835	0	11	32	1205	34	11	33	1070	31
11	34	393	0	11	35	1607	12	11	36	1095	28
11	37	764	29	11	38	739	60	11	39	932	20
11	40	1320	55	11	41	973	20	11	42	1833	26
11	43	1736	29	11	44	6740	0	11	45	1086	0
11	46	1144	22	11	47	733	25	11	48	1198	20



11 49 34 529 43 11

EFFICIENCY STATISTICS

MOST EXPENDABLE CENTER IS 45, WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY 184450 IF DROPPED WITHOUT REPLACEMENT

MAXIMUM DISTANCE TRAVELED IS 76, FROM NODE 7 TO CENTER 34

TOTAL WEIGHTED DISTANCE IS 1561823

AVERAGE DISTANCE TO NEAREST CENTER IS 22

RATIO OF REFERENCE TO CURRENT VALUE OF OBJECTIVE FUNCTION IS 1.0000000

PERCENT CHANGE IN OBJECTIVE FUNCTION FROM STARTING SOLUTION IS 11.8825912

PERCENT CHANGE IN OBJECTIVE FUNCTION FROM PRECEDING CYCLE IS 11.8825912

TEITZ AND BART ALGORITHM CYCLE 2

OLD CENTER NEW CENTER

END OF TEITZ AND BART CYCLE 2  
THERE WERE 0 CHANGES DURING THIS CYCLE

CRITICAL DISTANCE IS 73

TEITZ AND BART ALGORITHM CYCLE 1

OLD CENTER NEW CENTER

1	2	1634253	44	34	3	16	2	45	31	11	12	10
2	22	1602799	44	34	3	16	22	45	31	11	12	10

END OF TEITZ AND BART CYCLE 1  
THERE WERE 2 CHANGES DURING THIS CYCLE

LIST BY DEMAND

CENTER	WEIGHT	DISTANCE * WEIGHT	AVERAGE DISTANCE	COST IF DROPPED
--------	--------	-------------------	------------------	-----------------

44	12686	147090	12	4117459
34	6717	223835	33	14658085
3	9661	232711	24	2172744
16	4553	56047	12	2015645
22	5421	183636	34	7301664
45	6299	186985	30	184450
31	5422	92066	17	189418
11	6047	188383	31	7359211
12	9030	246095	27	194100
10	4126	45951	11	188405

72



FOR THE ASSIGNMENT WHICH FOLLOWS:  
 TOTAL WEIGHTED DISTANCE IS 1602799

AVERAGE DISTANCE TO NEAREST CENTER IS 23

DESTINATION	CENTER	WEIGHT	DIST.	DESTINATION	CENTER	WEIGHT	DIST.	DESTINATION	CENTER	WEIGHT	DIST.				
11	1	22	2811	40	11	2	22	592	14	11	3	3	4027	0	11
11	4	45	1536	55	11	5	3	1241	36	11	6	11	845	30	11
11	7	22	422	44	11	8	11	819	35	11	9	12	2981	44	11
11	10	10	2767	0	11	11	11	933	0	11	12	12	3356	0	11
11	13	16	767	33	11	14	16	904	34	11	15	44	734	38	11
11	16	16	2882	0	11	17	44	1337	32	11	18	3	1905	55	11
11	19	11	1198	36	11	20	10	546	38	11	21	34	133	20	11
11	22	22	857	0	11	23	31	1087	18	11	24	34	1245	66	11
11	25	10	813	31	11	26	45	755	47	11	27	12	355	47	11
11	28	12	2328	42	11	29	45	1119	18	11	30	3	1755	37	11
11	31	31	1835	0	11	32	34	1205	34	11	33	44	1070	31	11
11	34	34	393	0	11	35	44	1607	12	11	36	34	1095	28	11
11	37	31	764	29	11	38	22	739	60	11	39	11	932	20	11
11	40	11	1320	55	11	41	34	973	20	11	42	45	1933	26	11
11	43	31	1736	29	11	44	44	6740	0	11	45	45	1086	0	11
11	46	34	1144	22	11	47	3	733	25	11	48	44	1198	20	11
11	49	34	529	43	11										

73

EFFICIENCY STATISTICS  
 MOST EXPENDABLE CENTER IS 45, WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY 184450 IF DROPPED WITHOUT REPLACEMENT

MAXIMUM DISTANCE TRAVELED IS 66, FROM NODE 24 TO CENTER 34

TOTAL WEIGHTED DISTANCE IS 1602799  
 AVERAGE DISTANCE TO NEAREST CENTER IS 23

RATIO OF REFERENCE TO CURRENT VALUE OF OBJECTIVE FUNCTION IS 0.9744347  
 PERCENT CHANGE IN OBJECTIVE FUNCTION FROM STARTING SOLUTION IS 9.5707417  
 PERCENT CHANGE IN OBJECTIVE FUNCTION FROM PRECEDING CYCLE IS -2.6235580

TEITZ AND BART ALGORITHM CYCLE 2

OLD CENTER NEW CENTER

END OF TEITZ AND BART CYCLE 2  
 THERE WERE 0 CHANGES DURING THIS CYCLE



THE ABOVE ASSIGNMENT IS FEASIBLE UNDER A DISTANCE CONSTRAINT OF 73

CENTER LOCATIONS AT BEGINNING OF PROBLEM 1

44 34 3 28 1 42 31 8 9 10

LOCATIONS OF CENTERS AT END OF ALGORITHM

44 34 3 16 22 45 31 11 12 10

END OF PROBLEM 1

WEIGHTED DISTANCE MATRIX REREAD FROM EXTERNAL STORAGE

WEIGHTED DISTANCE MATRIX REREAD FROM EXTERNAL STORAGE







43	1736	120
44	6740	0
45	1086	121
48	1198	20

CENTER ID 21 WEIGHT SERVED IS 17539 WEIGHT \* DISTANCE IS 1561661

AVERAGE DISTANCE FROM ALLOCATION TO CENTER IS 89  
 COST IF DROPPED WITHOUT REPLACEMENT IS 2012999  
 ALLOCATION

NODE	WEIGHT	DISTANCE
3	4027	128
5	1241	149
7	422	96
18	1905	83
21	133	0
24	1245	86
30	1755	91
32	1205	54
34	393	20
36	1095	48
38	739	80
41	473	40
46	1144	42
47	733	153
49	529	23

76

EFFICIENCY STATISTICS  
 MOST EXPENDABLE CENTER IS 21, WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY 2012999 IF DROPPED WITHOUT REPLACEMENT

MAXIMUM DISTANCE TRAVELED IS 153, FROM NODE 47 TO CENTER 21

TOTAL WEIGHTED DISTANCE IS 5369441  
 AVERAGE DISTANCE TO NEAREST CENTER IS 77

RATIO OF REFERENCE TO CURRENT VALUE OF OBJECTIVE FUNCTION IS 0.8590192  
 PERCENT CHANGE IN OBJECTIVE FUNCTION FROM STARTING SOLUTION IS 0.0  
 PERCENT CHANGE IN OBJECTIVE FUNCTION FROM PRECEDING CYCLE IS 0.0

MARANZANA ALGORITHM CYCLE 1



OLD CENTER NEW CENTER

21 3

END OF MARAZANA CYCLE 1  
THERE WERE 1 CHANGES DURING THIS CYCLE

LIST BY CENTERS

FOR THE ASSIGNMENT WHICH FOLLOWS:  
TOTAL WEIGHTED DISTANCE IS 4612453  
AVERAGE DISTANCE TO NEAREST CENTER IS 66

CENTER ID 44 WEIGHT SERVED IS 44673 WEIGHT \* DISTANCE IS 2919866  
AVERAGE DISTANCE FROM ALLOCATION TO CENTER IS 65  
COST IF DROPPED WITHOUT REPLACEMENT IS 5637011  
ALLOCATION

NODE	WEIGHT	DISTANCE
1	2811	75
2	592	129
4	1536	74
6	845	147
8	819	118
9	2981	55
10	2767	56
11	933	117
12	3356	73
13	767	91
14	904	92
15	734	38
16	2882	58
17	1337	32
19	1198	103
20	546	94
22	857	115
25	813	52
26	755	88
27	365	98
29	1119	103
33	1070	31
35	1607	12
39	932	97
40	1320	151
42	1803	113
44	6740	0
45	1066	121
48	1198	20



CENTER ID 3 WEIGHT SERVED IS 25289 WEIGHT \* DISTANCE IS 1892587  
 AVERAGE DISTANCE FROM ALLOCATION TO CENTER IS 67  
 COST IF DROPPED WITHOUT REPLACEMENT IS 2769987  
 ALLOCATION

NODE	WEIGHT	DISTANCE
3	4077	0
5	1241	36
7	422	130
18	1905	55
21	133	128
23	1087	85
24	1245	84
28	2328	84
30	1755	37
31	1835	67
32	1205	133
34	393	108
36	1095	90
37	764	54
38	739	122
41	973	128
63	1736	48
46	1144	130
47	733	25
49	529	151

EFFICIENCY STATISTICS

MOST EXPENDABLE CENTER IS 3, WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY 2769987 IF DROPPED WITHOUT REPLACEMENT  
 MAXIMUM DISTANCE TRAVELED IS 151, FROM NODE 40 TO CENTER 44

TOTAL WEIGHTED DISTANCE IS 4612453  
 AVERAGE DISTANCE TO NEAREST CENTER IS 66  
 RATIO OF REFERENCE TO CURRENT VALUE OF OBJECTIVE FUNCTION IS 1.0000000  
 PERCENT CHANGE IN OBJECTIVE FUNCTION FROM STARTING SOLUTION IS 14.0983885  
 PERCENT CHANGE IN OBJECTIVE FUNCTION FROM PRECEDING CYCLE IS 14.0983885

MARAZANA ALGORITHM CYCLE 2

OLD CENTER NEW CENTER



END OF MARAZANA CYCLE 2  
THERE WERE 0 CHANGES DURING THIS CYCLE  
CENTER LOCATIONS AT BEGINNING OF PROBLEM 2

44 21

LOCATIONS OF CENTERS AT END OF ALGORITHM

44 3

END OF PROBLEM 2

WEIGHTED AGGREGATE DISTANCE AT START AND END OF EACH PROBLEM  
1 1772434 1602799  
2 5369441 4612453



## CHAPTER IV

### A DATA STRUCTURE FOR LARGE P-MEDIAN PROBLEMS

This chapter describes the method that ALLOC VI uses to store the p-median data base, and describes a short program, UNRAVEL, that converts a distance matrix into this form.

#### Storing the p-Median Data Base in ALLOC VI

To this point, discussion of the p-median data base in the monograph has assumed that the data base would be stored as a matrix. This assumption was convenient for discussing the p-median problem. In addition, a matrix is a convenient, easily explained means for preparing and storing data. For large problems, however, a matrix is an inefficient form of data storage. It is inefficient in that it holds more data --in this case, weighted distances--than are needed to conduct an analysis. In this sense the matrix wastes core storage. In addition, if the program that analyzes the data cannot distinguish between needed and unneeded data, it will perform unnecessary calculations. In this sense the matrix wastes computer time.

#### Unnecessary Distance Data

Chapter I mentioned one type of unneeded information: when a candidate node is excluded from the p-median data base, the distances in its column of the distance matrix are no longer needed. When a p-median problem is large, however, or even if it is a moderate-sized one of 150 nodes or so, the data base will probably contain other types of unnecessary data.

For example, if the data base contains a large number of nodes and more than just a few centers, the longer distances in the data base are usually unnecessary. Consider an analysis that uses the townships and incorporated places of Iowa as nodes (Figure IV-1). At this scale of analysis, it is highly unlikely that a solution requiring Keokuk, Iowa to receive service from Sioux City would ever be considered an acceptable solution to the problem. Moreover, such a solution is unlikely to occur, given the distances involved and the distribution of population and activity within the state. Centers would almost certainly occur somewhere between these two nodes and permit them to receive service at a shorter distance. As the number of centers



in the problem increases, the likelihood of intervening centers would also increase, and the maximum distance needed to conduct the analysis would decrease.

This maximum distance is implicit in the problem. If the length of this implicit maximum distance were known, it could be treated as an ordinary maximum distance constraint. That is, distances greater than the implicit maximum constraint could be replaced by very large constants. A program could store the necessary distances and keep track of the constants in much less space than needed to store the distance matrix. Although the size of the implicit maximum distance constraint cannot be known with certainty until the optimization problem has been solved, its value can often be guessed at with confidence. To continue with the Iowa example, it may be known that no one will travel for more than an hour to receive a particular type of service. Assuming that distances are measured using the Manhattan metric, this yields a maximum travel radius for each node similar to that illustrated in Figure IV-1. Distances from node I in the figure to candidates outside the shaded area are irrelevant. Even if one hour is suspected of being too large a value for the implicit maximum distance constraint, it still can exclude a substantial portion of the distances from the data base. This example is based on an analysis of 2990 nodes (Rushton et al., 1976). The full distance matrix contained 8,940,100 distances, but only 661,454 of them, or 7.4 percent, were 55 miles or less.

For many problems the use of an implicit maximum distance constraint will identify most of the unneeded distances in the data base. If a problem contains fixed centers, however, other distances may also be superfluous. Returning to the Iowa example, assume that Des Moines has a center as in Figure IV-2. This assumption may reflect the fact that Des Moines is known to offer a particular service and is likely to do so in the future. Alternatively, this assumption may reflect a requirement that any statewide pattern of centers must have a center in Des Moines. In either case, the certainty that all solutions to the problem will have a center in Des Moines makes some distances unnecessary. Since each node will be served from its nearest center, no node will be served from a candidate when Des Moines is closer. In Figure IV-2, every node in the shaded region is nearer to Des Moines than to candidate j. Ignoring the distances from these nodes to candidate j will not affect the solution to the p-median problem. For a different candidate a different set of nodes would be affected. The principle here is easily extended to identify unneeded distances if the problem has two or more fixed centers. All else being equal, the more fixed centers in a problem, the more distances will become unnecessary.



FIGURE IV-1 Reducing the Number of Distances with an Implicit Maximum Distance Constraint

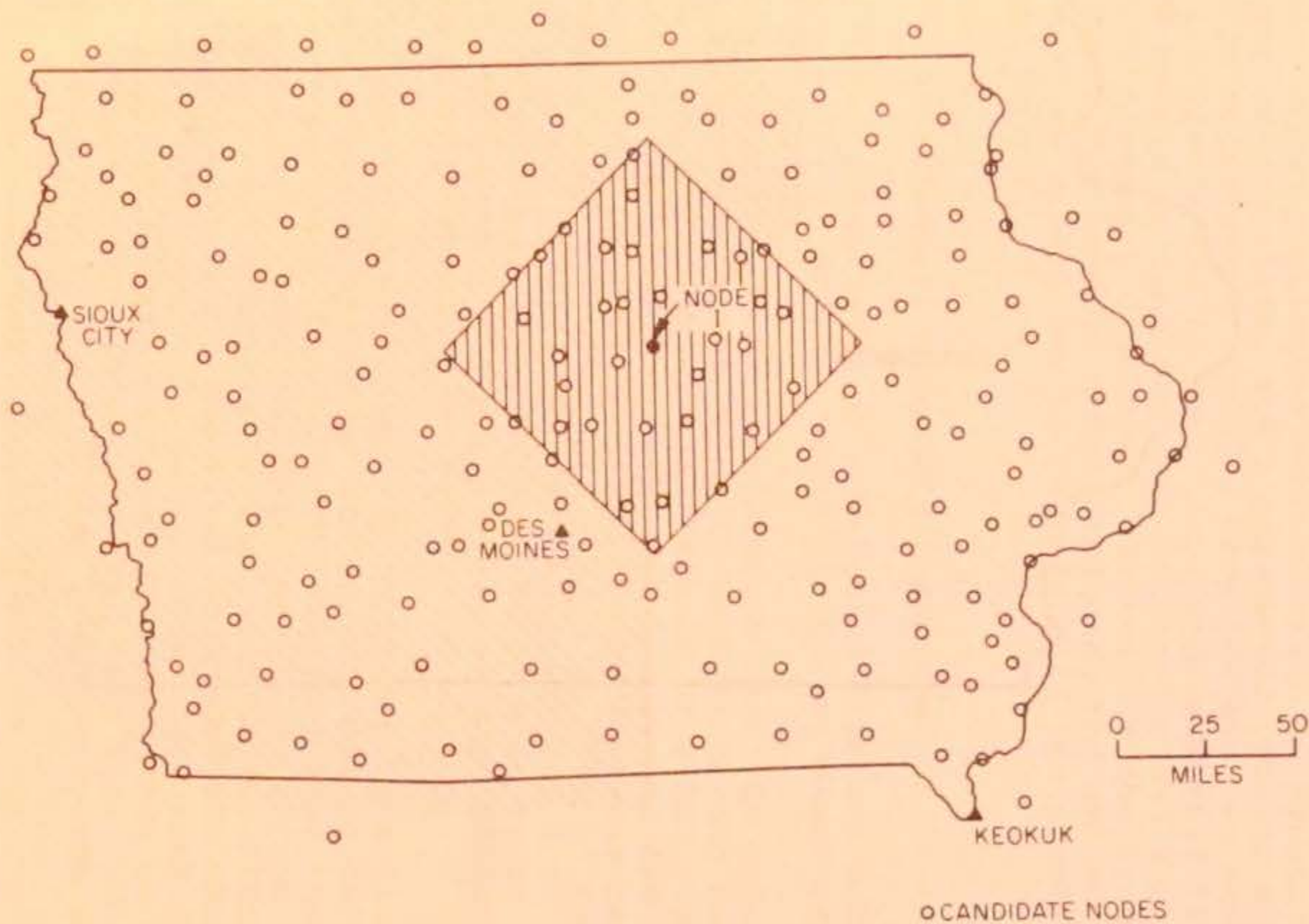
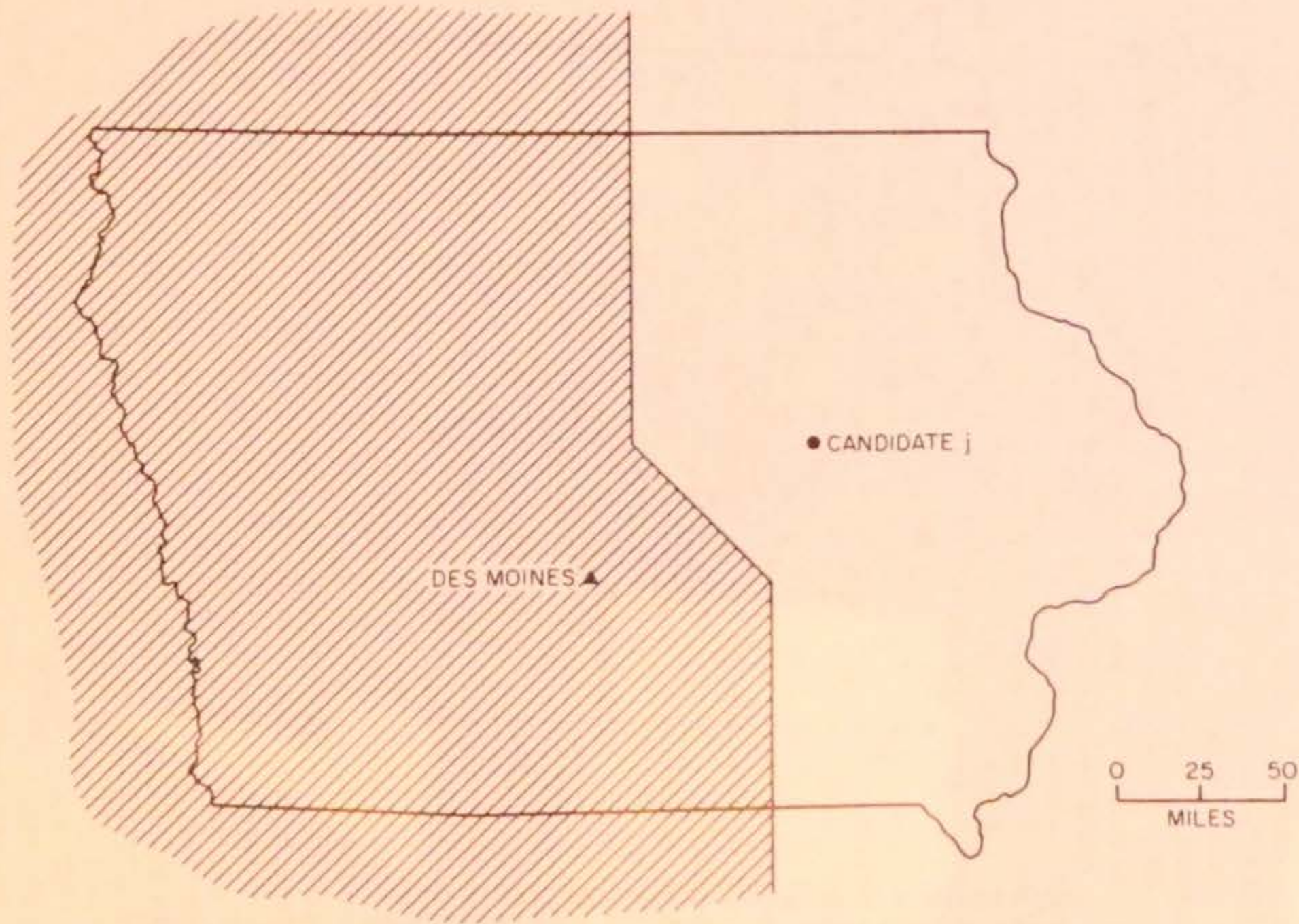




FIGURE IV-2 Reducing the Number of Distances  
with Fixed Centers





Fixed centers and the implicit maximum distance constraint will not identify unnecessary distances in every data base. Obviously, not all problems have fixed centers, and in some cases the important question is whether an existing center should continue. In some problems, the implicit maximum distance may be so large as to be useless. For example, the major population concentration in Wyoming is in one corner of the state, and a one-center problem would have to allow cross-state travel. Nevertheless, so many distances are irrelevant in most large problems that it becomes worthwhile to consider ways of storing and working with just the necessary distances. Indeed, if the problem has more than a few hundred nodes, reducing the amount of data that must be stored may be necessary if the problem is to be solved at all.

