

Processing of Map Information in a Minicomputer

Oscar Garcia

(Under the direction of Jerome L. Clutter)

Abstract

The documentation for a computer system called MAPS (Map Assembly and Processing System) is presented. MAPS is a package of 18 programs that can be used for input, processing and display of land classification maps and associated numerical information. The programs are written in BASIC for execution on a Wang 2200 minicomputer with 8K of core memory. In addition to the Wang 2200 C.P. U., a graphics terminal, a digitizer, two cassette units and a printer are used by the system. Currently, hard copies of output maps are obtained on a CALCOMP plotter driven by magnetic tapes produced on an IBM 360-370 to which the Wang 2200 communicates via telephone line. The present system has limited capacity for the automatic handling of overlays obtained by superimposing separate source maps. A conceptual design for a system with full overlaying capability is presented in an appendix.

MAPS uses the polygon method to obtain a computer representation of a map. With this method, the map is considered as a partition of the plane into regions delimited by boundaries that are approximated by polygonal lines. The sequences of vertices of these polygonals represent the map in the computer. Arcs -- pieces of boundary that separate two regions -- are the basic units or building blocks for the system. The system can produce maps consisting of any subset of the arcs of the original map. Specifications for map preparation may be conditions on numerical data associated with the regions of the base map. In general, processing of a source map will involve a map data file and a numerical data file.

The functions of MAPS include the following: entry and editing of maps; computation of areas; entry, editing and processing of numerical data; sorting of numerical data; selection and display of regions satisfying some specified logical condition concerning the associated numerical data; generation of maps obtained by grouping regions of the source map in classes defined by the associated numerical data; superposition of maps; general geometrical transformations of maps.

The appendix on overlays describes algorithms for adjusting for closure errors in the region boundaries, finding intersections between arcs, and recognizing regions formed by intersection.

PROCESSING OF MAP INFORMATION
IN A MINICOMPUTER

by

OSCAR GARCIA

Ingeniero Forestal, University of Chile, 1968

M.Sc.(Math.Statistics), CIENES - U. of Chile, 1972

A Dissertation Submitted to the Graduate Faculty
of the University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

1976

PROCESSING OF MAP INFORMATION

IN A MINICOMPUTER

by

OSCAR GARCIA

Approved:

James L. Clutter Date 3/19/1976
Major Professor

James C. Fulton Date 3/19/76
Chairman, Reading Committee

Approved:

Robert M. Hardy
Dean, Graduate School

3/24/76
Date

PREFACE

There seems to be a need for a relatively inexpensive and easy to use computer system for manipulating information from maps (5). As a step in this direction, a system called MAPS (Map Assembly and Processing System) has been developed. It can be operated at low cost in a mini-computer. The documentation for MAPS is presented in this report.

The present system has limited capacity for the handling of overlays. Specifically, it does not detect intersections in a composite map obtained by superimposing two separate source maps. A conceptual design for a system with full overlaying capability is presented in Appendix 2.

It is hoped that some of the ideas contained in MAPS and the concepts presented in Appendix 2 will be useful for the improvement of existing systems and in the design of new procedures.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS.....	iii
PREFACE.....	iv
LIST OF FIGURES.....	vii
CHAPTER	
I. OVERVIEW.....	1
Introduction.....	1
Map Input and Editing.....	2
Numerical Data Handling.....	3
Map Processing.....	5
II. GENERAL DESCRIPTION.....	8
III. PROGRAM DESCRIPTIONS.....	10
Program "MAPIN".....	11
Program "COMAP".....	22
Program "COPY".....	23
Program "NUMIN".....	24
Program "CONUM".....	31
Program "PRONUM".....	32
Program "SORT".....	36
Program "AREA".....	39
Program "SELECT".....	41
Program "SELECT2".....	47
Program "SELIST".....	49

	Page
Program "DISPLAY".....	50
Program "TRANS".....	51
Program "TRANS2".....	53
Program "TABLE".....	54
Program "FROMAP".....	55
Program "PLOT".....	59
Program "PLNMTR".....	64
APPENDIX	
1. PROGRAM LISTINGS.....	66
2. OVERLAYS.....	123
BIBLIOGRAPHY.....	134

CHAPTER I

OVERVIEW

Introduction

The MAPS system is a set of programs for input, processing, and display of maps and associated numerical information. It uses the polygon approach as opposed to the grid approach. That is, the map is considered as a partition of the plane into regions delimited by boundaries that are approximated by polygonal lines. The sequences of vertices of these polygonals represent the map in the computer.

The system has been implemented in BASIC language for the Wang 2200 minicomputer with 8K of core memory. The peripherals used are two cassette units, a Tektronix 4010 CRT terminal, a Texas Instruments terminal used as a printer, and a Numonics digitizer. Currently, hard copies of output maps are obtained on a CALCOMP plotter driven by magnetic tapes produced on an IBM 360-370 to which the Wang 2200 communicates via telephone line.

As already mentioned, the input map consists of a set of regions (not necessarily connected). Associated with each region is an identification number. A node will be defined as a point at which three or more regions meet. An arc is a piece of boundary between two successive nodes. A very long arc may be subdivided by the system, or by the user, into several parts in which case each part is an arc. Arcs are the

basic units or building blocks for the system. MAPS can produce maps consisting of any subset of the arcs of the original map.

Specifications for map preparation may be conditions on numerical data associated with the regions of the base map. Programs for input, editing, and manipulation of these numerical data are included in the system. In general, processing of a source map will generate a map data file and a numerical data file. A file may be contained in any number of cassettes, so that there is no limit on the number of arcs or regions that can be handled. Up to 28 variables can be defined in a numerical file.

Although written primarily for processing maps, many of the programs contained in MAPS may be useful in other applications. The programs for handling the numerical data files constitute a self-contained numerical data base management system and can be used, for example, for preparing data for statistical analysis. Several of the programs for map data can be used for any kind of line drawings.