#### An Alternative to Matrix Storage

Figure IV-3 illustrates an unweighted distance matrix and its conversion to another means of storing distance data.

To illustrate the conversion, assume that the data base has ten nodes and that the problem to be defined can be solved within an implicit maximum distance of 20km. That is, all distances greater than 20km are irrelevant and may be discarded. For candidate node 1, these distances may be ignored. These distances are 26km in row 4, 40km in row 8, and 32km in row 9. In addition, assume that node 6 is nearer to a fixed center, say at node 7, than it is to candidate 1. Then the 13km in row 6 may also be discarded. This leaves six distances to be saved.

The six remaining distances may be placed in a list, as at the right of the figure. The distances could be in any order for many purposes, but ALLOC VI requires that they be sorted into ascending order and that the distance from the diagonal element of the matrix be the first in the list. To complete the conversion for the first column of the matrix, it is necessary to record in a second list where each distance came from. This would be necessary regardless of the order that distances were stored in the first list. Thus, the 0km came from row 1, the 5km came from row 5, the 8km from row 2, and so forth.

The relevant distances from the second column of the matrix may be extracted, sorted, and placed in the list in a similar way, beginning immediately after the last distance from the first column. Thus, the 0km came from row 2, the 8km from row 1, and so forth. Again, it is assumed that node 6 is nearer to a fixed center than it is to candidate 2.



The result of this conversion process is termed a distance file. The part of the file that applies to each candidate is termed a distance string. Thus, the first six distances in the file are the distance string for the first candidate, and the next four are the string for the second candidate. The complete distance file would contain a distance string for each candidate.

In order to use the distance file, it is necessary to know where each distance string begins and ends. This information is recorded in an index file, as the number of distances in each distance string. For the first candidate in the example, the index file would note that the first string contains six distances. For the second, it would note that the string contains four distances, and so forth.

It should be apparent from the preceding description that a distance file and index file together require slightly more than twice as much core storage, per distance stored, as a distance matrix. The distance file thus saves no storage unless at least half of the distances can be discarded.<sup>6</sup> This magnitude of reduction is fairly easy to achieve in problems with many nodes and many centers. In the 2990-node example cited earlier, the implicit maximum distance permitted roughly 93 percent of the distances to be discarded, and the use of fixed centers eliminated even more distances. Even on a moderate-sized problem such as the 150-node, 21-center test problem, a very loose implicit maximum distance constraint reduced the number of distances from 22,500 to 9,188. A smaller maximum distance, but one that was still larger than needed to solve the problem, reduced the number of distances further, to 4,844.

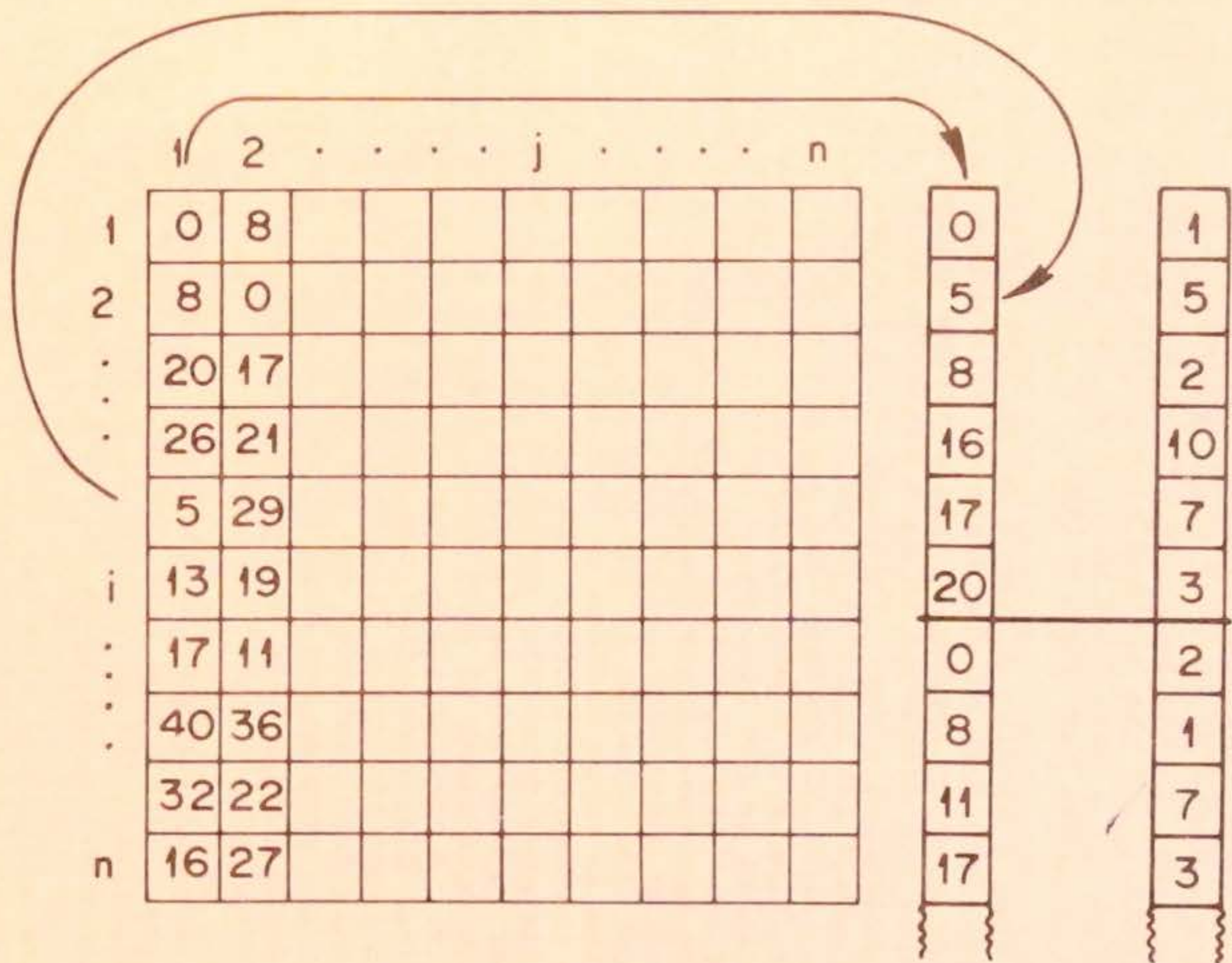
In addition to affecting the amount of storage needed for solving a problem, the implicit maximum distance constraint also affects the execution time of an algorithm that uses the distance file. Only the distances in the strings will be used in the algorithm's calculations. Reducing the number of distances in the strings reduces the number of calculations and therefore the time needed to solve the problem. Table II-3 reported execution times on the 150-node, 21-center test problem using two implicit maximum distances. The maximum distance of 100 yielded a file of 9,188 distances, and the maximum distance of 70 left 4,844. ALLOC VI needed much less time to solve the problem when it used the smaller distance file.

---

<sup>6</sup> ALLOC VI uses halfword storage for the row subscripts in the distance file. When this type of storage is available the distance file will save storage if as few as one third of the distances can be discarded.



FIGURE IV-3 Converting a Distance Matrix to Distance Strings





Although shortening the maximum distance reduces both core storage and execution time, it must be remembered that the maximum distance constraint was proposed as an implicit constraint. That is, solving the problem without the constraint will yield the same solution as solving the problem with it. If the implicit maximum distance is made too small in order to save time and storage, it will become a binding constraint that affects the location of service centers and the pattern of their service areas. There is thus a limit to the benefits of using the implicit distance constraint, and the minimum size of the implicit maximum distance for a data base will often not be known until after a few problems have been solved.

When a binding distance constraint is desired in a problem, the form of the distance file makes such a constraint easy to impose. Earlier it was stated that ALLOC VI requires the distances in each distance string to be sorted into ascending order. Consider again the example in Figure IV-3, where the implicit maximum distance constraint was 20km. ALLOC VI assumes that any distance not in the distance file is an arbitrarily large number. This causes the algorithms in the program to treat the implicit maximum distance as though it were a real maximum distance constraint on the problem. Since the implicit maximum distance is not a binding constraint, ALLOC VI's assumption does not affect the optimum solution to the problem. Now assume that a problem is to be defined so that no node is more than 16km from its nearest center. One way to impose the 16km maximum distance constraint would be to place a very large number in positions 5, 6, and 10 of the distance file. This is the equivalent of what ALLOC V would do to a distance matrix. A more efficient way, in terms of recovering the original distances after the problem has been solved, would be to reset the index file to record that the first string has only four distances and the second string only three. After the problem has been solved, ALLOC VI can recover the distances more easily by recomputing the index file than by rereading the distance file.

Several features of the index file require additional comment. First, the file contains distance ranges or classes. In describing the conversion of a distance matrix into a distance file, the index file was assumed to have a single distance class of 0-20km. It would have been possible to define additional classes, such as 0-5km, 0-10km, 0-15km, and 0-20km, and to create an index file that recorded the number of distances in each class. In this example, the file would



have recorded 2, 3, 3, and 6 for the first candidate and 1, 2, 3, and 4 for the second. If the index file is prepared with multiple classes, the classes must be ordered from smallest to largest, and each successive class must contain all of the distances in the class that precedes it, as in the above example.

ALLOC VI does not use multiple distance classes, but it was written to be compatible with other programs--as yet undocumented--that do. A single class, such as 0-20km in the example, would be adequate, and ALLOC VI could impose on a problem any distance in this class as a maximum distance constraint. The input data for ALLOC VI must indicate both the number of classes in the index file and the maximum distance to be stored from the distance file--in this case, 20km.

A second feature of the index file is that it simultaneously defines both the list of nodes and the list of candidates for a data base. The preceding examples have assumed that all nodes were to be candidates. If a node is to be served but is not a candidate, the node must appear in the index file, and it must record that the distance string for the node contains no distances. The distance file should not contain any string for a node that is not a candidate. Thus, if the first node in Figure IV-3 were not to be a candidate in the data base, the distance file to be read by ALLOC VI would begin with the string for the second node.

The directions for running ALLOC VI and RETRENCH specify the precise order of the information in the index and distance files. If the program UNRAVEL is used to convert a distance matrix into distance and index files, the resulting files will be completely suitable for use in these two programs. The structure of the files makes it easy to remove distances from the files but difficult to add distances to them. Both ALLOC VI and RETRENCH can remove distances from a distance file, but neither one can insert additional distances. For this reason, if there is doubt about how large to make the implicit maximum distance constraint when preparing the files, it is preferable to use a distance that is too large than one too small. In addition, it is probably most convenient for analysts to treat every node as a candidate when the files are prepared. The program RETRENCH can be used to remove candidates, shorten the implicit maximum distance constraint, and remove distances made superfluous by the use of fixed centers.

The examples in this section have used only two methods to determine which distances to place in a distance file.



Other methods could have been used as well. For example, the implicit maximum distance constraint might vary from node to node, or from candidate to candidate. Alternatively, the populations of one node could be prevented from interacting with another by excluding the distance between them from the distance file. UNRAVEL, ALLOC VI, and RETRENCH cannot use these methods to prepare distance and index files, but both ALLOC VI and RETRENCH can use these files in their analyses.

### Running UNRAVEL

This section describes UNRAVEL, a program to convert a distance matrix into the distance and index files required by ALLOC VI and RETRENCH. It also gives directions for running the program.

UNRAVEL reads each column of a distance matrix, identifies those distances within a specified maximum distance, sorts them, and writes them in a machine-readable format for use by ALLOC VI and RETRENCH. UNRAVEL presently uses a tree sort adapted from Day (1972), but any sorting procedure could be used. The study that led to the development of ALLOC VI (Rushton et al., 1976) used a different program based upon a bucket sort (Aho et al., 1974). This type of sort would probably be more efficient than the one in UNRAVEL, but the code for that program was insufficiently general to be included in this monograph.

It should be emphasized that UNRAVEL reads and sorts columns of a distance matrix, not rows. When the distances in the matrix are symmetrical, as they are when computed by DISTANCE or by SPA, this distinction is irrelevant. If the distances in the matrix are not symmetrical, however, and if the matrix has been used in ALLOC V, then the matrix must be transposed before it is processed by UNRAVEL.

For compatibility UNRAVEL can be directed to create index and distance files as either formatted or unformatted data. ALLOC VI can read either type of data, but RETRENCH requires that the two files be unformatted. The files may be written on punched cards or other storage media. If the files are written as formatted data, UNRAVEL also creates the format cards that ALLOC VI will need to read them (Figure IV-4). A detailed description of the machine-readable output from UNRAVEL is given in the description of the input data for ALLOC VI.

Input to UNRAVEL consists of a control card, a variable format card, and a formatted, square distance matrix (Figure



IV-4). The control card and format must be read from punched cards, but the matrix may be read from another storage medium designated by the control card. With the exception of the variable format card, all data are to be punched as integer variables and are to be right-justified in their allotted fields. The variable format should be punched left-justified, for convenience.

1.0 CONTROL CARD (required).

- 1.1 In columns 1-5, punch the number of nodes in the square distance matrix (rows or columns).
- 1.2 In columns 6-10, punch the largest distance to be saved in the distance file. If left blank, this value will default to 32000.
- 1.3 If your distance matrix is to be read from punched cards, skip to 1.4 below. Otherwise, in columns 11-15, punch the unit number for the tape, disk or other medium from which the matrix is to be read.
- 1.4 If your distance and index files are to be written on punched cards, skip to 1.5 below. Otherwise, in columns 16-20, punch the unit number for the tape, disk, or other medium on which the matrix is to be read.
- 1.5 If you want your distance and index files written as formatted data, skip to 2.0 below. Otherwise, punch a 1 in column 25 to have these files written as unformatted data.

2.0 FORMAT TO READ DISTANCE MATRIX (required).

- 2.1 On the next card, punch a Fortran format to read the ID number and distances in one column of the square distance matrix. This format must specify integer fields.

As an example, assume that a distance matrix for the 99 counties in Iowa has been punched one column at a time, with the county ID number punched at the start of the column, and with each ID number and distance punched in a field of five columns. Each ID number and column of the matrix might require 6 cards of 15 numbers each and one card with only 10 numbers. This ID number and column could be read using a format of (15I5), or a format of (6(15I5/),10I5).



If your matrix has been prepared by SPA (Ostresh, 1973), your format is (14I5). If your matrix has been prepared by DISTANCE (Chapter I), your format is also (14I5), and you should use the format card punched by that program.

### 3.0 DISTANCE MATRIX (required).

- 3.1 The remainder of the input deck contains the node ID numbers and distances to be read with the format. The ID number for each column must precede the distances in the column. If your matrix has been prepared by DISTANCE or by SPA, it meets all requirements of UNRAVEL and may be used without change.

### Error Conditions

The following condition will produce an error message and stop the program.

1. Request for the distance and index files to be written on punched cards (1.4) but as unformatted data.

### Program Dimensions

The array dimensions of UNRAVEL presently permit its use to reformat a matrix for 150 nodes. The comment cards at the beginning of the program give directions for changing the dimensions of the program arrays. With its current dimensions, UNRAVEL requires only 36 K bytes of core storage after it has been compiled. Core requirements will be somewhat greater if the matrix is not read from punched cards, or if the index and distance files are not written on punched cards. In these cases, additional space will be needed for input and output buffers. For its present dimensions of 150 nodes, UNRAVEL requires 36K bytes of core if all input and output use punched cards. The use of disks or tapes may require small additional amounts of storage.



FIGURE IV-4a Sample Input Deck for UNRAVEL

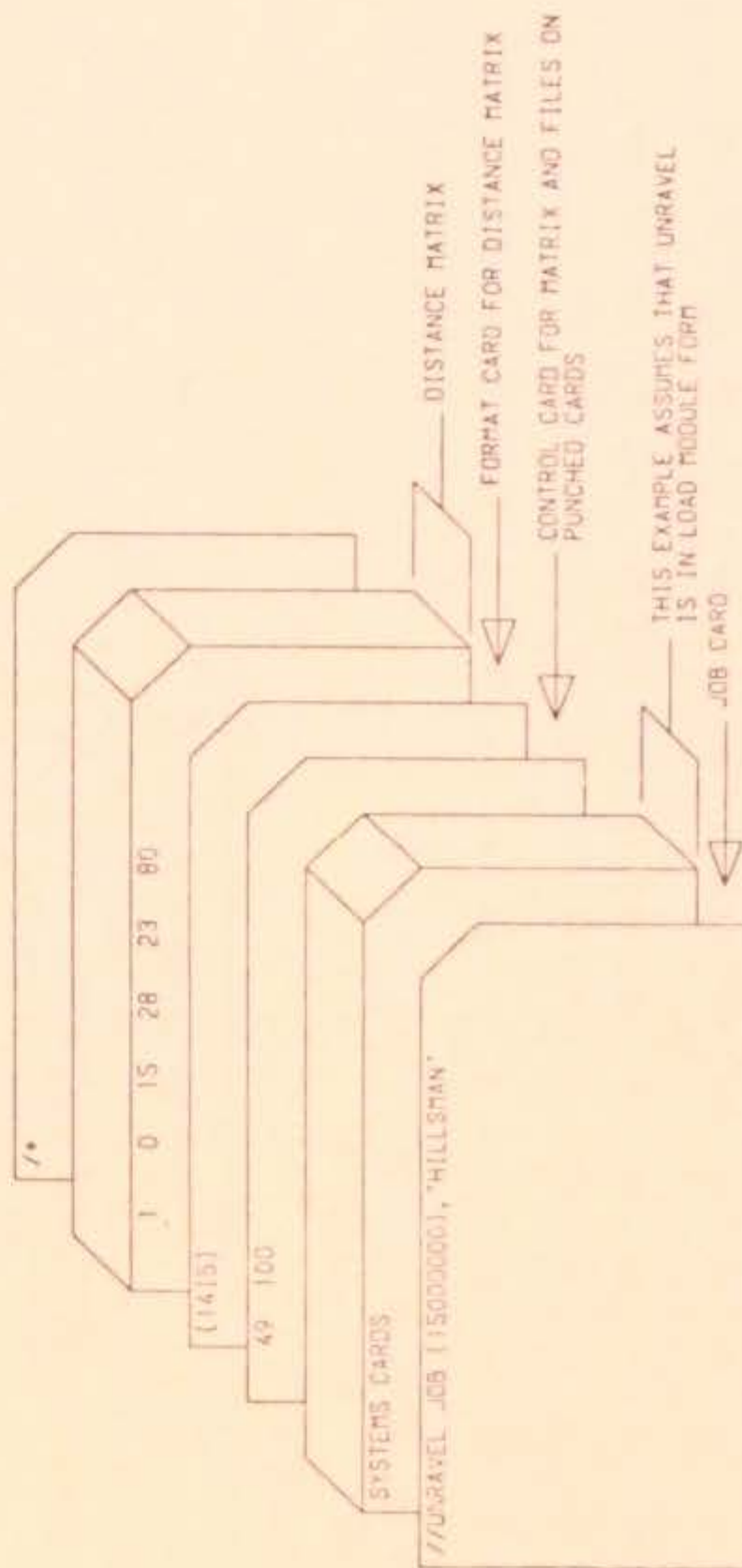




FIGURE IV-4b Sample Punched Output Deck from UNRAVEL

94

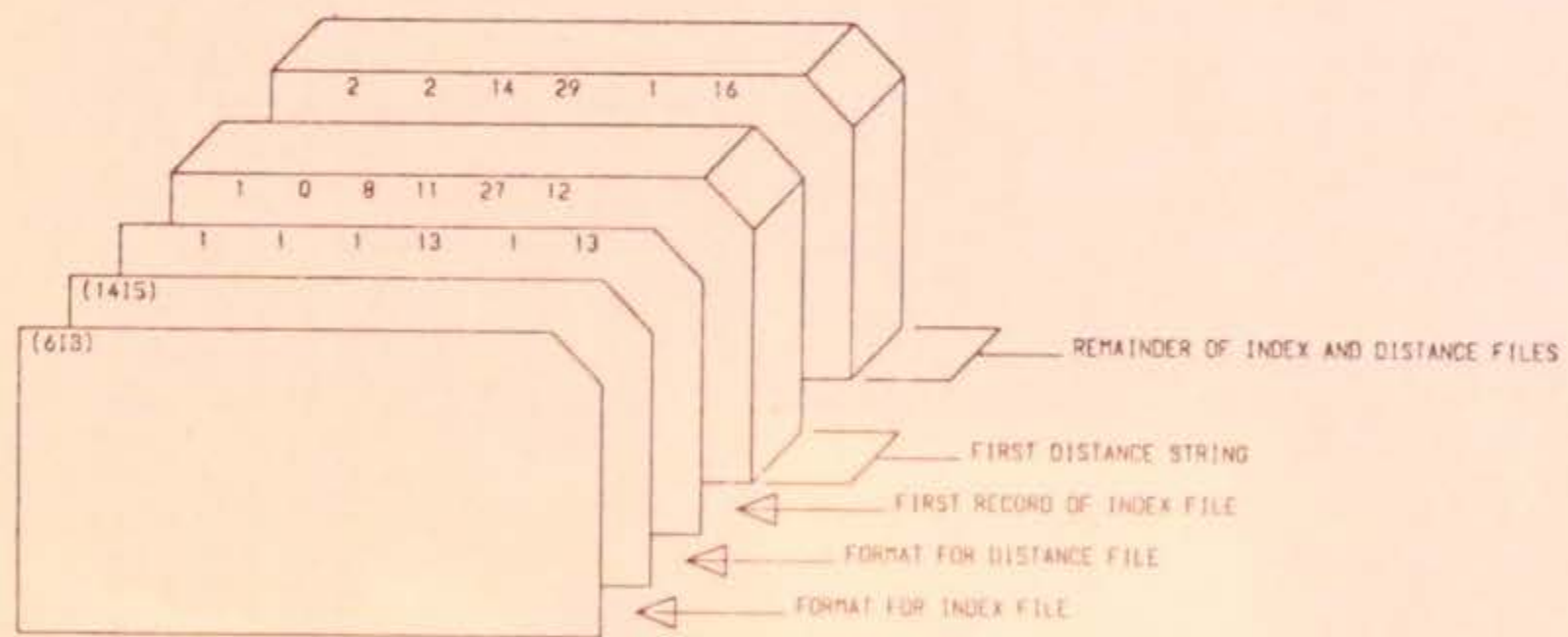




FIGURE IV-5 Sample Printed Output from UNRAVEL

PROGRAM UNRAVEL

WRITTEN BY EDWARD L. HILLSMAN

DEPARTMENT OF GEOGRAPHY

THE UNIVERSITY OF IOWA

NOVEMBER, 1974

CREATE INDEX AND DISTANCE FILES FOR 49 NODES

ORDER DISTANCES IN INCREASING VALUES UP TO AND INCLUDING VALUES OF 100

READ DISTANCE MATRIX FROM UNIT 5

WRITE INDEX AND DISTANCE FILES ON UNIT 7

THESE FILES WILL BE FORMATTED

READ DISTANCE MATRIX WITH FORMAT OF (18I4)

A LISTING OF THE INDEX AND DISTANCE FILES FOLLOWS

1	0	22	40	33	44	28	51	2	54	44	75	17	81	7	84	43	87	35	87
12	93	48	95	38	100														
2	0	22	14	7	34	38	50	1	54	28	59	36	82	43	95	33	98		
3	0	47	25	5	36	30	37	43	48	37	54	18	55	31	67	28	84	24	84
23	85	36	90																
4	0	26	17	25	36	45	55	35	62	10	67	29	73	44	74	42	81	48	94
5	0	3	36	47	61	24	63	30	73	43	84	37	90	18	91				
6	0	11	30	40	40	39	50	19	50	8	65	20	82	10	97				
7	0	38	16	2	34	22	44	36	48	34	76	18	83	1	84	28	89	32	91
41	96	21	96	46	98														







26	0	4	17	25	36	45	47	29	65	10	67	42	73	35	76	44	88	14	96
27	0	27	314		330		1	17											
27	0	8	41	9	43	12	47	31	54	23	72	40	74	11	76	48	78	43	83
37	83	17	88	28	89	47	96	39	96	10	96	44	98						
28	0	28	331		348		1	18											
28	0	43	36	12	42	22	45	1	51	17	52	2	59	31	65	23	83	44	84
3	84	18	86	9	86	33	89	27	89	7	89	37	94	35	96				
29	0	29	349		356		1	8											
29	0	45	18	42	29	14	31	16	45	26	65	13	67	4	73				
30	0	30	357		369		1	13											
30	0	3	37	24	47	47	62	34	71	5	73	43	85	41	91	37	91	21	91
18	92	46	93	36	99														
31	0	31	370		382		1	13											
31	0	23	18	43	29	37	29	47	42	12	47	27	54	28	65	3	67	18	79
17	88	9	91	8	95														
32	0	32	383		393		1	11											
32	0	41	14	34	34	36	43	21	54	46	56	38	75	49	77	18	78	7	91
24	100																		
33	0	33	394		408		1	15											
33	0	44	31	17	37	35	43	1	44	48	51	15	69	12	78	25	83	22	84
9	86	10	87	28	89	16	89	2	98										
34	0	34	409		420		1	12											
34	0	41	20	21	20	46	22	36	28	32	34	49	43	38	60	18	63	24	66
30	71	7	76																
35	0	35	421		438		1	18											
35	0	44	12	48	32	25	40	33	43	17	44	10	44	15	50	4	62	9	67
16	70	26	76	20	82	12	85	1	87	19	91	28	96	39	98				
36	0	36	439		454		1	16											
36	0	34	28	38	32	18	35	32	43	41	48	21	48	7	48	46	50	49	71
2	82	43	85	3	90	22	92	24	94	30	99								
37	0	37	455		465		1	11											
37	0	31	29	47	34	23	47	3	54	43	58	12	76	27	83	5	90	30	91
28	94																		
38	0	38	466		477		1	12											
38	0	7	16	36	32	2	50	34	60	22	60	18	67	32	75	41	80	21	80
46	82	1	100																
39	0	39	478		491		1	14											
39	0	11	20	19	30	6	50	10	54	8	55	20	58	9	60	40	75	25	85
48	55	27	96	44	97	35	98												
40	0	40	492		499		1	8											
40	0	8	33	6	40	11	55	27	74	39	75	19	90	9	96				
41	0	41	500		511		1	12											
41	0	32	14	34	20	21	40	46	42	36	48	49	63	38	80	18	83	24	86
30	91	7	96																
42	0	42	512		519		1	8											
42	0	45	26	29	29	16	55	14	60	26	73	4	81	13	88				
43	0	43	520		536		1	17											
43	0	31	29	28	36	23	47	3	48	18	50	37	58	47	71	12	76	22	81
27	83	5	84	36	85	30	85	1	87	17	88	2	95						
44	0	44	537		556		1	20											
44	0	35	12	48	20	33	31	17	32	15	38	25	52	9	55	10	56	16	58



12	73	4	74	1	75	28	84	26	88	13	91	14	92	20	94	39	97	27	98
45	0	45	557	565	26	26	47	14	49	4	55	16	63	25	83	13	85		
46	0	46	566	577	38	41	42	21	42	36	50	32	56	38	82	18	85	24	88
30	93	7	98	578	34	31	42	23	60	5	61	30	62	43	71	18	80	12	89
47	0	3	25	37	604	9	35	33	51	17	52	15	58	25	72	10	76	27	78
27	96	48	589	32	94	39	95	1	95	8	98								
48	0	44	20	35	611	34	43	41	63	36	71	32	77						
16	78	12	79	4															
49	0	49	605																
49	0	21	23	46															



## Chapter V

### ALLOC VI

Several features of ALLOC VI are different from those of ALLOC V and the general features described in Chapter I. The first section of this chapter is a description of these differences. The description is followed by a description of the input data and operating instructions for ALLOC VI.

#### Differences between ALLOC VI and ALLOC V

The differences between ALLOC VI and ALLOC V are of several types. First, ALLOC VI contains several features not found in ALLOC V. For example, ALLOC VI permits use of the add algorithm and the trade-off algorithm described in Chapter II. Both of these algorithms require data in addition to that which ALLOC V requires. In addition, some features of ALLOC VI were developed for one specific purpose or another but are seen as potentially useful for other analyses as well. Thus, ALLOC VI has features that let the analyst consider the edge of a study region. In addition to these enhancements to the general ALLOC features, the design of ALLOC VI for use on very large problems led to different forms of input and output. For example, the method of preparing location constraint data for ALLOC V is awkward for large numbers of nodes, and it encourages rather than discourages errors; it was changed in ALLOC VI. Similarly, the use of distance and index files made some types of output easy to produce, and code was written to obtain them. The differences between the two programs are discussed below, first for the data base and then for the problem definition data required by ALLOC VI. The discussion assumes familiarity with the general discussion of program features in Chapter I.

Before proceeding, however, it should be noted that ALLOC VI lacks two features found in ALLOC V. It cannot use the Maranzana algorithm to solve a problem, and it cannot automatically construct a spatial hierarchy.

#### Data Base Features

ALLOC VI requires a title card as part of its data base. The program prints the title information at the beginning of the printed description of the problem data base and at the beginning of each problem definition. This makes it easier to identify the printed output from different job submissions.

Chapter I noted that the ALLOC programs contain limits on the size of the population that may be used in a data



base. In ALLOC V this limit depends upon the number of nodes in the list of nodes and the largest weighted distance: the product of these two numbers must not exceed 2,147,483,647 or the algorithms may fail to find a solution.<sup>7</sup> A similar limit exists in ALLOC VI, but it is computed using the number of distances in the longest distance string instead of the number of nodes. For a given size of problem this permits the use of larger populations. If a data base exceeds this limit, dividing the population of every node by a constant will reduce the size of the largest weighted distance without affecting the definition or solution of a problem. ALLOC VI can be requested to scale the population in this manner, with the resulting population rounded to the nearest integer. When the range of populations in a data base is large, however, dividing the population by a constant may cause some of the smaller populations to be recorded as zeroes. The program will treat these nodes as "dummy" nodes in the printed output.

ALLOC VI can read a distance file and impose a tighter implicit maximum distance constraint on the data base. For example, assume that the distance file was created with a maximum distance of 100km and that all of the problems to be defined can be solved with distances of 70km or less. ALLOC VI will read all of the distances in the file, but it can be directed to discard those that are too large. Use of this feature causes ALLOC VI to store fewer distances and operate in less time. Tightening the implicit maximum distance in this way does not preclude or require the use of tighter maximum distance constraints later, when problems are defined and solved.

To reduce the time needed for reading the p-median data base, ALLOC VI has been written to read the distance and index files and the node populations as either formatted or unformatted data. Unformatted files are read and processed more quickly than formatted ones, although no tests have been done to determine how much time is saved with unformatted files. If either the index or the distance file is unformatted, the other must be. The populations may be formatted or unformatted regardless of the form of the distance and index files. If the populations are formatted they may be in any order within the populations deck, as for ALLOC V. If the populations are unformatted, the populations must be in the same order as the index file and the list of nodes. The program RETRENCH, discussed in Chapter VI, will unformat a formatted population deck when it edits the distance file of a data base.

If fixed centers are used to identify and eliminate unneeded distances in the distance file, as discussed in Chapter IV, then the fixed centers must be made a part of

---

<sup>7</sup>This number is machine dependent and may be larger or smaller on machines other than the IBM 360/65 on which the programs were developed.



the resulting data base. In addition, fixed centers may be placed in a data base even if they have not been used to reduce the size of the distance file. When a data base contains fixed centers, ALLOC VI automatically makes these centers a part of the starting solution for every problem definition. This eliminates the need to prepare the list as a part of every starting solution, and it is particularly useful if a large number of centers must appear in every problem. The section on punching the data base gives directions for preparing the list of fixed centers.

If the trade-off algorithm is to be used to solve a problem, the data base must contain a measure of the suitability of each candidate node. As noted in the discussion of the trade-off algorithm, the suitability scores must range from zero to one, inclusive. As with fixed centers, suitability information may be included in a data base even when the trade-off algorithm will not be used. In addition to using the suitability information in the trade-off algorithm, ALLOC VI prints the suitability of each node in the starting solution and final solution for each problem. The suitability information is labeled "SECOND FACTOR" in the printed output. If no second factor information is read, ALLOC VI sets the suitability score of each candidate to zero.

The final new feature of the ALLOC VI data base is its recognition that study areas have edges. For example, Figure IV-1 shows a study region, Iowa, and several candidate nodes that lie beyond its borders. These candidates might have centers that could serve persons within the study region. It is also possible that some nodes might lie outside the study region and yet be served by centers within it. ALLOC VI requires that the list of nodes declare whether or not each node lies inside the study region or outside of it. If no distinction is to be made, then all nodes must be declared to be inside. ALLOC VI uses these declarations to compute two sets of summary statistics for the starting and final solutions to each problem. One set is computed using all of the nodes in the data base, and one is computed using only the nodes inside the study region. These statistics include the aggregate and average distance for each group of nodes and the longest distance traveled by a node in each group to its nearest center. The longest distance from a node inside the study region may be to a center outside.

The use of a study region edge within a data base affects only the computing and printing of these statistics.



It does not affect the operation of the algorithms, the use of maximum distance constraints, or any other program feature. The algorithms locate centers to minimize aggregate distance from all nodes in the data base to their nearest center, not just from those inside the study region. Similarly, ALLOC VI applies a maximum distance constraint in a problem definition to all nodes, not just those inside.

A node is designated as lying inside or outside the study region by means of an extra field on the index file. The program UNRAVEL assumes that all nodes fall inside the boundary of the study region when it creates index file records. A node may be declared to be outside by processing the index file from UNRAVEL and altering this field on the appropriate records.

#### Problem Definition Features

The basic elements of a problem definition for ALLOC VI are similar to those for ALLOC V. The definition must specify the number of centers in the problem, a starting solution and, usually, an algorithm to solve the problem. Location constraints may be imposed on a problem and may be repeated automatically for several problem definitions. Maximum distance constraints may be imposed on a problem, and a constraint may be computed after a problem is solved, to try to improve the solution. ALLOC VI requires most of this information in different forms than ALLOC V does, however, and it also permits additional information in a definition.

The form in which ALLOC VI reads location constraint data was designed to make it easier to keep records of the constraints imposed on a problem, and to be more convenient than ALLOC V when the data base contains many nodes. For example, when ALLOC V reads information to change even one constraint, it requires a list of the constraints for every node in the data base, and it requires the list in a form that is susceptible to keypunching errors. ALLOC VI requires information on location constraints only for the nodes whose constraints are to be changed, and it requires that constraints be imposed using the ID number of the node. Where ALLOC V assumes that constraints are to be cleared after each problem unless specifically requested to repeat them, ALLOC VI assumes that constraints are to be repeated unless specifically requested to clear or change them. ALLOC VI prints information about location constraints in a form that is easier to interpret than the one from ALLOC V. ALLOC VI may be requested to print the current constraints without changing them.

The printed output from ALLOC VI differs slightly from that of ALLOC V. Unlike ALLOC V, ALLOC VI does not print information about the solution at the end of each cycle of an algorithm. It prints information only about a problem's



starting solution, and about the final solution from each algorithm if the algorithm improves the solution it starts with. If the problem definition contains several maximum distance constraints, ALLOC VI will print information about the best solution found for each constraint.

ALLOC VI prints summary information about the entire solution and about the service area of each center, regardless of whether it prints information about the list of nodes. As with ALLOC V, ALLOC VI can print information about the list of nodes in the order that the list was read into the data base, or it can group nodes by service area. Both forms of output are printed three nodes to a line.

ALLOC VI can also produce machine-readable output. When it is requested, machine-readable output is written at the same time as the printed output discussed above. All machine-readable output is written on output unit 7, and the unit may be defined to write on punched cards, magnetic tape, or disk. Two levels of machine-readable output may be requested. The first level writes a header record containing the problem definition number, the data base title, and information from the control card. This information is written only once for the problem. The first level also writes the number and locations of centers each time printed output is prepared. The second level writes all of the information in the first and, in addition, it writes additional information about the list of nodes and the service areas of the centers. Table V-1 lists the information written for each level and the formats used to write it. The information is written on unit 7 in the sequence given in the table.

ALLOC VI is quite different from ALLOC V in its treatment of nodes that cannot be served within a maximum distance. ALLOC V assigns these nodes to receive service arbitrarily from one of the centers at very high cost. ALLOC VI creates a "dummy" fixed center with an ID number of zero. The "dummy" center is assumed to follow the last node in the list of nodes. All nodes that cannot be served within a maximum distance are assumed to receive service from this "dummy" center. ALLOC VI prepares a list of these nodes as part of its printed output. The weighted distance from each of these nodes to the "dummy" center is recorded as a very large number within ALLOC VI. The program cannot recover the unweighted distance from these nodes to the "dummy" center, however, and it arbitrarily reports the unweighted distances from these nodes as -1 in the printed and machine-readable output.

The problem definitions for ALLOC VI may contain information not required in ALLOC V. Each problem definition



that uses the trade-off algorithm must specify the parameters to be used in the trade-off function (Figure II-3). In addition, when the add algorithm is used, the problem definition must contain special control cards to tell the algorithm how many centers to add and how to add them.

The add algorithm can operate in either of two modes. Both require a starting solution containing at least one center, and both require a control card to specify the number of centers to be added to that solution. In one mode, the algorithm operates exactly as described in the earlier section on algorithms. That is, it adds the center that gives the new solution the lowest aggregate distance, and it does so until it has added the desired number of centers to the starting solution. In the second mode, the user tells the algorithm where to add each center and whether to leave it in the solution or drop it after information about its service area is printed.

After a center has been added in either mode, ALLOC VI prints a list of the nodes in the new center's service area, the population served, and the aggregate distance for the entire solution. ALLOC VI may be directed to write the list of nodes in the service area in machine-readable form on unit 7. Table V-1 indicates the content and format of the service area list. Machine-readable output from the add algorithm is controlled separately from the rest of the machine-readable output. That is, requesting it from the add algorithm will not produce the machine-readable output described earlier, and requesting the output described earlier will not produce machine-readable output from the add algorithm. A problem definition can request both types, either, or neither.

#### Running ALLOC VI

This section contains directions for punching the data base cards and problem definition cards needed to run ALLOC VI. The directions that follow assume familiarity with the features of the ALLOC programs described in an earlier section and with the distance file, index file, and program features just discussed.

For convenience the directions assume that all data are to be read from punched cards, but only the title card and control card must actually be punched cards. The control card indicates to the program whether remaining data are to be read from punched cards or from other storage media.

The title card and the three variable format cards in the data base are read as alphanumeric data. The file of



Table V-1

## Machine-readable Output from ALLOC VI

Level	Information	Number of Items	Format
1,2	problem definition number, number of centers, number of nodes, number of fixed centers, code for first algorithm, code for second algorithm, maximum distance in data base, level of machine-readable output, title	8*	(7I5,5X,10A4)
1,2	number of centers, position of each center in the list of nodes	M+1	(16I5)
2	population served by each center	M	(8I10)
2	aggregate distance traveled within each service area	M	(8I10)
2	average distance traveled within each service area	M	(8F10.3)
2	expendability of each center	M	(18I5)
2	position of each node's nearest center in the list of nodes	N	(16I5)
2	position of each node's second nearest center in the list of nodes	N	(16I5)
2	distance from each node to its nearest center	N	(16I5)
2	weighted distance from each node to its nearest center	N	(8I10)
2	weighted distance from each node to its second nearest center	N	(8I10)
**	number of nodes in the service area of a center added by the add algorithm, position of each of these nodes in the list of nodes	K+1	(16I5)

Source: Compiled by author. M denotes the number of centers in the problem, N denotes the number of nodes in the list of nodes, and K denotes the number of nodes in the service area of a center added by the add algorithm.

\*The title is counted as a single item.

\*\*Controlled separately from the rest of the machine-readable output. See text for discussion.



suitability scores or other second factor information and the parameters of the trade-off algorithm are read as real numbers. All other data are read as integers. The distance, index, and population files may be read as formatted or as unformatted integer data. All formatted integers are to be right-justified in their allotted fields.

The section contains three parts. The first gives directions for punching the p-median data base, and the second gives directions for punching cards to define and solve problems. The third section lists conditions that will stop the program or cause error messages to be printed.

### Punching the p-Median Data Base

The p-median data base for ALLOC VI consists of a title card, a control card, as many as three format cards, and as many as five decks or files of information. The five decks contain: the index file; the distance file; the node populations; a list of fixed centers; and the suitability or second factor of each candidate node. Figure V-1 illustrates a sample input deck for ALLOC VI.

#### 1.0 TITLE CARD (required).

In columns 1-40, punch any desired title information. This title will be printed at the start of each problem, and it will appear in the header record of any machine-readable output.

#### 2.0 CONTROL CARD (required).

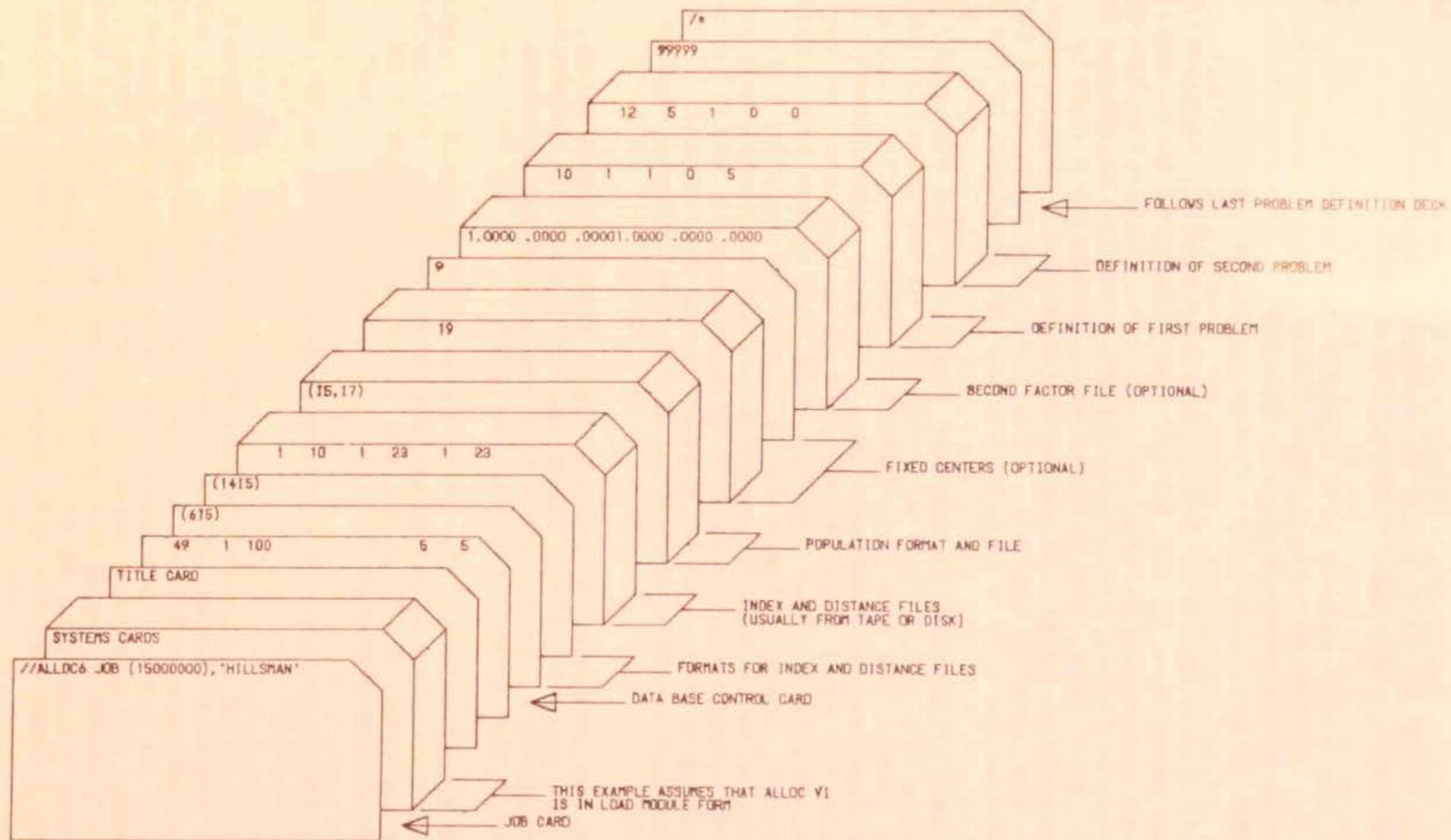
2.1 In columns 1-5, punch the number of nodes in the list of nodes (number of rows in the original distance matrix). This number will be referred to as  $N$ , and it will appear in the header record of any machine-readable output.