Map Input and Editing

Program MAPIN is used to enter and edit map data. The map is digitized in continuous mode, one arc at a time, with each arc identified by the numbers of the regions at the left and at the right of the arc (consider the exterior of the map as a region). The program contains an algorithm that selects for storage and subsequent processing only those points necessary to ensure an error not greater than a prescribed limit in the polygonal approximation. An arc may contain up to 58 points. Longer arcs are subdivided into several arcs by the program. At input time, the area between the arc and the x-axis is also computed

for use by other programs which compute areas of regions. A line code for selecting different pens in drawing the arcs in the plotter is assigned.

In the output cassette the first record is a header that contains the name of the file, cassette number, comments, and other information. For each arc, the arc identification, area, line code, number of coordinates, and the coordinates, are stored in packed format in one physical record. The cassette is closed by an end of file record. There is space for 298 arcs in each cassette.

In edit mode, the program makes available a comprehensive set of commands for modifying the information in the header or in any arc, and for finding, displaying, listing, deleting, or adding arcs. After deleting arcs it is convenient to use program COMAP for compressing the file (i.e., eliminating the unused spaces in the tape).

Map files or numerical files to be processed by a MAPS program must start at the beginning of a cassette. Program COPY transfers files from any position in a cassette to any position in another cassette and can be used to duplicate files or to store several small files in the same cassette.

Numerical Data Handling

Program NUMIN is used to enter and edit numerical data. The data consists of any number of observations on up to 28 variables. In a mapping application, an observation contains the data associated with one region and one of the variables is the region identification number. The variables are identified by alphanumeric names.

In the output cassette the first record is a header containing the

name of the file, cassette number, comments, number of variables, and number of observations in the cassette (with an indicator showing whether or not the file continues in another cassette). The second record contains the name list for the variables, and the observations follow, occupying one physical record each. The cassette is closed by an end of file record. There is space for 297 observations in each cassette.

Edit mode of the NUMIN program provides a set of commands for modifying the header and names of variables, and for finding, listing, changing, deleting, or adding observations or variables. After observations have been deleted, program CONUM is used to compress the file.

Program COPY can also be used with numerical files to duplicate files and to store several files in the same cassette.

Program PRONUM is a flexible tool for processing and manipulating data files in several ways and either generating a new file or simply displaying the results. Variables can be added or deleted. A user supplied subroutine performs the desired computations for transforming variables, and/or generating new variables. Observations can be deleted or several observations can be generated from one observation in the original file. PRONUM can also be used to select a subset of a file or to concatenate several files.

Program SORT orders the observations according to increasing values for a given set of variables. Up to 297 observations (one cassette) can be accommodated.

Program SELECT can be used to find which observations satisfy a user-specified condition on the values of the variables. This program is discussed in more detail in the next section.

Map Processing

Program AREA computes the approximate areas of all the regions in a map data file. It can also incorporate these values into the corresponding numerical data file as a new variable. The errors seem to be typically of about .01 ~ .1 square inches. More accurate areas can be obtained directly from the original map with a planimeter or by using the digitizer and program PLNMTR.

Programs SELECT, SELECT2, and SELIST construct the subset of base map regions containing only those regions that satisfy some condition on the values of the variables. SELECT uses the condition to interrogate the numerical file and produce the list of regions to be included in the subset. This list is displayed and stored in a common area in core. The condition for selection of regions is specified as a logical expression that may contain names of variables, constants, and arithmetic, relational, and logical operators. This expression is interpreted by the program and evaluated for all the regions. All regions for which the values of the expression are true are included in the list.

SELECT has the optional capability to call the program SELECT2 which displays and/or produces a file with the map of regions included in the list. It also gives the combined area of the selected regions. The internal boundaries of the selected subset may be included if desired. If they are included, their arcs are identified with a different line code so that they can be plotted in a different color or different line thickness, if desired. Program SELIST is used to enter a selection list directly. After entry of the list SELIST then loads and executes SELECT2.

Program DISPLAY is similar in function to SELIST plus SELECT2, but

It only displays the map without producing a new file. DISPLAY can also be used to display the entire map without the specification of a list.

Programs TRANS, TRANS2 and TABLE are used for processing of a more general nature. Given a source map, these programs can produce any map that can be generated by combining regions of the original map. Program TRANS accesses a user-supplied subroutine to compute new identification numbers for the regions from the data in the numerical file. The old and new identifications are stored as a table in a common area in core, and program TRANS2 is loaded and executed. TRANS2 produces the new map by changing the identification numbers and eliminating arcs that now have the same region number at both sides. A simple example of the use of TRANS is production of a map showing the distribution of the values of some variable that takes on only a small number of values. This is accomplished by replacing the old region numbers with the values of the variable. Other applications involve the computation of some index as a function of several variables and subsequent use of the index as the new identification number. Program TABLE is used to directly enter the table that specifies the transformation from old to new identification numbers. After entry of the table, TRANS2 is automatically loaded and executed.

Program PROMAP provides the capability to access a user-supplied subroutine for doing any processing not possible with the programs discussed above. It gives access to all the information about the arcs, making possible, for example, geometric transformations such as changes of scale and cartographic projections. As in PRONUM, arcs can

be deleted or several arcs can be generated from the original one. With the subroutine omitted, PROMAP can be used to concatenate map files, changing the line codes if desired. This makes possible the superposition of several maps with each original map displayed in a different line type.

Finally, program PLOT provides communication between the Wang 2200 and the University of Georgia IBM 360-370. It transmits the map data, and controls the execution of a FORTRAN program that prepares the instruction tape for the CALCOMP plotter. Several maps can be transmitted and stored for processing by the FORTRAN program in the same IBM job which then produces only one plotter tape. The scale can be selected at transmission time and the selection of plotter pens is controlled by the line code of each arc. This program is considered to be a temporary component of the MAPS system and will be replaced by a program for controlling a plotter directly connected to the Wang when such a plotter becomes available.

CHAPTER II

GENERAL DESCRIPTION

MAPS is a package of 18 programs, written in BASIC for the WANG 2200. The programs operate in a conversational way, displaying instructions, and prompting the user for input and selection of options. Once the user familiarizes himself with a program, reference to the written documentation will seldom be necessary. Use of the system is simplified if special function keys 0, 1, and 2 are labeled as "ENTER", "DROP", and "EXIT", respectively.

In order to reduce searching and loading time, and in some cases to increase capacity, the programs have been compressed by a Wang utility program (9) to eliminate remarks, unnecessary blanks, and un-referenced statement numbers. The programs are contained in two cassettes. The first cassette contains MAPIN, COMAP, AREA, PROMAP, TABLE, TRANS, TRANS2, DISPLAY, PLOT, and PLNMTR. The second contains NUMIN, CONUM, COPY, PRONUM, SELIST, SELECT, SELECT2, and SORT (all the programs necessary for processing numerical data are in this cassette). In each cassette, the compressed programs are located first, followed by the source (uncompressed) programs, in the same order. An S has been added at the end of the program names listed above to form names for the corresponding source programs. Both the source and the compressed programs are listed in Appendix 1.

The following hardware items and device addresses are used in the

MAPS system:

Address	Item
-	Wang 2200B Central Processing Unit with 8K memory and Matrix ROM.
001	Wang 2222 Alpha-Numeric Typewriter Keyboard.
005	Tektronix 4J10-1 Computer Display Terminal.
10A,10B	Wang 2217 Single Tape Cassette Drive (2).
25A	Numonics Digitizer, Model 224-112.
01D	Texas Instruments Silent 700 Terminal, Model 733 KSR.
019,01D	UDS Acoustical Coupler.

PROGRAM "MAPIN"

PURPOSE.

Input and editing of map data.

USAGE.

Basic operating instructions.

Load and run the program. The printer should be on. Follow the directions supplied by the program, and select options when required. The program can be operated in either input mode or edit mode.

a) Input mode.

When requested to do so, enter the file identification name, and any comments or additional information about the file.

Several options for scaling the map on the screen are available. The scale factor entered at this point controls only the size of the map on the CRT screen. An edit mode command is available for making subsequent changes in this scale factor.

As the operator traces map segments with the digitizer, points are sampled continuously at a rate of one to three points per second. A "thinning" procedure is used for selecting and retaining only those points needed to approximate the map within a prescribed limit of error. The "maximum error for thinning" is the maximum distance between non-selected sampled points and the polygonal joining the selected

points. A reasonable minimum value for this quantity is the resolution of the digitizer (.025"). Larger values are appropriate in many applications.

The map is entered one arc at a time, in any order. The digitizer should be in continuous mode. When "ARC ID?" is printed by the program, position the digitizer stylus at the initial point of the arc and enter from the keyboard the numbers of the regions at the left and right of the arc, separating these by a comma. A "beep" indicates that arc-tracing can begin. Subsequent beeps signal when points are accepted by the thinning procedure. At the end of the arc, push the HALT/STEP key, and then the "ENTER" (special function key 0) and RETURN keys. If a mistake has been made, push the "DROP" key (special function key 1) instead of "ENTER". The arc will then be ignored (but not erased from the screen).

To end the entry of input, push the "EXIT" key (special function key 2). The EXIT key can also be used to terminate the program at any time (in input or in edit mode). Termination closes the file and resets the input/output addresses to the standard devices. This may be useful for continuing input after receiving an error message from the system. In this case, rerun the program and enter the remaining data using the edit mode.

A long "beep" during arc-tracing indicates that the maximum of 58 points for an arc has been reached. When this happens, keep the stylus stationary until the previously entered points are saved on tape and a beep indicates that tracing of the rest of the arc can be continued.

When the user selects edit mode, the tape is immediately positioned at the first arc. The following commands are then available:

- MAP. Draws on the screen all the arcs from the current position to the end of the tape. The tape then remains positioned at the last arc.
- LIST. Lists the arc identification, area under the arc, in square inches $\times 10^4$, line code, and number of coordinates (twice the number of points), for all the arcs from the current position to the end of the tape. The tape remains positioned at the last arc.
- INPUT. Sets the program in input mode and positions the tape for inclusion of additional arcs at the end of the currently existing file.
- NEXT. Positions the tape at the next arc.
- BACKSPACE. Backspaces the tape one arc. After DELETE, the tape is backspaced two arcs (see below).
- REWIND. Positions the tape at the first arc.
- FIND. Requests an arc identification, and positions the tape at the first arc, located at or subsequent to the current position, that separates the specified regions.
- PRINT. Prints the arc identification, area under the arc, line code, and number of coordinates, for the arc at which the tape is positioned. The tape remains at the same position.
- SHOW. Draws the arc at which the tape is positioned. The tape remains at the same position.
- CHANGE. Provides the capability for changing the identification

and/or line code of the arc at which the tape is positioned.

The tape remains at the same position.

DELETE. Deletes the arc at which the tape is positioned. The tape remains at the same position (at the preceding arc for the effects of BACKSPACE).

HEADER. Provides the capability for changing information in the header record: file name, cassette number, screen scale factor, maximum error for thinning, comments. The tape is positioned at the first arc.

END. Terminates the program. The same effect is achieved by pressing the "EXIT" key.

Command names in edit mode are recognized by the program by their first character only. They can be abbreviated in any way so long as the first character is unchanged.

Additional information.

The primary cassette unit is 10A unless otherwise designated by a previously entered SELECT command.