2.2 If you obtained your index file from the program UNRAVEL or the program RETRENCH, punch a 1 in column 10. Otherwise, in columns 6-10, punch the number of distance classes in your index file.

2.3 In columns 11-15, punch the longest distance to be stored during this submission of the program. ALLOC VI will read every distance from the distance file, ignore any that are greater than this maximum, and compute a new index file for the shortened distance file. If all distances from the distance file are to be stored, this field must contain a distance equal to or larger than the longest distance in the file.



FIGURE V-1 Sample Input Deck for ALLOC VI





The value in this field will appear in the header record for any machine-readable output.

- 2.4 If the nodes are to have unequal populations, skip to 2.5. Otherwise, in columns 16-20, punch the population to be given to each of the  $N$  nodes in the data base. Then skip to 2.6.
- 2.5 This field controls the population scaling feature of the program. If you do not want the program to scale your node populations, skip to 2.6. Otherwise, punch a scale factor in columns 21-25. Before weighting the distance file, the program will divide each node's population by the scale factor and round the quotient to the nearest integer. The resulting value will be used as the node's population for the remainder of the run.
- 2.6 This field controls whether the index and distance files are to be read as formatted or unformatted data. If your files are from RETRENCH (Chapter VI), they must be read as unformatted data. If your files come directly from UNRAVEL (Chapter IV), they must be read as created by that program. If both files are unformatted, leave columns 26-30 blank and skip to 2.7. Otherwise, both files must be formatted data, and you must supply the program with formats with which to read the two files. In this case, punch in columns 26-30 the input unit number from which the formats will be read.
- 2.7 If the index file is to be read from punched cards, leave columns 31-35 blank and skip to 2.8. Otherwise, in columns 31-35, punch the input unit number from which the index file will be read.
- 2.8 If the distance file is to be read from punched cards, leave columns 36-40 blank and skip to 2.9. Otherwise, in columns 36-40, punch the input unit number from which the distance file will be read.
- 2.9 If the nodes are to have equal populations (2.4), skip to 2.11.

This field controls whether the node populations are to be read as formatted or unformatted data. If the populations are unformatted, leave columns 41-45 blank and skip to 2.10. Otherwise, the populations are formatted data, and you must supply the program with a format with which to read the population file. In this case, punch in columns 41-45 the input unit number from which the format for the node population file will be read.

- 2.10 If the population file is to be read from punched cards, leave columns 46-50 blank and skip to 2.11. Otherwise, in columns 46-50, punch the input unit number from which the population file will be read.



- 2.11 If your data does not contain any fixed centers, leave columns 51-55 blank and skip to 2.12. Otherwise, in columns 51-55, punch the input unit number from which the list of fixed centers is to be read.
- 2.12 If your data base does not contain a second factor file, leave columns 56-60 blank and skip to 2.13. Otherwise, punch in columns 56-60 the input unit number from which the second factor file is to be read.
- 2.13 If your problem definitions are to be read from punched cards, leave columns 61-65 blank and skip to 2.14. Otherwise, punch in columns 61-65 the input unit number from which the program is to read your problem definitions.
- 2.14 If your index and distance files are unformatted (2.6), skip to 4.0.
- 3.0 FORMATS TO READ INDEX AND DISTANCE FILES (optional).
- 3.1 On the next card, punch a Fortran format to read the information for one node in the index file. This format must specify five integer fields, plus one integer field for each distance class in the index file. An example would be (I5,I8,2I7,3X,I2,3X,I4) for a file with one distance class. If your index files were created by UNRAVEL, your format is (6I8).
- 3.2 On the next card, punch a Fortran format to read the distance string for one node in the distance file. An example would be (I4I5), the format for files created by UNRAVEL.
- 4.0 INDEX AND DISTANCE FILES (required).
- 4.1 The contents of the index and distance files were described in an earlier section, with an example. The contents of these files are listed briefly here. If your files were prepared by the program UNRAVEL or the program RETRENCH, they match the descriptions below and you may skip to 4.9.
- 4.2 For each node in the list of nodes, the index file must contain:
1. The sequence number (row number of the original distance matrix, or position of the node in the list of nodes) of the node.
  2. The ID number of the node.
  3. Any integer value. } These values are read
  4. Any integer value. } but ignored.



5. A 1 if the node is considered to be inside the study region; a 0 if it is considered to be outside.
  6. The number of distances in the first distance class; 0 if the node is not a candidate.
  7. The number of distances in the second distance class; 0 if the node is not a candidate.
- Continue for as many distance classes as the file contains, to a maximum of eleven classes.

The information in the index file must be provided in the order above. If your index file is unformatted, and you are running the program under IBM Fortran, items 1, 2, 3, and 4 must be four-byte integers and the remaining items must be two-byte integers.

4.3 For each candidate node in the list of nodes, the distance file must contain:

1. The sequence number (row number of the original distance matrix, or position of the candidate in the list of nodes) of the node.
  2. The diagonal element of the original distance matrix for this candidate.
  3. The row number of the node nearest to this candidate.
  4. The distance to the node in 3 above.
  5. The row number of the node second nearest to this candidate.
  6. The distance to the node in 5 above.
- Continue for as many nodes as there are within the string.

The information in each string must appear in the order above.

There must not be any string in the distance file if a node is not to be a candidate node in the database.

If your distance file is unformatted and you are running the program under IBM Fortran, all items in the string must be two-byte integers.

4.4 The program reads the index and distance files in alternating fashion starting with the index for the first node in the list of nodes, followed by the distance string for the first node (if the node is a candidate), followed by the index for the second node in the list of nodes, and so forth. If both files are read from the same input unit, as from punched cards, the data on that unit must be organized in this same fashion, as:



index for first node  
string for first node  
index for second node  
string for second node

and so forth, with strings omitted for nodes that are not candidates. If the files are on separate input units, the elements of the files need only be in the order of the list of nodes.

- 4.9 If your nodes have equal populations (2.4), skip to 6.9. If your population file is unformatted data (2.9), skip to 6.0.

5.0 FORMAT TO READ NODE POPULATION FILE (optional).

On the next card, punch a Fortran format to read one card of the population file. The format must specify integer fields.

For example, the population file might be punched one population to a card, with the ID number punched in columns 1-10 and the population in columns 11-20. The format for this deck would be (2I10). If the populations were punched four to a card, with ten columns each for the ID number and population, the format for the file would be (8I10) or (4(2I10)).

6.0 NODE POPULATION FILE (optional).

- 6.1 The node population file contains the node ID numbers and populations punched together for each node, with the ID number preceding the node population. You may punch more than one ID and population per card, as long as you do not split a node ID and population between two cards, and as long as each card (except possibly the last in the deck) has the same number of node ID numbers and populations. If the population file is read as formatted data (2.9), the node ID numbers and populations need not be in the same order as the node ID numbers in the index file (4.4). The program will place them in the proper order after it reads them.

If your node population file is read as formatted data (2.9), skip to 6.9.

- 6.2 When the population file is unformatted, the node ID numbers are read by the program but ignored, and the populations must be in the same order as the node ID numbers in the index file.



If you are running the program under IBM Fortran, all items in an unformatted node population file must be four-byte integers.

6.9 If your data base does not contain fixed centers (2.11), skip to 7.9.

7.0 LIST OF FIXED CENTER LOCATIONS (optional).

7.1 Fixed centers may only be located at candidate nodes. Using one card for each fixed center, punch the ID number of the candidate where it is located in columns 2-10. The ID numbers need not be in any particular order. Column 1 of each card must be blank. If fixed centers are used by the program RETRENCH to prepare the distance file for this data base, RETRENCH will punch the list of fixed centers on cards using the format required by ALLOC VI.

7.2 On the card following the last of the fixed center locations, punch a 9 in column 1 and leave the rest of the card blank. A sample list of three fixed centers located at nodes 1005, 1009, and 1021 would be:

```
    1005  
    1009  
    1021  
9
```

7.9 If your data base does not have a file of second factor values (2.12), skip to 8.9.

8.0 SECOND FACTOR FILE (optional).

8.1 Second factor values are punched in fields of six columns, thirteen fields per card. Each field corresponds to a node in the list of nodes. The first field on the first card corresponds to the first node in the list, the second to the second node, the first field of the second card to the fourteenth node, etc. If a node is not a candidate node, its field in the second factor file may be left blank.

Second factor values must lie within the range from zero to one, inclusive. They are real values, not necessarily integers. If the second factor values are punched without a decimal point, the decimal point is assumed to fall between the second and third column of each field. If the values are punched with a decimal point, the decimal point may fall anywhere in the field.



- 8.9 This completes the p-median data base for a series of problems.

#### Punching the Problem Definition Cards

Each problem definition consists of one control card and as many as five decks of cards. The five decks contain: parameters for the trade-off algorithm; location constraints; a list of centers to serve as a starting solution; maximum distance constraints; and information to control the add algorithm. Only the control card and the list of centers are necessary to define a problem. Each deck begins on a new card. Figures V-2 and V-3 illustrate two problem definitions.

As many as one hundred problems may be defined on each run of ALLOC VI by adding as many problem definitions to the input deck as desired.

#### 9.0 PROBLEM DEFINITION CONTROL CARD (required).

9.1 In columns 1-5, punch the number of centers in the starting solution, including any fixed centers (7.0). For example, if the data base contains three fixed centers and you want to locate fourteen additional centers, punch 17 in columns 4 and 5.

9.2 Column 10 controls the algorithm to be used in solving the problem, and its value appears as MALG on the printed output. If you want the program to compute and print information about the starting solution, but do not want an algorithm to try to improve it, skip to 9.3. Otherwise, in column 10, punch the number from the following list that corresponds to the algorithm you wish to use.

- 1 Teitz and Bart algorithm
- 2 Hillsman-Rushton algorithm, phase 1 only
- 3 Hillsman-Rushton algorithm, phase 2 only
- 4 Hillsman-Rushton algorithm, complete
- 5 Add algorithm or add selected centers
- 6 Trade-off algorithm

The value of MALG will appear on the header record for machine-readable output.

9.3 Column 15 controls the use of location constraints, and its value appears as ICON on the printed output. If you want to clear an existing location constraint,



FIGURE V-2 Sample Problem Definition for ALLOC VI,  
Showing Use of Add Algorithm

114

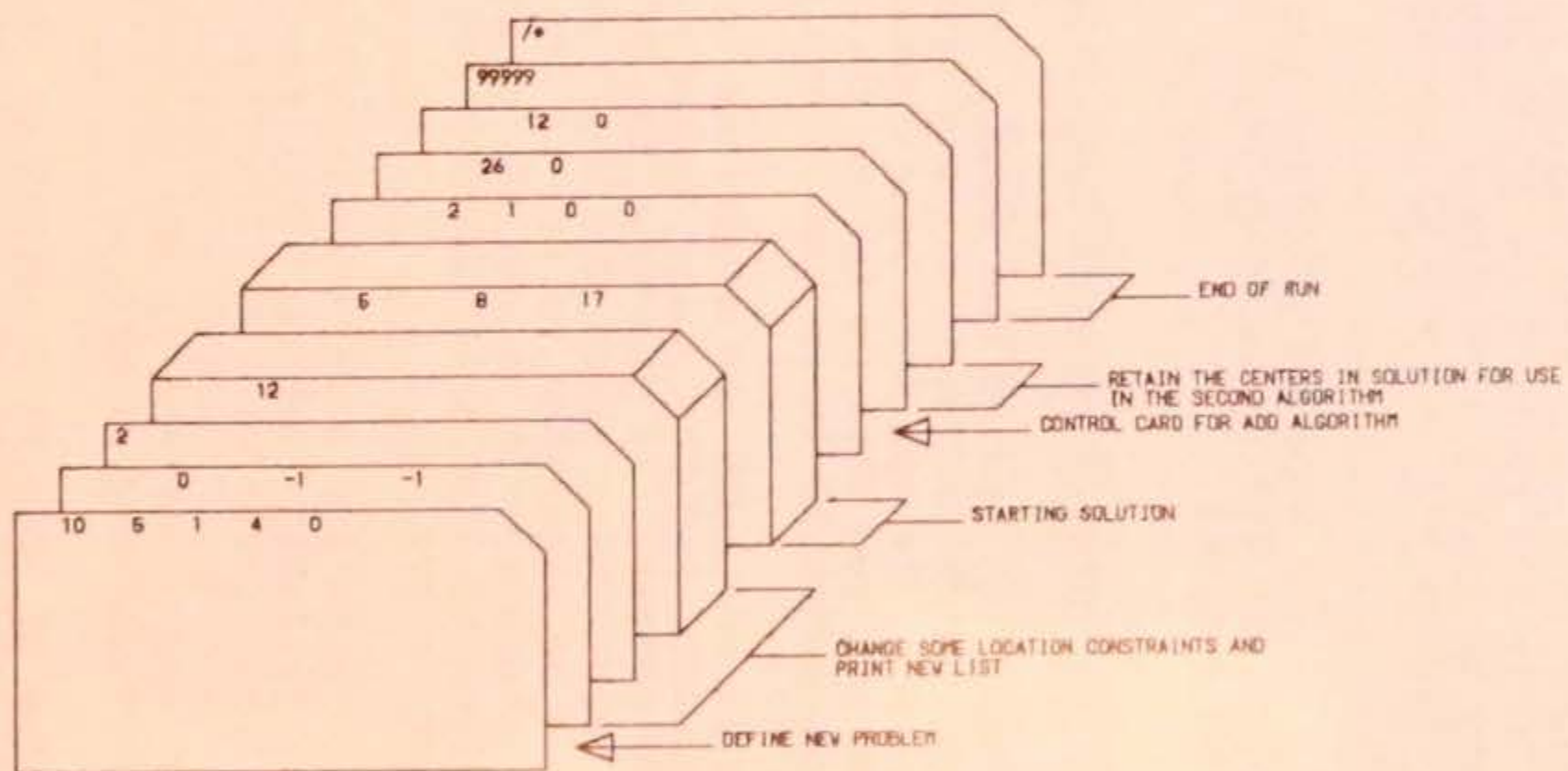
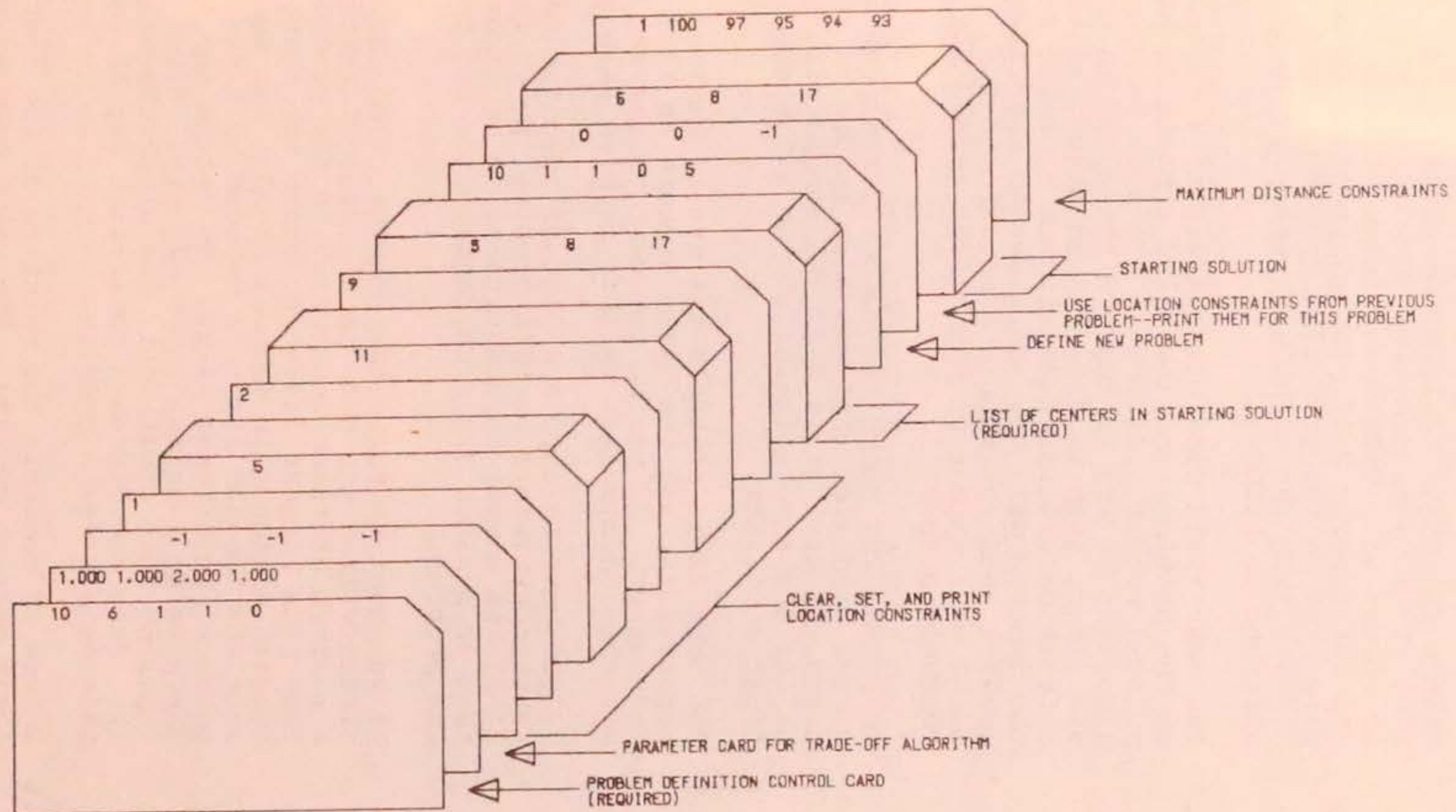




FIGURE V-3 Sample Problem Definitions for ALLOC VI, Showing Use of Trade-off Algorithm and Location Constraints





change any existing location constraints, or print a list of the nodes that have location constraints, punch 1 in column 15. Otherwise, leave columns 11-15 blank and the program will use any location constraints that were used on the preceding problem. Note that this differs from the method of repeating location constraints in ALLOC V.

- 9.4 The next field controls the use of a second algorithm on the problem, and its value appears as MALG2 on the printed output. If you do not want to use a second algorithm to try to improve the solution obtained by the first, skip to 9.5. In addition, if the problem definition will include maximum distance constraints, the program will not use a second algorithm on the problem, and you should also skip to 9.5. The value of MALG2 will appear on the header record for machine-readable output.

In column 20, punch the number corresponding to the second algorithm that you wish to use on the problem, using the list at (9.2). The second algorithm will use as its starting solution the solution obtained by the first algorithm.

- 9.5 The next field controls the use of maximum distance constraints, and its value appears as KRIT on the printed output. If you do not want to use any maximum distance constraints on this problem skip to 9.6.

If you want to impose a list of specific maximum distance constraints on this problem, punch in columns 24-25 the number of constraints to be imposed. No more than 30 maximum distance constraints may be imposed on one problem.

If you want to solve the problem, find the longest distance from any node to its nearest center, and impose a maximum distance constraint on the problem just less than this longest distance, punch the change in the longest distance in columns 21-25. For example, if you want the longest distance reduced by 1 distance unit, punch -1 in columns 24-25. To reduce the longest distance by 2 units, punch -2, etc. After computing and imposing this constraint, the algorithm will try to meet it. Only one distance constraint may be computed in this manner for a problem.

- 9.6 Column 30 controls the printing of the list of nodes and their nearest centers. The value punched in column 30 appears as NMAP on the printed output. If you want the



program to print the list of nodes and their nearest centers, punch a 1 in column 30. Otherwise, leave columns 26-30 blank.

- 9.7 Column 35 controls the printing of the list of nodes in each center's service area, and its value appears as MMAP on the printed output. If you want this information printed, punch a 1 in column 35. Otherwise, leave columns 31-35 blank.
- 9.8 Column 40 controls the production of machine-readable output for this problem, and its value appears as MACH on the printed output. If you do not want any machine-readable output from this problem, skip to 9.9.

If you want only the problem header card, number of centers, and list of centers written in machine-readable form, punch a 1 in column 40.

If in addition to the problem header card, number of centers, and list of centers you want additional information in machine-readable form, punch a 2 in column 40. If you use MACH=2, it is recommended that you specify output unit 7 as a tape or disk data set, rather than as punched cards. The value of MACH will appear on the header record for machine-readable output.

- 9.9 If you have a value of aggregate (not average) distance with which you wish to compare the aggregate distance for this problem, punch the value you wish to compare in columns 41-50. The program will make the comparison for the starting solution and for the final solution to each problem. The comparison is made as the ratio of your aggregate distance value to the aggregate distance of the current solution to the problem, times 100.
- 9.10 If you are not using the trade-off algorithm on this problem, skip to 10.9.

#### 10.0 PARAMETERS FOR TRADE-OFF ALGORITHM (optional).

- 10.1 Punch the value of parameter a of the trade-off function (Figure II-3) in columns 1-6, with a decimal point in column 3.
- 10.2 Punch the value of parameter b of the trade-off function in columns 7-12, with a decimal point in column 9.
- 10.3 Punch the value of parameter c of the trade-off function in columns 13-18, with a decimal point in column 15.



- 10.4 Punch the value of parameter  $d$  of the trade-off function in columns 19-24, with a decimal point in column 21.
- 10.9 If you are not clearing, changing or printing any location constraints on this problem, skip to 12.0.
- 11.0 LOCATION CONSTRAINTS DECK (optional).
- 11.1 The first card of the location constraints deck is a control card. This card specifies whether or not the program is to clear existing location constraints; change existing location constraints or impose new ones; or print lists of the candidate nodes having location constraints. These operations will be performed before solving the problem. You may request any combination of the three operations without having to request all of them, but you may request all three if you wish. An example of a location constraints deck appears in 11.7 below.
- 11.2 If you want to clear the location constraints from all nodes before doing anything else, punch -1 in columns 9-10 of the control card. Otherwise, leave columns 1-10 of the control card blank.
- 11.3 If you want to change the current location constraints for any candidates, or if you have cleared the existing location constraints (11.2) and wish to impose new ones for this problem, punch -1 in columns 19-20 of the control card. Otherwise, leave columns 11-20 of the control card blank.
- 11.4 If you want the program to print a list of the candidate nodes having location constraints for this problem, punch -1 in columns 29-30 of the control card. Otherwise, leave columns 21-30 of the control card blank.

If you are not changing location constraints or imposing new ones for this problem (11.3), skip to 12.0.

- 11.5 Location constraints are changed, or new ones imposed, by specifying the type of constraint and following it with a list of the ID numbers of the candidates to receive that constraint. Location constraints types are identified by a 0, 1, or 2.

A constraint of 0 clears an existing constraint from a candidate node, and permits an algorithm to move a center to or from any node having this type of constraint. All candidates are given a location constraint of 0



when the data base is set up, and all candidates (except those having fixed centers) are given a location constraint of 0 when location constraints are cleared (11.2).

A location constraint of 1 will prevent an algorithm from moving a center from a candidate, and a location constraint of 2 will prevent an algorithm from moving a center to a candidate. (Note that this is the opposite of ALLOC V's use of location constraints.)

- 11.6 For each group of candidates to receive a constraint, punch the type of constraint in column 1 of the card, and punch the ID number of each candidate to receive the constraint in columns 2-10 of a separate card. You may use as many groups of candidate nodes as you wish. In column 1 of the card following the last ID number of the last group, punch a 9.
- 11.7 As an example, assume that nodes with ID numbers 7, 13, and 48 already have location constraints of type 1 from a previous problem definition, and that these are the only nodes that had constraints in that problem. A deck to change the constraints might be

		-1	-1
0			
	7		
1			
	29		
	6		
2			
	13		
	17		
9			

This deck will clear the constraint from node 7; impose a constraint of type 1 on nodes 6 and 29; replace the constraint of type 1 on node 13 with a constraint of type 2; and impose a constraint of type 2 on node 17. After imposing these location constraints, the program will print a list of candidates having constraint type 0 (7 and, in this example, all candidates except 6, 13, 17, 29, and 48), constraint type 1 (6, 29, and 48), and constraint type 2 (13 and 17).

## 12.0 STARTING SOLUTION (required).

- 12.1 In fields of ten columns, punch the ID numbers of your initial center locations, except for fixed centers.



Thus, if the data base contains three fixed centers, and you wish to find locations for an additional fourteen centers, punch only the ID numbers for the fourteen. If, as in this case, you must punch locations for more than eight centers, punch the ID of the ninth location in the first ten columns of a second card. Use as many cards as necessary. The ID numbers may be punched in any order.

Note that your initial center locations must be candidate nodes and must not contain ID numbers for fixed centers.

If you are using the trade-off algorithm to solve this problem (9.2), the starting solution should contain the ID numbers of the candidates that have the highest suitability or second factor values.

12.9 If you are not imposing a set of specific distance constraints on this problem (9.5), skip to 13.9.

13.0 MAXIMUM DISTANCE CONSTRAINTS (optional).

13.1 If you want to impose the first maximum distance constraint before the algorithm tries to solve the problem, leave the first five columns of this deck blank. Otherwise, punch 1 in column 5 to have the algorithm solve the problem without any maximum distance constraint before it imposes the first one.

13.2 In columns 6-10, punch the first maximum distance constraint to be imposed on this problem. In succeeding fields of five columns, punch any additional maximum distance constraints, in order of decreasing length. If more than fifteen constraints are to be imposed, punch the sixteenth constraint in columns 1-5 of a second card and continue the rest of the constraints on that card.

13.9 If you are not using the add algorithm on this problem, skip to 14.8.

14.0 INPUT TO THE ADD ALGORITHM (optional).

Input to the add algorithm consists of a control card, followed by an optional list of locations where centers are to be added.

14.1 In columns 1-10 of the control card for the add algorithm, punch the number of locations at which the algorithm is to add centers.



- 14.2 This field controls the way that locations are to be chosen for the additional centers. If you want to add centers at specific candidate nodes, punch a 1 in column 15. Otherwise, leave columns 11-15 blank to have the algorithm find the best locations, in sequence, for the centers.
- 14.3 This field controls the output to be printed after all centers have been added. To suppress the standard output (list of centers, summary statistics and, optionally, list of nodes, list of service areas, and machine-readable output from 9.6-9.8) punch a 1 in column 20. Otherwise, leave columns 16-20 blank to have the standard output printed.
- 14.4 This field controls whether the program will punch the list of nodes in the service area of each center as it is added. To have this list punched, punch a 1 in column 25. Otherwise, leave columns 21-25 blank.
- 14.5 If you have directed the algorithm to find the best locations for the new centers (14.2), skip to 14.8. Otherwise, you must prepare a list of candidate nodes where centers are to be added. This list of candidates consists of one card per candidate, punched in the following manner.
- 14.6 In columns 2-10, punch the ID number of the candidate where the center is to be added. If this center is to be dropped after information about its service area is compiled, printed and, optionally, punched (14.4), punch a 1 in column 15. Otherwise, leave columns 11-15 blank and the center will remain in the solution for this problem unless moved by a second algorithm.

If you request the algorithm to add a center at a candidate that already has a center, or at a node that is not a candidate, or at a candidate that has a location constraint that prevents it from receiving a center (11.5), the program will print a message to this effect, ignore your request, and proceed to the next candidate in the list.

The following input to the add algorithm will add centers at five specified locations, print the standard output afterward, and produce machine-readable output as centers are added.



5	1	1
17		
4		
22	1	
23		
2		

Centers will be added at nodes 17, 4, and 22. After printing and punching summary information about the center added at node 22, the algorithm will remove the center from 22 and add centers at nodes 23 and 2. The solution to the problem at the end of the algorithm will have four more centers than when the algorithm began, and it is this solution that the program will prepare its standard output for.

- 14.8 This completes the definition of a problem. The control card to define an additional problem would be placed immediately behind the last card for this problem.
- 14.9 Following the last card for the last problem definition, place a card containing 99999 punched in columns 1-5.

#### Error Conditions

The following conditions will produce an error message and stop the program.

1. List of nodes (2.1), size of distance file (4.3), or length of longest distance string exceeds program dimensions.
2. End-of-file while reading index file, distance file, population file or second factor file in data base, or while reading starting solution or maximum distance constraints for a problem definition.
3. Input error (IBM ERR=) while reading index file, distance file, or second factor in data base, or while reading starting solution for a problem definition.
4. First field of index file does not correspond to the position of the node in the list of nodes (4.2).
5. First element of a distance string is not the diagonal element from the candidate's column of the original distance matrix (4.3).
6. Data base has no nodes inside the study region (4.2).



7. Duplicate ID number in a formatted population deck (6.0).
8. ID number in formatted population deck (6.0) or starting solution (12.0) does not match any ID number in the list of nodes.
9. Row subscript in distance file is less than 1 or greater than N (2.1). Usually caused by reversing the positions of distances and row subscripts in the distance file.
10. Duplicate ID number in the list of fixed centers (7.0).
11. The product of the largest weighted distance and the length of the longest distance string exceeds the largest integer that can be stored.
12. The starting solution (12.0) contains less than one center, fewer centers than there are fixed centers, or more centers than the program can store. Usually caused by a misplaced card.
13. Invalid algorithm code (9.2). Usually caused by a misplaced card.
14. Invalid location constraints control card (11.1-11.4).
15. Maximum distance constraints (13.0) that are not in decreasing order.
16. Invalid location constraint type (11.5).
17. Attempt to impose a location constraint on a node that is not a candidate in the data base.
18. Specifying a location constraint type (11.5) without a list of nodes to be constrained.
19. Attempt to impose a location constraint on a candidate that has a fixed center.

The following conditions are less serious. They will produce an error message and the following actions.

1. Duplicate ID number in the starting solution (12.0). Start one center at the node with the duplicated ID. Then, search the list of nodes for the first candidate that does not have a center and that has not been constrained out of the problem; place the second center there.



2. Request for the add algorithm (14.0) to add a center at a node that is not a candidate, or that has a location constraint of 2 (11.5). Ignore the request.
3. Request for the add algorithm to add a center at a candidate that already has a center. Ignore the request.
4. Request for the add algorithm to add a center at a node that does not appear in the list of nodes. Ignore the request.

#### Program Dimensions

The array dimensions in the listing of ALLOC VI permit a data base of 150 nodes, a distance file of 10,000 distances, a distance string of 150 distances, and a problem definition of 150 centers. The comment cards at the beginning of the program give directions for changing the dimensions of the program arrays.



FIGURE V-4 Sample Output from ALLOC VI

```

-----
| PROGRAM ALLOC VI |
| WRITTEN BY EDWARD L. HILLSMAN |
| DEPARTMENT OF GEOGRAPHY |
| THE UNIVERSITY OF IOWA |
| IOWA CITY, IOWA 52242 USA |
| COPYRIGHT C 1977 EDWARD L. HILLSMAN |
-----

RUN TITLE/ SAMPLE RUN FOR USE IN MONOGRAPH NO. 7 7
NUMBER OF NODES IN FILES 49
PLUS ONE DUMMY NODE WITH ID OF 0

MAXIMUM DISTANCE TO BE SAVED 100
NUMBER OF DISTANCE CLASSES IN INDEX FILE 1
READ INDEX FILE FROM UNIT 5
READ DISTANCE FILE FROM UNIT 5
READ PROBLEM DEFINITIONS FROM UNIT 5

INDEX AND DISTANCE FILES ARE FORMATTED
READ FORMATS FROM UNIT 5

READ WEIGHTS FORMAT FROM UNIT 5
READ FORMATTED WEIGHTS FROM UNIT 5
DIVIDE ALL WEIGHTS BY 1

NO FIXED CENTERS

READ SECOND FACTOR VALUES FROM UNIT 5
-----

NUMBER OF DISTANCES STORED IS 611
MAXIMUM PERMITTED IS 10000
LENGTH OF LONGEST STRING IS 20

NUMBER OF NODES INSIDE STUDY REGION IS 49

TOTAL WEIGHT AFTER SCALING IS 69962
TOTAL INSIDE IS 69962

PROVIDE INFEASIBLE SERVICE AT A COST OF 660521
-----

```



LIST OF NODE ID NUMBERS AND POPULATIONS

1	2811	2	592	3	4027	4	1536	5	1241
6	845	7	422	8	819	9	2981	10	2767
11	933	12	3356	13	767	14	904	15	734
16	2882	17	1337	18	1905	19	1198	20	546
21	133	22	857	23	1087	24	1245	25	813
26	755	27	365	28	2328	29	1119	30	1755
31	1835	32	1205	33	1070	34	393	35	1607
36	1095	37	764	38	739	39	932	40	1320
41	973	42	1803	43	1736	44	6740	45	1086
46	1144	47	733	48	1198	49	529		

PROBLEM NUMBER 1

LOCATE 10 CENTERS

MALG ICON MALG2 KRIT NMAP MMAP MACH  
 6 0 1 0 0 1 0  
 MEASURE EFFICIENCY AGAINST A VALUE OF 1561823

PARAMETERS FOR TRADE-OFF FUNCTION

A= 1.000 B= 1.000 C= 2.000 D= 1.000

LIST OF CENTERS

CENTER	WEIGHT	DISTANCE * WEIGHT	AVERAGE DISTANCE	COST IF DROPPED	SECOND FACTOR
3	9765	273772	28.036	929819	0.6023
10	10325	457089	44.270	849204	0.4922
16	4553	56047	12.310	224199	0.4964
1	5421	175596	32.392	739337	0.4793
9	5485	194012	35.371	686900	0.5230
12	3356	0	0.0	140952	0.5149
28	2328	0	0.0	83808	0.5587
44	12686	147090	11.595	479249	0.9426
43	7658	292629	38.212	130092	0.4603
42	4008	60687	15.141	157251	0.4503

FOR THE LIST OF CENTERS ABOVE:

TOTAL WEIGHTED DISTANCE IS 5620048 INSIDE 5620048  
 AVERAGE DISTANCE TO NEAREST CENTER IS 80.330 INSIDE 80.330



LIST OF TRADE AREAS (NODE ID, WEIGHT, AND DISTANCE TO CENTER, WHICH IS FIRST ID IN EACH AREA)

3 4027 0	47 733 25	5 1241 36	30 1755 37	37 764 54
24 1245 84				
10 2767 0	25 813 31	20 546 38	19 1198 47	39 932 54
26 755 67	4 1536 67	11 933 74	6 845 97	
16 2862 0	13 767 33	14 904 34		
1 2811 0	22 857 40	2 592 54	7 422 84	38 739 100
9 2981 0	27 365 43	8 819 63	40 1320 96	
12 3356 0				
28 2328 0				
44 6740 0	35 1607 12	48 1198 20	33 1070 31	17 1337 32
15 734 38				
43 1736 0	31 1835 29	23 1087 47	18 1905 50	36 1095 85
42 1803 0	45 1086 26	29 1119 29		

127

THE FOLLOWING 6 NODES, WITH A TOTAL WEIGHT OF 4377 CANNOT BE SERVED WITHIN THE MAXIMUM DISTANCE

21	133	32	1205	34	393	41	973	46	1144	49	529
----	-----	----	------	----	-----	----	-----	----	------	----	-----

SUMMARY STATISTICS

TOTAL WEIGHTED DISTANCE IS 5620048 INSIDE 5620048  
 AVERAGE DISTANCE TO NEAREST CENTER IS 80.330 INSIDE 80.330

OVER ENTIRE PROBLEM, MAXIMUM DISTANCE TRAVELED IS 100 FROM NODE 38 TO CENTER 1  
 INSIDE STUDY REGION, MAXIMUM DISTANCE TRAVELED IS 100 FROM NODE 38 TO CENTER 1

AVERAGE VALUE OF SECOND FACTOR IS 0.5520

MOST EXPENDABLE CENTER IS 28  
 WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY 83808 IF DROPPED WITHOUT REPLACEMENT

EFFICIENCY OF CURRENT SOLUTION COMPARED WITH INPUT VALUE IS 27.7902

PERCENT CHANGE IN OBJECTIVE FUNCTION FROM INITIAL LIST OF CENTERS IS 0.0  
 FROM LAST PRINTING IS 0.0

-----



START TRADE-OFF ALGORITHM (MODIFIED TEITZ AND BARTI)

OLD CENTER	COST IF DROPPED	NEW CENTER	TOTAL COST	NET CHANGE	PERCENT CHANGE	SECOND FACTOR
42	157251	18	2607672	3012376	53.6005	0.5497
28	83808	19	2526441	81231	3.1151	0.5189
19	165039	21	1895237	631204	24.9839	0.4962
18	119514	29	1864426	30811	1.6257	0.4936
43	159417	31	1834425	30001	1.6091	0.4871
21	3851831	32	1792153	42272	2.3044	0.5085
29	150325	40	1725296	66857	3.7305	0.4978
9	132624	42	1700669	24627	1.4274	0.4906

END CYCLE 1  
 CHANGES = 8  
 END CYCLE 2  
 CHANGES = 0

END TRADE-OFF ALGORITHM

LIST OF CENTERS

CENTER	WEIGHT	DISTANCE * WEIGHT	AVERAGE DISTANCE	COST IF DROPPED	SECOND FACTOR
3	10906	337291	30.927	1557436	0.6023
10	8547	306082	35.812	288969	0.4922
16	4553	56047	12.310	224199	0.4964
1	4682	101696	21.721	882611	0.4793
42	4908	60687	15.141	157251	0.4503
12	9030	246095	27.253	208068	0.5149
32	6211	241473	38.878	3894103	0.2370
44	12686	147090	11.595	535107	0.9426
31	5422	92066	16.980	189418	0.3952
40	3917	112142	28.630	771458	0.2956

128

FOR THE LIST OF CENTERS ABOVE:

TOTAL WEIGHTED DISTANCE IS 1700669 INSIDE 1700669

AVERAGE DISTANCE TO NEAREST CENTER IS 24.308 INSIDE 24.308

LIST OF TRADE AREAS (NODE ID, WEIGHT, AND DISTANCE TO CENTER, WHICH IS FIRST ID IN EACH AREA)

3 4027 0	47 733 25	5 1241 36	30 1755 37	18 1905 55
24 1245 84				
10 2767 0	25 813 31	20 546 38	19 1198 47	39 932 54
26 755 67	4 1536 67			
16 2882 0	13 767 33	14 904 34		
1 2811 0	22 857 40	2 592 54	7 422 84	



42 1803	0	45 1086	26	29 1119	29						
12 3356	0	28 2328	42	9 2981	44	27 365	47				
32 1205	0	41 573	14	34 393	34	36 1095	43	21 133	54		
46 1144	56	38 739	75	49 529	77						
44 6740	0	35 1607	12	48 1198	20	33 1070	31	17 1337	32		
15 734	38										
31 1835	0	23 1087	18	43 1736	29	37 764	29				
40 1320	0	8 819	33	6 845	40	11 933	55				

SUMMARY STATISTICS

TOTAL WEIGHTED DISTANCE IS 1700669      INSIDE 1700669  
 AVERAGE DISTANCE TO NEAREST CENTER IS 24.308      INSIDE 24.308

OVER ENTIRE PROBLEM, MAXIMUM DISTANCE TRAVELED IS 84 FROM NODE      7 TO CENTER      1  
 INSIDE STUDY REGION, MAXIMUM DISTANCE TRAVELED IS 84 FROM NODE      7 TO CENTER      1

AVERAGE VALUE OF SECOND FACTOR IS 0.4906

MUST EXPENDABLE CENTER IS 42  
 WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY 157251 IF DROPPED WITHOUT REPLACEMENT

EFFICIENCY OF CURRENT SOLUTION COMPARED WITH INPUT VALUE IS 91.8358

PERCENT CHANGE IN OBJECTIVE FUNCTION FROM INITIAL LIST OF CENTERS IS 69.7392  
 FROM LAST PRINTING IS 69.7392

---

CENTER LOCATIONS AT BEGINNING OF PROBLEM 1

3	10	16	1	9	12	28	44	43	42
---	----	----	---	---	----	----	----	----	----

LOCATIONS OF CENTERS AT END OF ALGORITHM

3	10	16	1	42	12	32	44	31	40
---	----	----	---	----	----	----	----	----	----

USE SECOND ALGORITHM

START TEITZ AND BART ALGORITHM



OLD CENTER	COST IF DROPPED	NEW CENTER	TOTAL COST	NET CHANGE	PERCENT CHANGE	SECOND FACTOR
40	771458	11	1670276	30393	1.7871	0.4727
32	3894103	34	1589022	81254	4.8647	0.4532
42	157251	45	1561823	27199	1.7117	0.4267

END CYCLE 1  
 CHANGES = 3  
 END CYCLE 2  
 CHANGES = 0

END TEITZ AND BART ALGORITHM

LIST OF CENTERS

CENTER	WEIGHT	DISTANCE * WEIGHT	AVERAGE DISTANCE	COST IF DROPPED	SECOND FACTOR
3	9661	232711	24.088	973025	0.6023
10	4126	45951	11.137	188405	0.4922
16	4553	56047	12.310	220600	0.4964
1	4260	66248	15.551	879657	0.4793
45	6299	186985	29.685	184450	0.1855
12	9030	246095	27.253	208068	0.5149
34	7878	300247	38.112	3975357	0.0422
44	12686	147090	11.595	535107	0.9426
31	5422	92066	16.980	189418	0.3952
11	6047	188383	31.153	801851	0.1169

FOR THE LIST OF CENTERS ABOVE:

TOTAL WEIGHTED DISTANCE IS 1561823 INSIDE 1561823  
 AVERAGE DISTANCE TO NEAREST CENTER IS 22.324 INSIDE 22.324

LIST OF TRADE AREAS (NODE ID, WEIGHT, AND DISTANCE TO CENTER, WHICH IS FIRST ID IN EACH AREA)

3 4027	0	47 733	25	5 1241	36	30 1755	37	18 1905	55
10 2767	0	25 813	31	20 546	38				
16 2882	0	13 767	33	14 904	34				
1 2811	0	22 857	40	2 592	54				
45 1086	0	29 1119	18	42 1803	26	26 755	47	4 1536	55
12 3356	0	28 2328	42	9 2981	44	27 365	47		
34 393	0	41 573	20	21 133	20	46 1144	22	36 1095	28
32 1205	34	49 529	43	38 739	60	24 1245	66	7 422	76
44 6740	0	35 1607	12	48 1198	20	33 1070	31	17 1337	32
15 734	38								
31 1835	0	23 1087	18	43 1736	29	37 764	29		
11 933	0	39 932	20	6 845	30	8 819	35	19 1198	36
40 1320	55								