The file identification may contain up to 16 characters. Characters beyond the 16th are ignored. If the file identification contains commas or leading blanks, it must be enclosed in quotes (the quotes are not considered to be part of the identification). Imbedded blanks are valid characters. Comments can contain up to 64 characters (one line). The same rules mentioned above about use of quotes are valid here, and in general, for entry of any alphanumeric information. The file identification and comments are displayed by other programs using the file.

Three options are available for scaling the map on the CRT screen.

The full screen scaling option uses the coordinates of corners of the source document to compute the largest scale factor that will still permit the full map to appear on the screen (the same factor is used for vertical and horizontal scaling). The "standard" scaling option uses a scale factor of 32 screen units/map inch. This scales the full digitizer range (24" x 24") onto the full screen. The third alternative permits specification of a scale factor from the keyboard. The screen dimensions are 781 x 1024 screen units (approximately 140 screen units/screen inch).

As the user moves the digitizer stylus, the program samples data points at a rate of approximately 3 points per second. The thinning procedure generally accepts only some of these points for inclusion in the discrete point model of the arc. Points retained are selected so that the perpendicular distance between any rejected point and the polygonal joining adjacent accepted points is always less than the specified "maximum error for thinning". After a point is accepted, a delay of approximately one second occurs before the sampling rate of three points per second is resumed. The continuous sampling mode can be overridden at any time by switching the digitizer to discrete mode and entering points using the digitizer "SEND" switch. The change is most easily accomplished after entering the arc ID. The digitizer should be reset to continuous mode before pushing HALT/STEP. Alternatively, the digitizer can be kept in discrete mode, but the SEND switch should then be pressed twice for the first point of the arc and once again after pressing HALT/STEP.

The region identification numbers can be any positive or negative

integers of up to seven digits, or zero. The line code is set to zero by the program.

A single cassette has a maximum capacity of 298 arcs. As many cassettes as necessary can be used to hold data from a single source document. The program signals the operator when a cassette is full and provides instructions for mounting a new one.

The number of arcs indicated when entering edit mode includes any arcs deleted unless the tape has been compressed with program COMAP. In edit mode, a statement that the tape is positioned at an arc, actually means that part of the data for that arc has been read into core and the tape is positioned at the end of the arc record(s).

The file produced by MAPIN is organized as follows. The first record contains 6 fields: file identification (alphanumeric of length 16), cassette number, number of arcs in the cassette (with a minus sign if the file continues in another cassette), screen scale factor (in screen units per .01"), maximum thinning error (in hundredths of an inch), and comments (alphanumeric of length 64). Subsequently, each arc occupies one record, with one alphanumeric field of length 16, and 4 alphanumeric fields of length 58. The first field contains (in packed format) the left and right regions (arc identification)--each in format (+#####), area under the arc (in square inches $\times 10^4$)--format (+#####), line code--format (+###), number of coordinates--format (####). The fields of length 58 contain the coordinates, in hundredths of an inch, in the order $x_1, y_1, x_2, y_2, \dots$, and format (####). A system end of file record follows the last arc record on the tape.

METHODS.

As an arc is traced with the digitizer, the program receives a sequence p_1, p_2, \dots, p_n of points. The "thinning" procedure uses the given "maximum thinning error", ϵ , to select a subsequence with the following property. If p_r has been selected, the next point selected is p_s if, and only if, s is the largest integer (less than or equal to n) such that the distance between p_i and the line joining p_r and p_s is less than ϵ , for all i between r and s . The end points of the arc are always selected. Selected points are retained by the program for subsequent processing.

As coordinate pairs are obtained from the digitizer, they are examined to determine which points should be retained and which should be discarded. The basis of the thinning algorithm is the consideration that each p_i ($r < i < s$) constrains the next point selected, p_s , to lie inside the cone delimited by the tangents through p_r to the circle with center p_i and radius ϵ . As additional points are examined, the feasible set for p_s is the cone formed by the intersection of the cones for all the points between p_r and p_s . For each point, the feasible set is determined as the intersection of the previous feasible set with the cone for the point being processed. This point is discarded if the next point is inside the feasible set. If the next point is outside the feasible set, the point is saved and becomes p_r in the next iteration of the algorithm. The procedure is illustrated in Figure 1.

In the computer formulation of the thinning algorithm, cones are defined by the trigonometric tangents of the angles between the boundary lines of the cone and the x-axis (in Figure 2, $\tan \theta_+$ and $\tan \theta_-$). At