130







THE FOLLOWING 48 CANDIDATES HAVE A LOCATION CONSTRAINT OF 0

1	2	3	4	5	6	7	8	9	10
12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41
42	43	44	45	46	47	48	49		

THE FOLLOWING 1 CANDIDATES HAVE A LOCATION CONSTRAINT OF 1

11

END PROCESSING OF LOCATION CONSTRAINTS

LIST OF CENTERS CENTER	WEIGHT	DISTANCE * WEIGHT	AVERAGE DISTANCE	COST IF DROPPED	SECOND FACTOR
11	3609	98896	27.403	80366	0.1169
9	26766	1477231	55.191	3672705	0.5230
8	2504	58525	23.373	38635	0.1378
7	12269	874270	71.258	7051982	0.0653
6	845	0	0.0	25350	0.1456

FOR THE LIST OF CENTERS ABOVE:

TOTAL WEIGHTED DISTANCE IS 13737779 INSIDE 13737779  
 AVERAGE DISTANCE TO NEAREST CENTER IS 196.361 INSIDE 196.361

LIST OF NODES

NODE	CENTER	WEIGHT	DIST	NODE	CENTER	WEIGHT	DIST	NODE	CENTER	WEIGHT	DIST				
11	1	7	2811	84	11	2	7	592	34	11	3	0	4027	-1	11
11	4	0	1536	-1	11	5	0	1241	-1	11	6	6	845	0	11
11	7	7	422	0	11	8	8	819	0	11	9	9	2981	0	11
11	10	9	2767	53	11	11	11	933	0	11	12	9	3356	44	11
11	13	0	767	-1	11	14	0	504	-1	11	15	9	734	93	11
11	16	0	2882	-1	11	17	9	1337	50	11	18	7	1905	83	11
11	19	11	1198	36	11	20	11	546	68	11	21	7	133	96	11
11	22	7	857	44	11	23	0	1087	-1	11	24	0	1245	-1	11
11	25	9	813	84	11	26	0	755	-1	11	27	8	365	41	11
11	28	9	2328	86	11	29	0	1119	-1	11	30	0	1755	-1	11
11	31	9	1835	91	11	32	7	1205	91	11	33	9	1070	86	11
11	34	7	393	76	11	35	9	1607	87	11	36	7	1095	48	11
11	37	0	764	-1	11	38	7	739	16	11	39	11	932	20	11
11	40	8	1320	33	11	41	7	573	96	11	42	0	1803	-1	11
11	43	0	1736	-1	11	44	9	6740	55	11	45	0	1086	-1	11
11	46	7	1144	98	11	47	0	733	-1	11	48	9	1198	35	11
11	49	0	529	-1	11										



THE FOLLOWING 17 NODES, WITH A TOTAL WEIGHT OF 23969 CANNOT BE SERVED WITHIN THE MAXIMUM DISTANCE

3	4027	4	1536	5	1241	13	767	14	904	16	2892
23	1087	24	1245	26	755	29	1119	30	1755	37	766
42	1803	43	1736	45	1086	47	733	49	529		

SUMMARY STATISTICS

TOTAL WEIGHTED DISTANCE IS 1373779 INSIDE 1373779  
 AVERAGE DISTANCE TO NEAREST CENTER IS 196.361 INSIDE 196.361

OVER ENTIRE PROBLEM, MAXIMUM DISTANCE TRAVELED IS 98 FROM NODE 46 TO CENTER 7  
 INSIDE STUDY REGION, MAXIMUM DISTANCE TRAVELED IS 98 FROM NODE 46 TO CENTER 7

AVERAGE VALUE OF SECOND FACTOR IS 0.1977

MOST EXPENDABLE CENTER IS 6 25350 IF DROPPED WITHOUT REPLACEMENT  
 WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY

PERCENT CHANGE IN OBJECTIVE FUNCTION FROM INITIAL LIST OF CENTERS IS 0.0  
 FROM LAST PRINTING IS 0.0

START HILLSMAN-RUSHTON ALGORITHM, PHASE ONE  
 (REPLACE MOST EXPENDABLE CENTER UNTIL NO FURTHER IMPROVEMENT IS POSSIBLE)

OLD CENTER	COST IF DROPPED	NEW CENTER	TOTAL COST	NET CHANGE	PERCENT CHANGE	SECOND FACTOR
6	25350	3	8826420	4911359	35.7507	0.2891
8	59435	29	4042928	4783592	54.1963	0.3420
9	4529413	44	3672797	370031	9.1528	0.4259

END PHASE ONE  
 CHANGES = 3

THE MOST EXPENDABLE MOVABLE CENTER IS AT NODE 29

START HILLSMAN-RUSHTON ALGORITHM, PHASE TWO

OLD CENTER	COST IF DROPPED	NEW CENTER	TOTAL COST	NET CHANGE	PERCENT CHANGE	SECOND FACTOR
7	5563208	36	2876103	796694	21.6918	0.4534

END CYCLE 1  
 CHANGES = 1  
 END CYCLE 2  
 CHANGES = 0

END HILLSMAN-RUSHTON ALGORITHM, PHASE TWO

THE MOST EXPENDABLE MOVABLE CENTER IS AT NODE 29

LIST OF CENTERS



CENTER	WEIGHT	DISTANCE * WEIGHT	AVERAGE DISTANCE	COST IF DROPPED	SECOND FACTOR
11	6758	253251	36.397	3226852	0.1169
44	25414	964086	37.935	5423841	0.9426
29	10852	442141	40.743	2039647	0.4023
36	9987	448633	44.922	6359902	0.2030
3	16751	767992	45.848	3530930	0.6023

FOR THE LIST OF CENTERS ABOVE:

TOTAL WEIGHTED DISTANCE IS 2876103 INSIDE 2876103  
 AVERAGE DISTANCE TO NEAREST CENTER IS 41.109 INSIDE 41.109

LIST OF NODES

NODE	CENTER	WEIGHT	DIST	NODE	CENTER	WEIGHT	DIST	NODE	CENTER	WEIGHT	DIST				
11	1	44	2811	75	11	2	36	592	82	11	3	4027	0	11	
11	4	29	1536	73	11	5	3	1241	36	11	6	845	30	11	
11	7	36	422	48	11	8	11	819	35	11	9	2981	55	11	
11	10	44	2767	56	11	11	11	933	0	11	12	44	3356	73	11
11	13	29	767	67	11	14	29	904	31	11	15	44	734	38	11
11	16	29	2882	45	11	17	44	1337	32	11	18	36	1905	35	11
11	19	11	1198	36	11	20	11	546	68	11	21	36	133	48	11
11	22	36	857	92	11	23	3	1087	85	11	24	3	1245	84	11
11	25	44	813	52	11	26	29	755	65	11	27	11	365	76	11
11	28	3	2328	84	11	29	29	1119	0	11	30	3	1755	37	11
11	31	3	1835	67	11	32	36	1205	43	11	33	44	1070	31	11
11	34	36	393	28	11	35	44	1607	12	11	36	36	1095	0	11
11	37	3	764	54	11	38	36	739	32	11	39	11	932	20	11
11	40	11	1320	55	11	41	36	973	48	11	42	29	1803	29	11
11	43	3	1736	48	11	44	44	6740	0	11	45	29	1086	18	11
11	46	36	1144	50	11	47	3	733	25	11	48	44	1198	20	11
11	49	36	529	71	11										

SUMMARY STATISTICS

TOTAL WEIGHTED DISTANCE IS 2876103 INSIDE 2876103  
 AVERAGE DISTANCE TO NEAREST CENTER IS 41.109 INSIDE 41.109

OVER ENTIRE PROBLEM, MAXIMUM DISTANCE TRAVELED IS 92 FROM NODE 22 TO CENTER 36  
 INSIDE STUDY REGION, MAXIMUM DISTANCE TRAVELED IS 92 FROM NODE 22 TO CENTER 36

AVERAGE VALUE OF SECOND FACTOR IS 0.4534

MOST EXPENDABLE CENTER IS 29  
 WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY 2039647 IF DROPPED WITHOUT REPLACEMENT

PERCENT CHANGE IN OBJECTIVE FUNCTION FROM INITIAL LIST OF CENTERS IS 79.0643  
 FROM LAST PRINTING IS 79.0643



135

CENTER LOCATIONS AT BEGINNING OF PROBLEM 2

11 9 8 7 6

LOCATIONS OF CENTERS AT END OF ALGORITHM

11 44 29 36 3

USE SECOND ALGORITHM

START CENTER ADDING ROUTINE

A CENTER ADDED AT 28 WOULD COMMAND A TRADE AREA POPULATION OF 14602 FROM THE FOLLOWING 8 PLACES  
 AND WOULD REDUCE THE OBJECTIVE FUNCTION TO 2428480

28	2328	43	1736	12	3356	22	857	1	2811	2	592
31	1835	23	1087								

DROP THIS CENTER BEFORE CONTINUING THE ANALYSIS

A CENTER ADDED AT 34 WOULD COMMAND A TRADE AREA POPULATION OF 5622 FROM THE FOLLOWING 7 PLACES  
 AND WOULD REDUCE THE OBJECTIVE FUNCTION TO 2754032

34	393	41	973	21	133	46	1144	32	1205	49	529
24	1245										

RETAIN THIS CENTER IN FURTHER ANALYSIS

END CENTER ADDING ROUTINE

LIST OF CENTERS

CENTER	WEIGHT	DISTANCE * WEIGHT	AVERAGE DISTANCE	COST IF DROPPED	SECOND FACTOR
11	6958	253251	36.397	3223567	0.1169
44	25414	964086	37.935	2630214	0.9426
29	10852	442141	40.743	2039647	0.4023
36	5610	237967	42.418	47373	0.2030
3	15506	663412	42.784	1659863	0.6023
28	0	0	0.0	0	0.5587
34	5622	193175	34.361	122071	0.0422

FOR THE LIST OF CENTERS ABOVE:

TOTAL WEIGHTED DISTANCE IS 2754032 INSIDE 2754032  
 AVERAGE DISTANCE TO NEAREST CENTER IS 39.365 INSIDE 39.365

LIST OF NODES

NODE	CENTER	WEIGHT	DIST	NODE	CENTER	WEIGHT	DIST	NODE	CENTER	WEIGHT	DIST				
11	1	44	2811	75	11	2	36	592	82	11	3	4027	0	11	
11	4	29	1536	73	11	5	3	1241	36	11	6	845	30	11	
11	7	36	422	48	11	8	11	819	35	11	9	44	2981	55	11
11	10	44	2767	56	11	11	11	933	0	11	12	44	3356	73	11
11	13	29	767	67	11	14	29	504	31	11	15	44	734	38	11



11	16	29	2882	45	11	17	44	1337	32	11	18	36	1905	35	11
11	19	11	1198	36	11	20	11	546	68	11	21	34	133	20	11
11	22	36	857	92	11	23	3	1087	85	11	24	34	1245	66	11
11	25	44	813	52	11	26	29	755	65	11	27	11	365	76	11
11	28	3	2328	84	11	29	29	1119	0	11	30	3	1755	37	11
11	31	3	1835	67	11	32	34	1205	34	11	33	44	1070	31	11
11	34	34	393	0	11	35	44	1607	12	11	36	36	1095	0	11
11	37	3	764	54	11	38	36	739	32	11	39	11	932	20	11
11	40	11	1320	55	11	41	34	973	20	11	42	29	1803	29	11
11	43	3	1736	48	11	44	44	6740	0	11	45	29	1086	18	11
11	46	34	1144	22	11	47	3	733	25	11	48	44	1198	20	11
11	49	34	529	43	11										

SUMMARY STATISTICS

TOTAL WEIGHTED DISTANCE IS 2754032 INSIDE 2754032  
 AVERAGE DISTANCE TO NEAREST CENTER IS 39.365 INSIDE 39.365

OVER ENTIRE PROBLEM, MAXIMUM DISTANCE TRAVELED IS 92 FROM NODE 22 TO CENTER 36  
 INSIDE STUDY REGION, MAXIMUM DISTANCE TRAVELED IS 92 FROM NODE 22 TO CENTER 36

AVERAGE VALUE OF SECOND FACTOR IS 0.4097

MOST EXPENDABLE CENTER IS 28  
 WHICH WOULD INCREASE THE OBJECTIVE FUNCTION BY 0 IF DROPPED WITHOUT REPLACEMENT

PERCENT CHANGE IN OBJECTIVE FUNCTION FROM INITIAL LIST OF CENTERS IS 79.9525  
 FROM LAST PRINTING IS 4.2443

CENTER LOCATIONS AT BEGINNING OF PROBLEM 2

11 9 8 7 6

LOCATIONS OF CENTERS AT END OF ALGORITHM

11 44 29 36 3 28 34

END OF PROBLEM 2

OBJECTIVE FUNCTION AT START AND END OF EACH PROBLEM

1 5620048 1561823  
 2 13737779 2754032



## CHAPTER VI

### RETRENCH

The program RETRENCH was designed to read an index file and distance file, identify unnecessary distances, and prepare a new, shorter distance file and corresponding index file for use in ALLOC VI. RETRENCH considers four types of distances to be unnecessary. First, a distance may appear in the distance string of a node that is not a candidate. Second, a distance may be greater than some implicit maximum distance constraint. Third, a distance from a node to a candidate may be greater than the distance to a fixed center. Finally, a distance may be measured from a node that is to be ignored in an analysis.

RETRENCH was written to meet the needs of several specific analyses, including the study reported by Rushton et al. (1976) and several of that study's preliminary analyses. Because of its history, RETRENCH is somewhat awkward to explain and use. For example, the program was written for use with very large data bases. Unformatted files were more convenient and more efficient to use than formatted ones, because of their size. As a result, RETRENCH requires the index and distance files that it modifies to be unformatted. A complete rewriting of the program would simplify its description and use, but such a rewriting is beyond the scope of this monograph.

The following description of RETRENCH and its input data assumes familiarity with distance files, index files, the principles used to develop them, and the features of ALLOC VI. The only new principle involves the notion of ignoring a node, and the following paragraph explains this principle.

ALLOC VI locates centers to minimize the aggregate distance from all nodes in a data base to their nearest centers. If nodes outside a study region are placed in a data base, they will affect all solutions found by the algorithms in ALLOC VI. To prevent these nodes from affecting the solutions, the analyst could remove them from the data base altogether, but this might also require removing them from other files as well. These files, though not part of the p-median data base, may contain detailed census information or measures of economic activity that will be used to analyze a p-median solution or to conduct other analyses in a region. An alternative to removing nodes from all the files would be to remove them just from the distance strings in the distance file, yet leave them in the rest of the p-median data base. ALLOC VI assumes that



such nodes cannot be served within any maximum distance constraint, implicit or not, and its algorithms ignore them. The ignored nodes still appear in the printed and machine-readable output from ALLOC VI, and they can easily be cross-referenced with data in other files.

Since an ignored node causes all solutions in ALLOC VI to violate all maximum distance constraints, the use of such nodes prevents ALLOC VI from imposing more than one maximum distance constraint per problem definition. This loss in flexibility can be more than made up for by not having to recreate or reindex a large data base for a region, however.

### Program Features and Flow

RETRENCH consists of four sections. The first section reads data to determine the locations of candidates, fixed centers, and nodes to be ignored, as well as the size of the implicit maximum distance to be used in the new distance file. This section will be discussed in much more detail below. If fixed centers are used to exclude distances, the second section of RETRENCH reads the distance file and index file to find the distance from each node to its nearest fixed center. This operation is termed the first pass through the data. The third section of RETRENCH makes a second pass through the data, and this is where the program creates the new distance file and corresponding index file. Finally, the fourth section of RETRENCH writes out summary information about the new distance file and reformats some data for easier use by ALLOC VI.

#### Section 1: Defining How to Shorten the Distance File

RETRENCH uses three methods to determine which nodes are to be candidates and which are not. It can require each candidate to have some minimum population. It can require all candidates to lie within the study region boundary of a data base. Finally, it can permit the analyst to declare the candidacy status of individual nodes in the data base. This permits the analyst to override the results of the other two methods.

For most applications, it is probably simplest to use just one of the three methods to declare candidates. The three methods may be used in combination, but the order in which they operate is fixed by the program. Since the three methods may conflict for one or more nodes, the order in which the program conducts them determines the final status of each node. RETRENCH starts by assuming that the data base contains no candidates. It then reads the population of



each node and determines which nodes will be made candidates. By setting the minimum population sufficiently high or low, the analyst may make none or all of the nodes candidates with this operation. If nodes outside the study region are to be excluded from candidacy, the program then excludes them. Finally, RETRENCH reads any declarations of candidacy or exclusion supplied by the analyst and adjusts the list of candidates accordingly.

Specifying fixed centers is much simpler than deciding candidacy. If fixed centers are to be used, the program requires a simple list of the node ID numbers for these centers. If a fixed center occurs at a node that was not made a candidate earlier, RETRENCH adds the node to the list of candidates. Because the list of fixed centers must be prepared in a slightly different form from that needed by ALLOC VI, RETRENCH reformats the list of fixed centers and punches it in the form that ALLOC VI requires.

Finally, if any nodes are to be ignored in the new distance file, RETRENCH reads a list of these nodes and prepares to ignore them. If a fixed center appears in the list of nodes to be ignored, RETRENCH notes this with an error message and stops. If a candidate appears in the list of nodes to be ignored, the request to ignore will override candidacy, and the node will not appear as a candidate in the new files.

## Section 2: The First Pass

The first pass through the data has two purposes. The main purpose is to find the distance from each node to its nearest center. If fixed centers are not used to exclude distances, RETRENCH considers the first pass unnecessary and does not perform it.

The second purpose of the first pass is to determine that every node, other than those to be ignored, can be served from at least one candidate in the new distance file. RETRENCH assumes that the new data base should permit every node to be served, and it stops if it finds unservable nodes. The purpose of this check is to prevent RETRENCH from wasting time preparing a distance file that would be considered inadequate for most analyses. To prepare a distance file that will not allow service to some of the nodes, it is necessary to run RETRENCH a second time and direct it to ignore the unservable nodes identified during the first run of the program.

RETRENCH has an option to save the data from the first pass of one job submission and reread it during a later one.



The distance from each node to its nearest center is unaffected by the number and locations of candidate nodes, as long as the number and locations of fixed centers do not change. Similarly, the first pass data is gathered and used in a way that is independent of the implicit maximum distance constraint to be imposed on the new data base, as long as the length of this distance in the original distance file does not change. Thus, if two submissions of RETRENCH use the same fixed centers and modify the same input distance and index files, the first pass data from one submission could be saved and used in the other. This would reduce the amount of file reading required by the second submission and, therefore, the cost of creating a second new distance file.

### Section 3: The Second Pass

The third section of RETRENCH rewinds the index and distance files if a first pass was performed. It then reads the files, identifies unnecessary distances, computes a new index file, and writes the new index and distance files, unformatted, on the output units requested by the analyst. For simplicity, RETRENCH prepares the new index file with only a single distance class. RETRENCH makes a printed copy of the new index and distance files. This copy is frequently useful in determining why ALLOC VI fails to serve a specific node from a specific center, when the analyst's intuition suggests that it should.

It should be noted that the new index file and distance file are in exactly the same form as the original ones. The two sets of files differ only in their length and specific information.

### Section 4: Summary Information

After completing the second pass, RETRENCH writes the population file of the p-median data base as unformatted data on the same unit as the new distance file. This permits ALLOC VI to read both files from a single input unit and thereby reduces the amount of core storage needed for input buffers. This reduction, while small, has been valuable when the p-median data base was large.

RETRENCH prepares two summary tables at the end of its work. One contains the number of distances of any length in the new distance file. This information is useful for estimating the amount of core storage that ALLOC VI will need for storing the p-median data base. The second table gives the number of nodes that are a given distance from their



nearest candidate in the new distance file. That is, it lists the number of nodes whose nearest candidate is 0km away, 1km away, 2km away, and so forth. This information is useful when problem definitions in ALLOC VI are to contain maximum distance constraints, since a maximum distance constraint cannot be met if it is less than the distance from some node to its nearest candidate. When RETRENCH is directed to ignore some nodes in the new distance file, the summary table also ignores these nodes. As noted earlier, when a distance file ignores nodes all problem definitions that use the file in ALLOC VI will violate all maximum distance constraints, including the implicit one used to reduce the size of the distance file.

Finally, RETRENCH punches a list of candidate node ID numbers. Cards may be selected from this deck and used directly in the location constraints decks in ALLOC VI. Table VI-1 summarizes the punched output from RETRENCH.

### Input Data

The input data for RETRENCH consists of a control card; a population file and format; as many as four decks to define locations of candidates, fixed centers, and nodes to be ignored; an optional deck of data from an earlier first pass; and an index file and distance file to be modified. Figure VI-1 illustrates a sample input card deck for the program. The figure does not show the index and distance files, since these two files must be unformatted data. For convenience, the directions below assume that all data are to be read from punched cards. Only the population file and format, the data from an earlier first pass, and the index and distance files may be read from media other than punched cards.

Three header cards and the variable format for the population file contain alphanumeric data. The index and distance files contain unformatted integers. The remaining data are formatted integers, and they are to be right-justified in their allotted fields.

- 1.0 CONTROL CARD (required).
- 1.1 In columns 1-5, punch the number of nodes in the list of nodes. This value will be referred to as N.
- 1.2 In columns 6-10, punch the number of distance classes in the input index file. This value normally will be 1.



Table VI-1

Punched Output from RETRENCH

---

Information	Number of Items	Format
header card for fixed centers	1	*
ID numbers of fixed centers	KFIX	(I10)
header card for candidate nodes	1	*
ID numbers of candidate nodes	II	(I10)

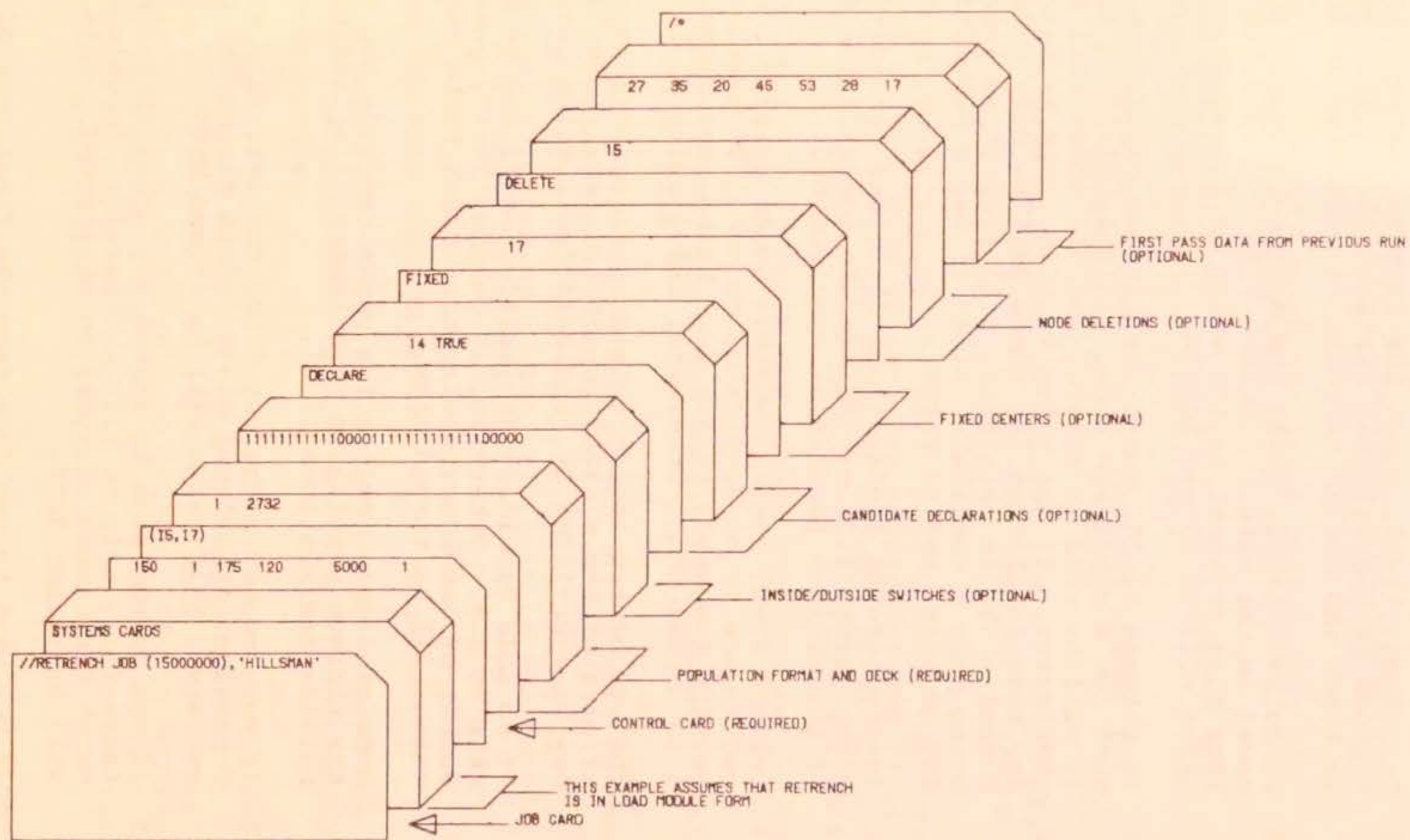
---

Source: Compiled by author. KFIX denotes the number of fixed centers and II denotes the number of candidate nodes.

\*The literal message on the card is self-explanatory.



FIGURE VI-1 Sample Input Deck for RETRENCH





- 1.3 In columns 11-15, punch the largest distance that appears in the original distance file. This value must be exact and not an estimate as permitted in ALLOC VI.
- 1.4 In columns 16-20, punch the maximum distance to be saved in the output distance file. This value may be less than or equal to the value in 1.3 above. This value will be termed LAMBDA.
- 1.5 In columns 21-30, punch the minimum population requirement for a candidate node. If this field is left blank it will be read as a zero, and all nodes will be made candidates. If the value in this field is larger than any node population, no nodes will be made candidates. Punching 350 in this field will prevent any node from being made a candidate unless it has a population of 350 or more, and so forth.

Inside/outside declarations (1.6) and candidate declaration cards (1.7) may be used to override the effects of the minimum population requirement.

- 1.6 Columns 31-35 control whether nodes outside the study region are to be candidates or not. To prevent nodes outside the study region from being candidates, punch a 1 in column 35. Otherwise, leave columns 31-35 blank.

The use of the inside/outside declarations overrides the effects of the minimum population requirement (1.5) and may be overridden by candidate declaration cards (1.7).

- 1.7 If no node candidacy declaration cards are to be read, skip to 1.8. Otherwise, in columns 36-40, punch the number of candidacy declaration cards to be read. This value will be termed KARB.

Node candidacy declaration cards override the effects of the minimum population requirement (1.5) and the inside/outside declarations (1.6).

- 1.8 If you are not using any fixed centers, skip to 1.9. Otherwise, in columns 41-45, punch the number of fixed centers. This value will be termed MFIX.
- 1.9 If you do not wish to ignore any nodes in the new distance file, skip to 1.10. Otherwise, in columns 46-50, punch the number of nodes to be ignored. This value will be termed JUNK.
- 1.10 If you wish to use data from the first pass of an earlier run of the program during this run, skip to 1.11. Otherwise punch a 1 in column 55 and, if you are using fixed centers (1.8), the program will make the first pass.



- 1.11 In columns 61-62, punch the input unit number for reading the format for the population file.
- 1.12 In columns 63-64, punch the input unit number for reading the formatted population file.
- 1.13 In columns 65-66, punch the input unit number for reading the unformatted index file.
- 1.14 In columns 67-68, punch the input unit number for reading the unformatted distance file.
- 1.15 If first pass data from an earlier run are to be read (1.10), in columns 69-70 punch the input unit number for reading the first pass data.
- 1.16 If this run will make the first pass through the data (1.10) and you wish to save the results of the first pass for use in future runs, in columns 71-72 punch the output unit number for punching the first pass results.
- 1.17 In columns 73-74, punch the output unit number for the new index file.
- 1.18 In columns 75-76, punch the output unit number for the new distance file.
- 1.19 In columns 77-78, punch the output unit number for the list of fixed centers and the list of candidates to be used in setting location constraints in ALLOC VI. It is strongly recommended that this unit be a card punch (unit 7 at most installations).

## 2.0 FORMAT TO READ NODE POPULATION FILE (required).

On the next card, punch a Fortran format to read one card of the population file. The format must specify integer fields.

For example, the population file might be punched one population to a card, with the ID number punched in columns 1-10 and the population in columns 11-20. The format for this deck would be (2I10). If the populations were punched four to a card, with ten columns for each ID number and population, the format for the file would be (8I10) or (4(2I10)).

## 3.0 POPULATION FILE (required).

The node population file contains the node ID numbers and populations punched together for each node, with



the ID number preceding the population of the node. You may punch more than one ID and population per card as long as you do not split a node ID and population between two cards, and as long as each card (except possibly the last in the deck) has the same number of ID numbers and populations.

Unlike the ALLOC programs, RETRENCH requires that the node ID numbers and populations be in the same order as the node ID numbers in the index file (8.0 below). RETRENCH will not rearrange the populations into the proper order.

3.9 If you are not limiting node candidacy to nodes within the study region (1.6), skip to 4.9.

4.0 INSIDE/OUTSIDE DECLARATION DECK (optional).

Each card column of the inside/outside declaration deck corresponds to one node in the list of nodes. If the list of nodes contains more than 80 nodes, then the first column of the second card of the deck corresponds to the 81st node in the list, and so forth.

If a node is inside the study region, punch a 1 in its column of the deck. If a node is outside the study region, punch a 0 in its column.<sup>8</sup>

4.9 If you are not using any candidacy declaration cards (1.5), skip to 5.9.

5.0 CANDIDACY DECLARATION CARDS (optional).

5.1 A candidacy declaration deck consists of a header card followed by a list of KARB (1.5) nodes with their candidacy status.

In columns 1-4 of the header card, punch DECL.

5.2 A candidacy declaration card may be punched in either of two formats, and a single deck may contain cards in both formats if desired.

To use the first format, punch a 1 in column 1 of the declaration card. In columns 2-10, punch the node ID number of the node to be declared. In column 15, punch T if the node is to be a candidate and F if it is not.

---

<sup>8</sup>Although the index file (9.2 below) indicates which nodes are inside the study region and which are not, RETRENCH must have this information before it begins to read the index file.



To use the second format, leave column 1 blank. In columns 2-10, punch the position of the node in the list of nodes (the row subscript of the node in the original distance matrix). In column 15, punch T if the node is to be a candidate and F if it is not.

- 5.3 If a node appears more than once in the declaration deck, the last status declaration for the node will override the preceding one(s).
- 5.9 If you are not using any fixed centers (1.6), skip to 6.9.

6.0 LIST OF FIXED CENTERS (optional).

- 6.1 The fixed centers deck consists of a header card followed by MFIX (1.6) cards to indicate locations of fixed centers.

In columns 1-4 of the header card, punch FIXE.

- 6.2 A fixed center location may be punched in either of two formats, and a single deck may contain cards in both formats if desired.

To use the first format, punch a 1 in column 1. In columns 2-10, punch the node ID number of the node that is to have a fixed center.

To use the second format, leave column 1 blank. In columns 2-10, punch the position of the node in the list of nodes (the row subscript of the node in the original distance matrix).

- 6.9 If you wish all nodes in the list of nodes to be included in the new distance file (1.7), skip to 7.9.

7.0 LIST OF NODES TO BE IGNORED (optional).

- 7.1 The list consists of a header card followed by JUNK (1.7) cards indicating nodes to be ignored.

In columns 1-4 of the header card, punch DELE.

- 7.2 A node that is to be ignored may be punched in either of two formats. A single deck may contain cards in both formats if desired.

To use the first format, punch a 1 in column 1. In columns 2-10, punch the node ID number of the node that is to be ignored.



To use the second format, leave column 1 blank. In columns 2-10, punch the position of the node in the list of nodes (the row subscript of the node in the original distance matrix).

7.9 If the program will not read data from an earlier first pass during this submission (1.10), skip to 8.9.

8.0 FIRST PASS DATA (optional).

The following description is for the sake of completeness, since you normally will not punch the first pass data yourself.

Each field of five columns corresponds to a node in the list of nodes. When the list of nodes contains more than sixteen nodes, the first field on the second card corresponds to the seventeenth node in the list and so forth.

In the field for each node, punch the distance from the node to its nearest fixed center (6.0). If the node does not appear in the distance string of any fixed center (that is, if it cannot be served by at least one fixed center within the implicit maximum distance constraint used to create the old distance file), punch 32000 in the field for the node.

8.9 This completes the punched input data for RETRENCH. The remaining data, described below, are unformatted and must be read from disk, tape, or some other medium that can be rewound and reread during execution of the program.

9.0 INDEX AND DISTANCE FILE (required).

9.1 The following descriptions are similar to those for the distance and index files read by ALLOC VI. There are two differences, however. ALLOC VI can read these files as formatted or unformatted data; RETRENCH requires that they be unformatted. In addition, RETRENCH requires that every node in the list of nodes have a string in the distance file that it reads. ALLOC VI permits strings to be missing, as for nodes that are not candidates. Thus, if RETRENCH creates a distance file with missing distance strings, as it will do when some nodes are not candidates, it cannot read that distance file to shorten it again.

9.2 For each node in the list of nodes, the index file must contain:



1. The sequence number (row number of the original distance matrix, or position of the node in the list of nodes) of the node.
2. The ID number of the node.
3. Any integer value. } These values are read but
4. Any integer value. } ignored.
5. A 1 if the node is considered to be inside the study region; a 0 if it is considered to be outside.
6. The number of distances in the first distance class.
7. The number of distances in the second distance class.

Continue for as many distance classes as the file contains, to a maximum of eleven classes.

The information in the index file must be provided in the order above. If you are running the program under IBM Fortran, items 1, 2, 3, and 4 must be four-byte integers and the remaining items must be two-byte integers.

- 9.3 For each node in the list of nodes, the distance file must contain:

1. The sequence number (row number of the original distance matrix, or position of the node in the list of nodes) of the node.
2. The diagonal element of the original distance matrix for this node.
3. The row number of the node nearest to this node.
4. The distance to the node in 3 above.
5. The row number of the node second nearest to this node.
6. The distance to the node in 5 above.

Continue for as many nodes as there are within the distance string.

The information in each string must appear in the order above. If you are running the program under IBM Fortran, all items in the string must be two-byte integers.

- 9.4 The program reads the index and distance files in alternating fashion starting with the index file for the first node in the list of nodes, followed by the distance string for the first node, followed by the index file for the second node in the list of nodes, and so forth. If both files are read from the same input unit, the data on that unit must be organized in this same fashion, as:



index for first node  
string for first node  
index for second node  
string for second node

and so forth. If the files are on separate input units, the elements of the files need only be in the order of the list of nodes.

9.9 This completes the input data for RETRENCH.

#### Error Conditions

The following conditions will produce an error message and stop the program.

1. Improper header cards for candidate declarations (5.1), fixed centers (6.1), or ignored nodes (7.1).
2. Failure to find node ID numbers for candidate declarations (5.2), fixed centers (6.2), or ignored nodes (7.2) in the list of node ID numbers obtained from the population file (3.0).
3. Node subscript less than 1 or greater than N (1.1) in the distance file. Most commonly caused by reversing the order of distances and node subscripts in the file.
4. Attempt to ignore a node that has a fixed center.

#### Execution Time and Core Storage

The execution time for RETRENCH depends upon the number of nodes in the distance files, the number of candidates and fixed centers in the new distance file, the size of the implicit maximum distance for the new and old files, and whether or not the first pass is required. In addition, because so much of the program's operation involves reading and writing data, execution time is affected by the record length and blocking factors for the new and old distance files. With this many variables involved, the execution times presented below can only be illustrative.

The version of RETRENCH in the listing was run twice on the 150-node test problem (Chapter II), using the IBM Fortran G compiler on an IBM 360/65. Both cases used the same original distance file. The file had a logical record length of 200 and had ten records per block. It contained 9,188



distances, and neither run used a shorter implicit maximum distance for the new file than for the old. One run used 150 candidates and two fixed centers to create a new distance file of 5,382 distances. RETRENCH required 10.69 seconds, excluding the time that it would have needed to write the new file on a tape. The second run used 80 candidates and one fixed center to produce a new file of 3,014 distances. This run required 8.29 seconds, including the time needed to write the new file on tape in the same form as the original file. This comparison indicates that reducing the number of candidates can reduce the program's running time by more than the amount needed to write the new file. Both runs required only 48 K bytes of core storage. Neither run produced more than 3,000 lines of printed output.

The version of RETRENCH in the listing is an improved, more efficient version of the program than the one that was used on the 2990-node distance file.<sup>9</sup> Because of the expense, the current version was never used to replicate a run made with the earlier version. Accordingly, the following case represents an upper bound on the amount of time that the new version would require on such a problem. Again, RETRENCH was run using the G compiler on an IBM 360/65. The old distance file contained 661,454 distances, with a logical record length of 1604 and four records per block. RETRENCH used 159 candidates and 35 fixed centers, with the same implicit maximum distance constraint, to produce a new file of 34,160 distances. The new file was written with a logical record length of 800 and one record per block. The run required the first pass. RETRENCH required 2 minutes, 50.42 seconds, including all input and output time. The time would have been markedly lower if the new file had been written with more records per block. Also, as noted above, the current version of RETRENCH would run more quickly. The program required 138 K bytes of core storage. This run produced 14,800 lines of printed output.

The array dimensions in the listing of RETRENCH permit the use of a distance file containing 150 nodes and an implicit maximum distance constraint of 150 units. The comment cards at the beginning of the program give directions for changing the dimensions of the program arrays.

---

<sup>9</sup>The improvement in efficiency resulted largely from a change in the forms of the index and distance files to the ones described in this monograph.



FIGURE VI-2 Sample Printed Output from  
RETRENCH (abridged)

---

PROGRAM RETRENCH

WRITTEN BY EDWARD L. HILLSMAN

DEPARTMENT OF GEOGRAPHY

THE UNIVERSITY OF IOWA

SUMMER, 1977

---

EDIT INDEX AND DISTANCE FILES FOR 49 NODES

INPUT INDEX FILE CONTAINS 1 CRITICAL DISTANCE CLASS(ES)--OUTPUT FILE WILL CONTAIN 1

MAXIMUM DISTANCE IN INPUT DISTANCE FILE IS 100--MAXIMUM IN OUTPUT FILE WILL BE 100

READ POPULATION FILE FROM I/O UNIT 5  
POPULATION FILE HAS BEEN READ

CANDIDACY

USING A POPULATION OF 500 AS A LOWER LIMIT FOR CANDIDACY, 45 NODES ARE CANDIDATES

MAKE 1 ARBITRARY DECLARATIONS OF NODE CANDIDACY  
THESE MAY DUPLICATE THE PROGRAMMED DECLARATIONS ABOVE

THE FOLLOWING 44 NODES ARE CANDIDATES FOR CENTERS

1	2	3	4	5	6	8	9	10	11
12	13	14	15	16	17	18	19	20	22
23	24	25	26	28	29	30	31	32	33
35	36	37	38	39	41	42	43	44	45
46	47	48	49						

ALL OTHER NODES ARE INELIGIBLE



CENTERS ARE TO BE FIXED AT EACH OF THE FOLLOWING 1 NODES  
THESE NODES HAVE BEEN ADDED TO THE LIST OF CANDIDATES

44

MAKE INITIAL PASS THROUGH DISTANCE FILE TO FIND DISTANCE FROM EACH NODE TO ITS NEAREST FIXED CENTER  
USE INDEX FILE FROM UNIT 9 AND DISTANCE FILE FROM UNIT 9

FIRST PASS COMPLETE--RESULTS FOLLOW

NUMBER OF INELIGIBLE NODES WITH A GIVEN DISTANCE TO THEIR NEAREST CANDIDATE NODE

DISTANCE 1	0
DISTANCE 2	0
DISTANCE 3	0
DISTANCE 4	0
DISTANCE 5	0
DISTANCE 6	0
DISTANCE 7	0
DISTANCE 8	0
DISTANCE 9	0
DISTANCE 10	0
DISTANCE 11	0
DISTANCE 12	0
DISTANCE 13	0
DISTANCE 14	0
DISTANCE 15	0
DISTANCE 16	1
DISTANCE 17	0
DISTANCE 18	0
DISTANCE 19	0
DISTANCE 20	1
DISTANCE 21	0
DISTANCE 22	0
DISTANCE 23	1
DISTANCE 24	0
DISTANCE 25	0
DISTANCE 26	0
DISTANCE 27	0
DISTANCE 28	0
DISTANCE 29	0
DISTANCE 30	0
DISTANCE 31	0
DISTANCE 32	0
DISTANCE 33	1
DISTANCE 34	0



DISTANCE 33	0
DISTANCE 34	0
DISTANCE 35	0
DISTANCE 36	0
DISTANCE 37	0
DISTANCE 38	0
DISTANCE 39	0
DISTANCE 40	0
DISTANCE 41	1
DISTANCE 42	0
DISTANCE 43	0
DISTANCE 44	0
DISTANCE 45	0
DISTANCE 46	0
DISTANCE 47	0
DISTANCE 48	0
DISTANCE 49	0
DISTANCE 50	0
DISTANCE 51	0
DISTANCE 52	0
DISTANCE 53	0
DISTANCE 54	0
DISTANCE 55	0
DISTANCE 56	0
DISTANCE 57	0
DISTANCE 58	0
DISTANCE 59	0
DISTANCE 60	0
DISTANCE 61	0
DISTANCE 62	0
DISTANCE 63	0
DISTANCE 64	0
DISTANCE 65	0
DISTANCE 66	0
DISTANCE 67	0
DISTANCE 68	0
DISTANCE 69	0
DISTANCE 70	0
DISTANCE 71	0
DISTANCE 72	0
DISTANCE 73	0
DISTANCE 74	0
DISTANCE 75	0
DISTANCE 76	0
DISTANCE 77	0
DISTANCE 78	0
DISTANCE 79	0
DISTANCE 80	0
DISTANCE 81	0
DISTANCE 82	0
DISTANCE 83	0
DISTANCE 84	0
DISTANCE 85	0
DISTANCE 86	0
DISTANCE 87	0
DISTANCE 88	0
DISTANCE 89	0
DISTANCE 90	0



DISTANCE 91 0  
 DISTANCE 92 0  
 DISTANCE 93 0  
 DISTANCE 94 0  
 DISTANCE 95 0  
 DISTANCE 96 0  
 DISTANCE 97 0  
 DISTANCE 98 0  
 DISTANCE 99 0  
 DISTANCE 100 0

ALL INFORMATION NECESSARY TO PERFORM MAIN EDITING ROUTINE HAS BEEN SAVED ON UNIT 8

INDEX AND DISTANCE FILES REWOUND FOR MAIN EDITING ROUTINE

MAIN EDIT TO USE GENERAL MAXIMUM DISTANCE OF 100

THE FOLLOWING NODE IS A CANDIDATE

1	0	22	40	28	51	2	54	7	84	43	87	38	100
---	---	----	----	----	----	---	----	---	----	----	----	----	-----