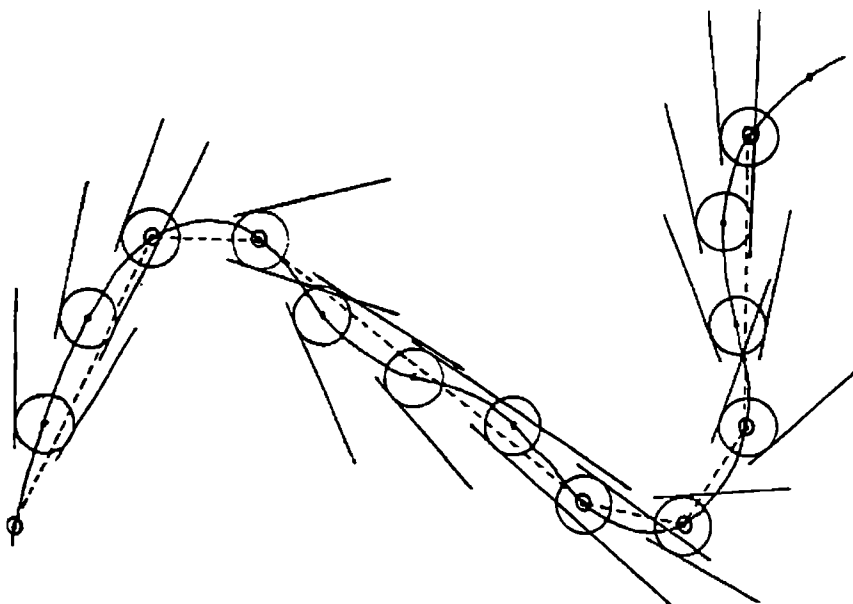


Figure 1

Illustration of Thinning Procedure

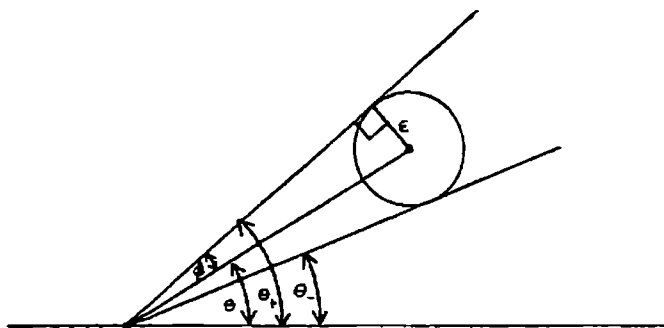


Figure 2

Feasible Cone Defined by Point p_i

any step in the process, the current feasible set is defined by U and L —the tangents for the upper and lower boundaries, respectively. To determine if point $p_i = (x_i, y_i)$ is inside the feasible set, the tangent of the angle θ between the line joining p_r and p_i and the x -axis is computed as

$$T = \tan \theta = \frac{y_i - y_r}{x_i - x_r}.$$

If $T > U$ or $T < L$, p_i is not inside the feasible set, so that p_{i-1} is saved and the procedure is restarted with $r=i-1$. If $L < T < U$, p_i is inside the feasible set, p_{i-1} is discarded and the feasible set is updated by letting U equal the minimum of the current U and $\tan \theta_+$ for p_i and L equal the maximum of the current L and $\tan \theta_-$ for p_i (Figure 2). These tangents are computed as

$$\tan \theta_+ = \tan (\theta + \phi) = \frac{\tan \theta + \tan \phi}{1 - \tan \theta * \tan \phi} = \frac{T + P}{1 - TP},$$

$$\tan \theta_- = \tan (\theta - \phi) = \frac{\tan \theta - \tan \phi}{1 + \tan \theta * \tan \phi} = \frac{T - P}{1 + TP},$$

where $P = \tan \phi = \frac{c}{\sqrt{d^2 - c^2}}$, $d^2 = (x_i - x_r)^2 + (y_i - y_r)^2$.

The formulas require all the angles to be in the first and fourth quadrants, or all in the second and third quadrants. To ensure this, point p_{r+1} is taken at a distance from p_r greater than δ , and a rotation of coordinates by 90° is performed if $|y_{r+1} - y_r| > |x_{r+1} - x_r|$. (Actually, the coordinates are not changed, but a different set of formulas is used). It can be shown that δ^2 must be greater than or equal to $(4+2\sqrt{2})c^2$; $\delta^2 = 7c^2$ was adopted.

A schematic flowchart for the thinning procedure is presented in Figure 3. The program iterates until execution is suspended by pressing