CUMULATIVE NUMBER OF DISTANCES=

THE FOLLOWING NODE IS A CANDIDATE

2	0	22	14	7	34	38	50	1	54	28	59	36	82	43	95
---	---	----	----	---	----	----	----	---	----	----	----	----	----	----	----

CUMULATIVE NUMBER OF DISTANCES= 15

THE FOLLOWING NODE IS A CANDIDATE

3	0	47	25	5	30	30	37	43	48	37	54	18	55	31	67	28	84	24	84
---	---	----	----	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

CUMULATIVE NUMBER OF DISTANCES= 27

THE FOLLOWING NODE IS A CANDIDATE

4	0	26	17	25	30	45	55	29	73	42	81								
---	---	----	----	----	----	----	----	----	----	----	----	--	--	--	--	--	--	--	--

CUMULATIVE NUMBER OF DISTANCES= 33

THE FOLLOWING NODE IS A CANDIDATE

5	0	3	30	47	61	24	63	30	73	43	84	37	90	18	91				
---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	--	--	--	--

CUMULATIVE NUMBER OF DISTANCES= 41



THE FOLLOWING NODE IS A CANDIDATE  
 6 6 72 48  
 0 11 50 40 40 59  
 CUMULATIVE NUMBER OF DISTANCES= 46

7 7 0 0 1 0

THE FOLLOWING NODE IS A CANDIDATE  
 8 8 44 50  
 0 40 33 11 35 27  
 CUMULATIVE NUMBER OF DISTANCES= 56

41 39 55 6 65 19 71 31 95

THE FOLLOWING NODE IS A CANDIDATE  
 9 9 27 67  
 0 27 43 12 44 10  
 CUMULATIVE NUMBER OF DISTANCES= 67

1 11 53 39 60 8 63 11 80 19 90 31 91 20 91

THE FOLLOWING NODE IS A CANDIDATE  
 10 10 68 78  
 0 25 31 20 38 19  
 CUMULATIVE NUMBER OF DISTANCES= 78

1 11 47 9 53 39 54 26 67 4 67 11 74 27 96

THE FOLLOWING NODE IS A CANDIDATE  
 11 11 29 35 23  
 0 30 65 30 57 2  
 CUMULATIVE NUMBER OF DISTANCES= 344

43 0 31 29 29 35 23 47 3 48 19 50 37 59 47 71 22 81 27 83

THE FOLLOWING NODE IS A CANDIDATE AND HAS A FIXED CENTER  
 12 12 345 364  
 0 35 12 48 20 33  
 CUMULATIVE NUMBER OF DISTANCES= 364

31 17 32 15 38 25 52 9 55 10 56 16 58

THE FOLLOWING NODE IS A CANDIDATE  
 13 13 265 371  
 0 29 18 42 26 26  
 CUMULATIVE NUMBER OF DISTANCES= 371

47 14 49 4 55 13 85

THE FOLLOWING NODE IS A CANDIDATE  
 14 14 372 383  
 0 34 22 44 38 41  
 CUMULATIVE NUMBER OF DISTANCES= 383

1 12 42 21 42 36 50 32 56 38 82 18 85 24 88

THE FOLLOWING NODE IS A CANDIDATE  
 15 15 394 393  
 0 3 25 37 34 31  
 CUMULATIVE NUMBER OF DISTANCES= 393

42 23 60 5 61 30 62 43 71 18 80 27 96







DISTANCE#	37	115
DISTANCE#	38	120
DISTANCE#	39	120
DISTANCE#	40	125
DISTANCE#	41	127
DISTANCE#	42	134
DISTANCE#	43	138
DISTANCE#	44	143
DISTANCE#	45	147
DISTANCE#	46	147
DISTANCE#	47	163
DISTANCE#	48	166
DISTANCE#	49	169
DISTANCE#	50	179
DISTANCE#	51	181
DISTANCE#	52	183
DISTANCE#	53	185
DISTANCE#	54	193
DISTANCE#	55	193
DISTANCE#	56	203
DISTANCE#	57	207
DISTANCE#	58	207
DISTANCE#	59	212
DISTANCE#	60	214
DISTANCE#	61	222
DISTANCE#	62	224
DISTANCE#	63	227
DISTANCE#	64	234
DISTANCE#	65	234
DISTANCE#	66	242
DISTANCE#	67	243
DISTANCE#	68	251
DISTANCE#	69	253
DISTANCE#	70	254
DISTANCE#	71	254
DISTANCE#	72	261
DISTANCE#	73	262
DISTANCE#	74	269
DISTANCE#	75	271
DISTANCE#	76	275
DISTANCE#	77	274
DISTANCE#	78	281
DISTANCE#	79	295
DISTANCE#	80	287
DISTANCE#	81	293
DISTANCE#	82	296
DISTANCE#	83	303
DISTANCE#	84	312
DISTANCE#	85	321
DISTANCE#	86	332
DISTANCE#	87	336
DISTANCE#	88	338
DISTANCE#	89	347
DISTANCE#	90	351
DISTANCE#		358



DISTANCE#	91	370
DISTANCE#	92	375
DISTANCE#	93	377
DISTANCE#	94	501
DISTANCE#	95	386
DISTANCE#	96	391
DISTANCE#	97	396
DISTANCE#	98	400
DISTANCE#	99	402
DISTANCE#	100	405

NUMBER OF INELIGIBLE NODES WITH A GIVEN DISTANCE TO THEIR NEAREST CANDIDATE NODE

DISTANCE	1	0
DISTANCE	2	0
DISTANCE	3	0
DISTANCE	4	0
DISTANCE	5	0
DISTANCE	6	0
DISTANCE	7	0
DISTANCE	8	0
DISTANCE	9	0
DISTANCE	10	0
DISTANCE	11	0
DISTANCE	12	0
DISTANCE	13	0
DISTANCE	14	0
DISTANCE	15	0
DISTANCE	16	1
DISTANCE	17	0
DISTANCE	18	0
DISTANCE	19	0
DISTANCE	20	1
DISTANCE	21	0
DISTANCE	22	0
DISTANCE	23	1
DISTANCE	24	0
DISTANCE	25	0
DISTANCE	26	0
DISTANCE	27	0
DISTANCE	28	0
DISTANCE	29	0
DISTANCE	30	0
DISTANCE	31	0
DISTANCE	32	0
DISTANCE	33	1
DISTANCE	34	0
DISTANCE	35	0
DISTANCE	36	0
DISTANCE	37	0
DISTANCE	38	0



## APPENDIX A

### USING THE ALLOC PROGRAMS TO SOLVE OTHER TYPES OF LOCATION-ALLOCATION PROBLEMS

Recent work, most of it unpublished, has shown that the p-median model is a powerful tool for solving a wide range of location-allocation problems (Church, 1974; Church and ReVelle, 1976; Hillsman, 1979). The approach has been to define two location-allocation models for a region--the one of interest and the p-median--and then show how to edit the p-median data base to yield a problem that is equivalent to the one of interest. The problem of interest then can be solved by solving the edited p-median problem. The use of distance constraints with the p-median problem, as discussed in Chapter I, is but one of many possible types of editing.

The purpose of this Appendix is to give directions for solving these edited p-median problems with the ALLOC programs. This discussion is necessary because the ALLOC programs were designed not for the full range of edited problems that they are capable of solving, but rather for unedited p-median problems in particular. As a result, the programs are often awkward to use on edited problems, and they require greater care when data for these problems are prepared and when solutions are interpreted. It is hoped that the programs will be revised in the next four years, and that these shortcomings will be eliminated.

Before proceeding to the directions, however, the Appendix presents four examples for reference during the directions. Although the examples cover a wide range of potential editing methods, they are not intended to exhaust the possibilities, and their presentation makes no effort to teach how to edit. Additional examples, and justification that the examples below perform as described, may be found in Church (1974) and in Hillsman (1979).

#### Examples

As described in Chapter I, the p-median data base consists of a list of nodes, a matrix of distances between the nodes, and a list of the node populations. The ALLOC programs use the distance matrix and the node populations to compute a coefficient matrix C in the following manner:

$$c_{ij} = w_i d_{ij} \quad \text{for all } i \text{ and } j$$

where  $w_i$  is the population of node  $i$ , and  $d_{ij}$  is the distance from node  $i$  to node  $j$ . The algorithms in the ALLOC programs operate on the computed coefficient matrix C, and



not directly on the distance matrix. By dividing each weighted distance by the weight, however, the programs can recover the original distances and use them to compute various summary statistics.

Chapter I also discussed the use of maximum distance constraints in the p-median model. At that time, it was noted that the programs imposed the constraints by identifying all distances greater than the maximum and then setting them equal to a very large number. This method is equivalent to computing the coefficient matrix C by setting:

$$\begin{aligned} c_{ij} &= w_i d_{ij} && \text{for } d_{ij} \leq S \\ c_{ij} &= M && \text{for } d_{ij} > S \end{aligned} \quad (\text{Example 1})$$

where M is an extremely large number, S is the largest distance to be used in the problem, and the remaining notion is the same as in the original p-median problem.

By generalizing the notion of the input data for the ALLOC programs in this manner, from that of a distance matrix to that of a coefficient matrix, it becomes possible to consider many additional problems. For example, Church and ReVelle (1976) have defined a problem in which the objective is to serve as many people as possible within a maximum distance, given a fixed number of centers. This "maximal covering location problem" can be solved using the following p-median coefficient matrix.

$$\begin{aligned} c_{ij} &= 0 && \text{for } d_{ij} \leq S \\ c_{ij} &= w_i && \text{for } d_{ij} > S \end{aligned} \quad (\text{Example 2})$$

The notation here is the same as in the earlier two cases.

As a third example of edited p-median problems, consider the problem solved by the trade-off algorithm in Chapter II. The problem seeks to minimize the average distance to the nearest center and to maximize the average resource score of places that have centers. An edited coefficient matrix for this problem would be:

$$\begin{aligned} c_{ij} &= w_i d_{ij} && \text{for } i \neq j \\ c_{ij} &= w_i d_{ij} - kr_j && \text{for } i = j \end{aligned} \quad (\text{Example 3})$$

where  $r_j$  is the resource score for node j and k is a weight chosen by the user. Increasing |k| will give relatively more emphasis to the average resource score and relatively less to accessibility.



The examples above assume that the number of centers to be located is known. It is possible to edit coefficient matrices for problems where the number of centers, as well as their locations, is variable. For example, if there is a fixed cost  $f_j$  associated with serving each node from a center, the problem might be to find the number and locations of centers to minimize total system costs. The coefficients for this problem are:

$$c_{ij} = w_i d_{ij} \quad \text{for } i \neq j \quad (\text{Example 4})$$

$$c_{ij} = w_i d_{ij} + f_j \quad \text{for } i = j$$

where  $d_{ij}$  is the transportation cost of serving one person at node  $i$  from a center at node  $j$ . To solve this problem, it would be necessary to run an algorithm for one center, two centers, and so forth. The number of centers that yielded the lowest total cost would be the optimum number of centers, and they would be optimally located.

#### Using Edited Coefficient Matrices in the ALLOC Programs

As noted earlier, when the problem to be solved is an unedited  $p$ -median problem, or one with maximum distance constraints, the ALLOC programs will compute the necessary coefficients. For other problems, the programs will accept a completely edited set of coefficients. There is one major restriction on the use of edited coefficients in the ALLOC programs, however, and several restrictions that are minor nuisances if only a few problems are to be solved and major shortcomings otherwise. The following discussion considers these restrictions for ALLOC V in detail; their extension to ALLOC VI is considered much more briefly, because it follows directly from an understanding of the distance file structure.

#### Restrictions

The major restriction on edited matrices in ALLOC V involves the values on the principal diagonal of the matrix: the minimum value in each row of the matrix must fall on the diagonal element of the row. In the examples given earlier, it is clear that as long as the distance to a candidate node from itself is taken to be zero and the distances to it from other nodes are positive, then the coefficients for the unedited  $p$ -median problem and the  $p$ -median with distance constraints will meet this requirement. By assuming this property for the weighted distance matrix, it was possible to design more efficient computer codes for the algorithms in the ALLOC programs.



The programs will not work on coefficient matrices that do not meet this requirement. As will be seen shortly, however, the restriction is less severe than it appears.

Many edited coefficient matrices contain the minimum element of each row on the diagonal. For example, in the resource/accessibility problem (Example 3) the coefficients are those of the unedited p-median, with the diagonal elements of the matrix made even smaller by subtraction of the  $k_{rj}$  term. In Example 2, the minimum value in each row is not unique, but it is still zero and it does fall on the diagonal. The matrix can be used as it stands although, as will be discussed later, it may make interpretation of results awkward.

In Example 4, the minimum value in each row of the matrix usually will not fall on the diagonal. To solve this problem, and those with similar matrices, it is necessary to introduce a correction term into the edited coefficient matrix, run the program, and then remove the correction term. Hillsman (1979) has shown that the correction term is a sufficiently large constant subtracted from every element on the principal diagonal. The size of the constant needed is computed by subtracting the minimum value in each row--whether on or off the diagonal--from the row's diagonal element, and adding one to the largest of these differences. Subtracting any constant of this size or larger from every principal diagonal element of the matrix will shift the row minimum in each row to the diagonal, but it will not affect the optimum solution to the problem. After the ALLOC program has solved the problem, the objective function for the problem will be too small by a value equal to the number of centers times the correction term used. This value must be added back into the function before solutions containing different numbers of centers are compared, as Example 4 would require.

If the distance matrix is a partial one, so that not all of the nodes to be served are candidates to receive centers, then the coefficient matrix will contain diagonal elements only for the candidate nodes. In this case, the row minimum requirement and the correction term calculations, apply only to the rows for the nodes that are candidates.

For ALLOC VI, the use of edited coefficients and the correction term is conceptually the same as described above, except that it makes use of a distance or coefficient file rather than a matrix. The distance (coefficient) strings within the file represent columns rather than rows, however. The choice of file structure was made for efficiency



in ALLOC VI, not for ease in checking compliance with the row minimum requirement of for computing the correction term. In general, the program RETRENCH cannot work with edited coefficients, so the actual editing of distances into coefficients should follow rather than precede the use of this program.

The ALLOC programs are made to read edited coefficients as input by substituting the coefficient matrix or file for the distance matrix or file, and by invoking the option to give every node an equal population of one. If the magnitudes of the coefficients are roughly the same as those of the original distances, the formats used for the data need not be changed. For ALLOC V, the distance matrix to be replaced is described at 2.0-2.9 in the instructions, and the equal unit weights option appears at 1.3. For ALLOC VI, the corresponding descriptions are at 3.0-4.5 and at 2.4.

In general, the ALLOC programs cannot re-edit the coefficients once they have read them. Thus, because imposing a maximum distance constraint is done through a form of editing, the programs cannot impose a tighter constraint, or use a looser one, than whatever constraint is embodied in the original coefficients. To change a maximum distance, as in Example 2, or to change another coefficient value, such as the  $k$  weight in Example 3, it is necessary to re-submit the program with the new set of coefficients.

It is the responsibility of the user to observe these limitations on edited coefficients. The ALLOC programs do not check to determine that the coefficients meet the row minimum requirement, or that problem definition cards are consistent with the coefficients in the data base.

#### Other Considerations

As noted in Chapter II, in the section on algorithm robustness, the use of maximum distance constraints can affect the robustness of some of the algorithms. The general effects of other forms of edited coefficients on robustness is not known. The performance of the Maranzana and the second phase of the Hillsman-Rushton algorithm is probably the most susceptible to degradation by editing, because these algorithms require accurate delineation of each center's service area. In the editing of Example 2, the coefficients permit the algorithm to determine whether or not a node is within the maximum distance of a center, but they do not contain enough information to determine which center within that distance is nearest. Thus, the coefficients do not permit service areas to be formed, and the two algorithms probably will do poorly on this set of coefficients and any others like it.



In general, the Teitz and Bart algorithm probably will be the least affected by different coefficients of all the algorithms in the programs. This subjective expectation must be confirmed by future research. There is also a possibility that the performance of some algorithms actually might improve when used on some types of coefficients although this, too, is speculation.

The final issue is that of interpreting the solutions that the programs find to edited problems. If coefficient matrices are used with care, as discussed earlier, the ALLOC programs will give an accurate list of centers in the optimum solution, an accurate objective function value (subject to the correction described earlier), and an accurate indication of the percentage change in the objective function. The list of centers and percentage change are properly labelled on the printed output; the objective function value is labelled "total weighted distance" in the printed output for each problem, and "weighted aggregate distance" in the summary at the end of the printed output for a run of the program.

The meaning, value, and accuracy of all of the remaining printed and machine-readable output is highly variable, because it depends in large part upon the edited coefficients used in the data base. Where the off-diagonal elements are weighted distances, as in the Examples 3 and 4, then the nearest center will be reported accurately for all nodes, as will the expendability of each center (labelled "cost if dropped" in some places). For other coefficients, this information may or may not be accurate for any specific node. As a rule, distances to the nearest center, distance times weight figures, average distances, and service center populations are not reported accurately for problems other than the simple p-median with or without distance constraints.

To obtain accurate information about service areas, average distances, and so forth, it is recommended that the solution from an edited problem be saved and submitted as the initial starting solution on a run of the program that uses an unedited p-median data base as input. The ALLOC programs can compute the necessary information very rapidly and inexpensively if the option is invoked to evaluate a solution without trying to improve it. The option is described at 5.2 for ALLOC V and at 9.2 for ALLOC VI.



LITERATURE CITED

- Aho, A.V., J.E. Hopcroft, and J.D. Ullman. 1974. The Design and Analysis of Computer Algorithms. Reading, Mass.: Addison-Wesley.
- Church, R.L. 1974. "Synthesis of a Class of Public Facilities Location Models." Ph.D. Thesis, The Johns Hopkins University.
- Church, R.L., and Meadows, M.E. 1977. "Results of a New Approach to Solving the p-Median Problem with Maximum Distance Constraints." Geographical Analysis, 9:364-378.
- Church, R.L., and ReVelle, C.S. 1974. "The Maximal Covering Location Problem." Papers of the Regional Science Association, 32:101-118.
- . 1976. "Theoretical and Computational Links Between the p-Median, Location Set-Covering, and the Maximal Covering Location Problem." Geographical Analysis, 8:406-415.
- Day, A. Colin. 1972. Fortran Techniques. New York: Cambridge University Press.
- Hakimi, L.S. 1964. "Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph." Operations Research, 12:450-459.
- Hillsman, E.L. 1977. "Spatial Competition in a Public Sector Location Model." Paper presented at the annual meeting of the Association of American Geographers, Salt Lake City, 1977.
- . 1979. "A System for Location-Allocation Analysis." Ph.D. dissertation, The University of Iowa.
- Hillsman, E.L., and Rushton, G. 1975. "The p-Median Problem with Maximum Distance Constraints: A Comment." Geographical Analysis, 7:85-90.
- Järvinen, P., J. Rajala, and H. Sinervo. 1972. "A Branch-and-Bound Algorithm for Seeking the p-Median." Operations Research, 20:173-178.
- Khumawala, B.M. 1973. "An Efficient Algorithm for the p-Median with Maximum Distance Constraints." Geographical Analysis, 5:309-321.



- Kuehn, A.A., and M.J. Hamburger. 1963. "A Heuristic Program for Locating Warehouses." Management Science, 9:643-666.
- Lin, S. 1975. "Heuristic Programming as an Aid to Network Design." Networks, 5:33-43.
- Maranzana, F.E. 1964. "On the Location of Supply Points to Minimize Transport Costs." Operational Research Quarterly, 15:261-270.
- Meneley, G.J. 1973. "Mobile Medical Diagnostic Clinics: A Study in Spatially Dynamic Location-Allocation." Mimeographed. Department of Geography, The University of Iowa.
- Ostresh, L.M. 1973. "SPA: A Shortest Path Algorithm." In G. Rushton, M.F. Goodchild, and L.M. Ostresh, Jr. (eds.), Computer Programs for Location-Allocation Problems. Department of Geography, Monograph No. 6. Iowa City, Iowa: The University of Iowa.
- Schilling, D.A., J.L. Cohon, C.S. ReVelle, and D.J. Elzinga. 1976. "Multiobjective Analysis in Public Facility Location." Paper presented at the spring meeting of the Operations Research Society of America, Philadelphia, 1976.
- ReVelle, C.S., and R.W. Swain. 1970. "Central Facilities Location." Geographical Analysis, 2:30-42.
- Rosing, K.E., E.L. Hillsman, and H. Rosing-Vogelaar. 1979a. "A Note Comparing Optimal and Heuristic Solutions to the p-Median Problem." Geographical Analysis, 11: 86-89.
- . 1979b. "The Robustness of Two Common Heuristics for the p-Median Problem." Environment and Planning A, 11:373-380.
- Rushton, G., K.J. Dueker, E.L. Hillsman, J.A. Kohler, and G.J. Meneley. 1978. "A Statewide Plan for Regional Primary Medical and Dental Care Centers in Iowa." Institute of Urban and Regional Research, Technical Report No. 57. Iowa City, Iowa: The University of Iowa.
- Rushton, G., M.F. Goodchild, and L.M. Ostresh (eds.) 1973. Computer Programs for Location-Allocation Problems. Department of Geography, Monograph No. 6. Iowa City, Iowa: The University of Iowa.



Rushton, G., and J.A. Kohler. 1973. "ALLOC: Heuristic Solutions to Multi-Facility Location Problems on a Graph." In G. Rushton, M.F. Goodchild, and L.M. Ostresh, Jr. (eds.), Computer Programs for Location-Allocation Problems. Department of Geography, Monograph No. 6. Iowa City, Iowa: The University of Iowa.

———. 1978. "Estimating the Socio-Economic Characteristics of Primary Medical and Dental Trade Areas." Institute of Urban and Regional Research, Technical Report No. 54. Iowa City, Iowa: The University of Iowa.



STATE LIBRARY OF IOWA



3 1723 02080 2187