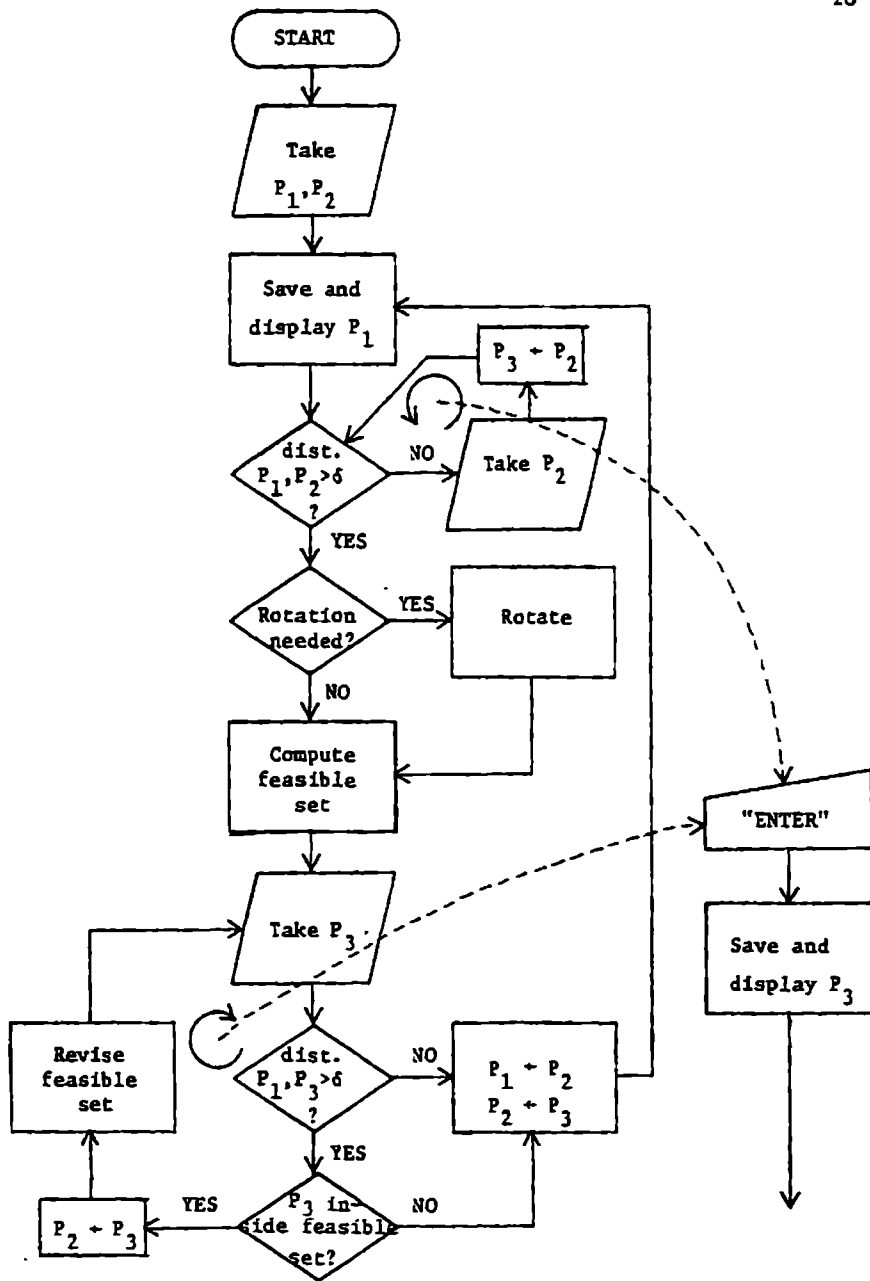


Figure 3

Thinning Procedure Flowchart

the HALT/STEP key. Execution is resumed by pushing the "ENTER" key and the RETURN key. The program then saves and displays P_j , computes the area under the arc, and saves the arc data on tape.

Another aspect of MAPIN that requires some explanation is the operation of the DELETE command. The procedure for deleting an arc makes use of the first two bytes in the physical records which normally are used only internally by the Wang system. The first byte is a hexadecimal 81 if the record is the last physical record of a logical record or a hexadecimal 82 otherwise. The second byte is used for numbering the physical records within the logical record.

To delete an arc, the first byte of the previous physical record is changed to hexadecimal 82, and the second byte of that record is added to the second bytes of the physical records corresponding to the arc to be deleted (an arc may be using more than one physical record due to previous deletions). Thus, the arc is not really eliminated from the tape but is made inaccessible to the programs since it becomes part of the previous logical record.

After deletion, the deleted arcs are still using space on the tape, and the number of arcs recorded in the header record is not updated since that number is actually the number of physical records used. Program COMAP eliminates the unused space by transferring to a new cassette only those physical records with a one in the second byte, changing the first byte back to 81 if necessary. COMAP also updates the number of arcs in the header record.

PROGRAM "NUMIN"

PURPOSE.

Input and editing of numerical data.

USAGE.

Basic operating instructions.

Load and run the program. Follow the directions supplied by the program and select options when requested. Output can be obtained from either the CRT (screen) or the printer. The user must select either input or edit mode.

a) Input mode.

When requested, enter the file identification name, and any comments or additional information about the file. Variables are designated by alphanumeric names of 8 characters or less. The user must enter the names, separated by commas.

To end the entry of input, push the "EXIT" key (special function key 2). The EXIT key can also be used to terminate the program at any time (in input or in edit mode). Termination closes the file and reselects the CRT as the output device if the printer was being used. This may be useful for continuing input after receiving an error message from the system. In this case, rerun the program and enter the remaining data using the edit mode.

b) Edit mode.

When the user selects edit mode, the tape is immediately positioned at the first observation. The following commands are then available.

- LIST.** Lists all observations from the current tape position to the end of the tape. Inclusion of the observation numbers in the list is optional. A specification of variables to be listed must be provided when requested. If "ALL" is given as the specification, all variables are listed. In either case, an empty line is necessary to indicate the end of the list of variables. The data are listed in 4 (or less) columns. For more than 4 variables, the values for each observation are printed in several lines. After listing, the tape remains positioned at the last observation.
- GET.** Positions the tape at a specified observation. If the observation number specified does not exist, a message is printed, and the tape is positioned at the first observation.
- FIND.** Positions the tape at the first observation, located at or subsequent to the current position, that has given values for specified variables. The program first requests the list of variables and then the list of corresponding values. The word ALL is acceptable as a list of variables. The end of the list of variables must be signaled with an empty line. If no such observation is found, a message is printed and the tape remains positioned at the last observation.

- NEXT.** Positions the tape at the next observation.
- BACKSPACE.** Backspaces one observation. When used immediately after a DELETE command, a backspace of two observations will occur (see DELETE).
- REWIND.** Positions the tape at the first observation.
- PRINT.** Prints the observation number and the values of given variables for the observation at which the tape is currently positioned. If ALL is specified instead of a list of variables, the values for all the variables are printed. The end of the list of variables must be signaled by an empty line. If no variables are specified, only the observation number is printed. The tape remains at the same position.
- DELETE.** Deletes the observation at which the tape is positioned. The tape remains at the same position. The numbering of observations is not updated until the tape is positioned at some observation located before the one deleted (e.g., with REWIND, or GET with a smaller observation number). Using BACKSPACE immediately after DELETE will cause a backspacing of two observations, reducing the observation count by one, so that the numbering will be incorrect until corrected with a REWIND command.
- HEADER.** Used to update information in the header record—file identification, cassette number, and comments. The tape is positioned at the first observation.
- VARNAMES.** Changes the names of specified variables. A list of current names for the variables involved must be specified.

followed by the corresponding list of new names. ALL is acceptable as a list of current variable names. The end of each list of variables must be signaled by an empty line. The tape is positioned at the first observation.

INPUT. Switches the program to input mode for adding observations at the end of the file.

ADD/SUBSTITUTE. Adds new variables to the file and/or substitutes new values for existing variables. A list of new variables and variables to be changed must be provided when requested. The end of the list of variables is signaled by an empty line. The starting observation number must also be specified. This will be 1 unless previous input under this command was suspended with EXIT and continuation from the last observation processed is desired. New values are entered for each observation. Input is terminated, and the file modification is completed, when the end of the tape or file is reached. Input may be interrupted at any point with the EXIT key, in which case, only the substitutions entered up to this point are incorporated in the file (the additions remain recorded in the tape, but these new variables are not considered to be part of the file). Input can be resumed from the point where interruption occurred by running the program again, using ADD/SUBSTITUTE with the same list of variables, and starting with the appropriate

observation number.

END. Terminates the program. The same effect is achieved by pressing the EXIT key.

Command names in edit mode are recognized by the program by their first character only. They can be abbreviated in any way so long as their first character is unchanged.

Additional information.

The primary cassette unit is 10A unless otherwise designated by a previously entered SELECT command.

The file identification may contain up to 16 characters. Characters beyond the 16th are ignored. If the file identification contains commas or leading blanks, it must be enclosed in quotes (the quotes are not considered to be part of the identification). Imbedded blanks are valid characters. Comments can contain up to 64 characters (one line). The same rules mentioned above about the use of quotes are valid here and, in general, for entry of any alphanumeric information. The file identification and comments are displayed by other programs using the file.

Lists of names of variables can be entered one per line, or several per line separated by commas. (The same is true for lists of numerical values.) Up to 28 variables can be defined. If more than 28 variables are defined only the first 28 will be considered. ALL is a reserved word, and it should not be used as a variable name. When entering lists of values corresponding to a list or variable names, values without a corresponding name will be ignored. The maximum capacity of a single cassette is 297 observations. As many cassettes as necessary can be used

for storage of a given file. The program signals when the end of a cassette is reached and provides instructions for mounting a new one. The numbering of observations begins from one in all cassettes including those which are continuations in multi-reel files. The number of observations indicated on entering edit mode includes any deleted observations if the tape has not been compressed with CONUM after deletion.

In edit mode, a statement that the tape is positioned at an observation, actually means that the data for that observation have been read into core, and the tape is positioned at the end of the observation record(s).

When using ADD/SUBSTITUTE in multi-reel files, care should be taken to ensure that the same modifications are made in all the cassettes involved. Any command beginning with A or S is interpreted as ADD/SUBSTITUTE.

The file produced by NUMIN is organized as follows. The first record contains 5 fields: file identification (16 alphanumeric characters), cassette number, number of variables, number of observations in the cassette (with a minus sign if the file continues in another cassette), and comments (64 alphanumeric characters). The second record contains the names of the variables in alphanumeric fields of 8 characters each. Subsequently, each observation occupies one record with each field containing the value of a numeric variable. A system end of file record follows the last observation.

METHODS.

Certain details concerning operation of the DELETE command may be of interest. The procedure for deleting an observation makes use of the

first two bytes in the physical records which normally are used only internally by the Wang System. The first byte is a hexadecimal 81 if the record is the last physical record of a logical record or a hexadecimal 82 otherwise. The second byte is used for numbering the physical records within the logical record.

To delete an observation, the first byte of the previous physical record is changed to hexadecimal 82, and the second byte of that record is added to the second bytes of the physical records corresponding to the observation to be deleted (an observation may be using more than one physical record due to previous deletions). Thus, the observation is not really eliminated from the tape but is made inaccessible to the programs since it becomes part of the previous logical record.

After deletion, the deleted observations are still using space on the tape, and the number of observations recorded in the header record is not updated since that number is actually the number of physical records used. Program CONUM eliminates the unused space by transferring to a new cassette only those physical records with a one in the second byte, changing the first byte back to 81 if necessary. CONUM also updates the number of observations in the header record.

PROGRAM "PRONUM"

PURPOSE.

PRONUM is a flexible, general purpose procedure for processing data in a numerical file. Through the inclusion of appropriate user-supplied subroutines, PRONUM can be used to delete specified variables and/or observations, generate additional variables and/or observations, and transform variables. The program can also be used to concatenate two or more numerical files.

USAGE.

Basic operating instructions.

After loading the program, push the RUN and RETURN keys and follow the instructions subsequently displayed. Available options provide for:

1. Selection of the CRT or the printer for producing written output
2. Specification of whether or not cassette output is to be produced.

In most cases cassette output is desired so that a new file is derived from the original. However, some applications require only a printout of certain quantities computed from the original file data. When cassette output is requested, the output file is written in cassette unit 10B. A new file identification name and any desired optional comments must be provided when requested. If additional cassettes are needed, appropriate mounting instructions will be displayed at the

proper time. If variables are to be added or deleted, the names of the variables involved must be provided when requested. Variable names in the same line should be separated by commas. The end of the list is indicated by an empty input line.

The program provides an opportunity for input of a user-supplied subroutine. (No subroutine is needed if the program is being used only for concatenating files without modification of the observations involved.) A subroutine could also be loaded after loading the main program but before executing it. The subroutine must be unlabeled (without DEFFN' statement) and must begin with statement number 1000. The file variables should be referenced as elements of a vector X. Exit from the subroutine must be through a RETURN statement. Execution of the program is resumed by pressing the "ENTER" key (special function key 0 which prints "RUN 490") and the RETURN key.

When processing of a file is completed, a message appears asking if there is another file. If the answer is NO, program execution is terminated. If the reply is YES, the next file is processed in the same way as the previous one, and the cassette output, if any, is saved as a continuation of any previous output. Through this procedure two or more files can be concatenated into a single new file. Files which are being concatenated must have the same number of variables.

The program can be terminated at any time by pressing the "EXIT" key (special function key 2). The new file is then closed in the normal way.

Additional information.

As in NUMIN, the new file identification is an alphanumeric field

containing 16 characters and the comments may contain up to 64 characters. Names of file variables are eight-character alphanumeric fields.

BASIC language variable designators containing 8 as the second character (e.g., A8, B8, A8\$) are reserved for the program, and should not be altered in the user-supplied subroutine (except for D8 and H8 which are discussed below). Tape commands addressing cassette units 10A or 10B should not be used. With these exceptions, any valid BASIC statements may be used in user-supplied subroutines. For example, the use of PRINT and INPUT statements can provide for user interaction in the processing of individual observations.

In the normal sequence of processing observations, the observation is first loaded from the original file in unit 10A. The user-supplied subroutine is then called to operate on the vector X which contains the original values of the file variables together with locations for values of any additional variables. On return from the subroutine, the new values are saved in the new file in unit 10B (if the cassette output option has been specified) with variables to be deleted excluded. This sequence can be altered using the "delete" and/or "hold" options in the subroutine.

Delete option. The variable D8 is set to 0 when the user-supplied subroutine is called. If D8 is set to 1 by the subroutine, the observation being processed will not be saved in the new file. This provides the user with a procedure for deleting unwanted observations and selecting specific subsets from a file.

Hold option. The variable H8 is set to 0 when the user-supplied subroutine is called. If H8 is set to 1 by the subroutine, no new observation will be loaded from the old file. This feature of the

program can be used to generate several new observations from a single original observation. At the next call to the subroutine, X will not generally contain the original observation so that any values necessary for further processing should be independently saved. It should also be noted, that successful operation of this option requires control through the use of counters or by user interaction.

Normal operating sequence can also be altered by using some specified logical condition to activate the command GOSUB'2 which will close the new file and terminate execution.

The order of the variables in the new file can be determined by first adding the new variables, in the order in which their names were entered, after the old variables in the original order and then substituting variables from the end of the list into positions vacated by deleted variables according to the order in which the deleted variable names were entered into PRONUM.

PROGRAM "SORT"

PURPOSE.

Orders the observations in a numerical data file according to increasing values for a given set of variables. Program capacity is limited to the contents of one cassette (up to 297 observations). The program will not sort multi-reel files.

USAGE.

Load the program, push the RUN and RETURN keys, and follow the directions subsequently displayed.

The file to be sorted should be mounted, unprotected, in cassette unit 10A. The sorted file is written over the original file in unit 10A. When it is necessary to retain a copy of the original file, a duplicate of the original file should be prepared with program COPY before SORT is used. The scratch cassette in unit 10B is needed only if the number of observations multiplied by the number of variables per observation exceeds 600 for more than two variables, 510 for two variables, or 255 for only one variable.

When the list of sorting variables is entered, variable names in the same line should be separated by commas. The end of the list is indicated by an empty line. The observations are ordered according to the values of the first variable in the list, for equal values of the first variable according to the value of the second variable, etc.

METHOD.

At each iteration of the sorting algorithm, as much data as possible is read from cassette unit 10A and sorted in core memory using a bubble sort (4). The sorted data in core are then merged with the data previously sorted and the merged file is stored on tape for subsequent merging with the next block of newly sorted data. Cassette units 10A and 10B alternate as storage devices for the merged data set prepared at each iteration. Storage into unit 10A is done onto tape space freed by previous reads. A schematic flowchart for the procedure is shown in Figure 4.

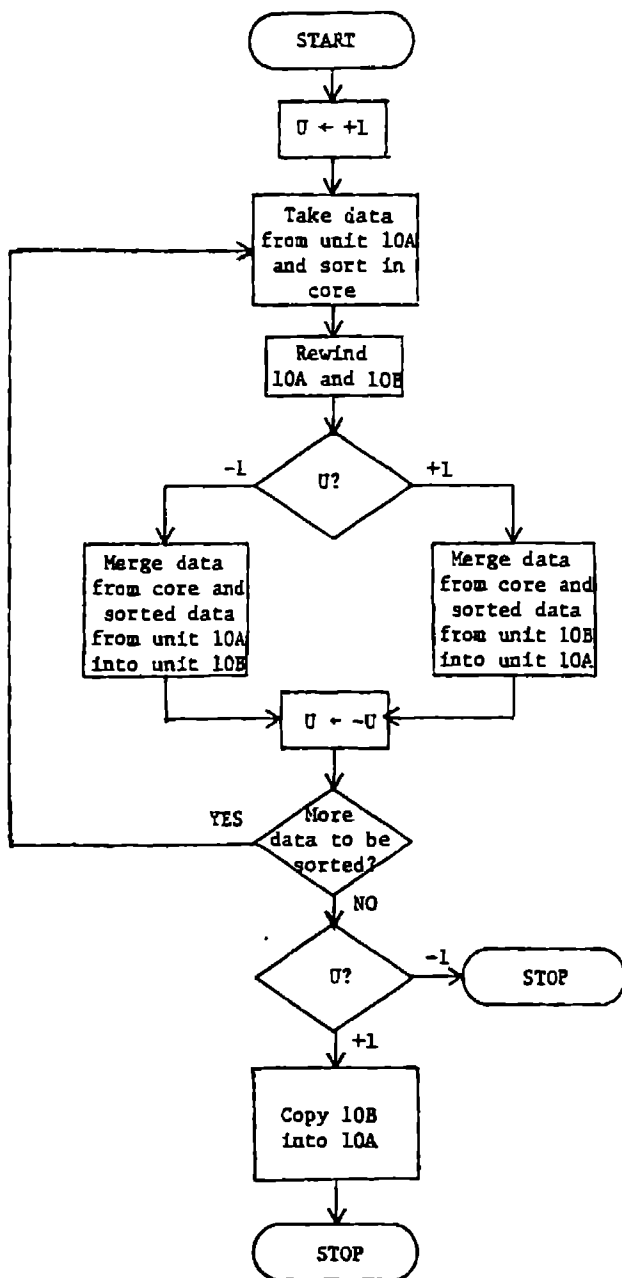


Figure 4. Schematic Flowchart for Program SORT

PROGRAM "AREA"

PURPOSE.

Computes the approximate areas of all the regions in a map data file and, optionally, incorporates these areas into the corresponding numerical data file as a new variable. Processing of a map data file with AREA is a useful procedure for detecting errors in the data.

USAGE.

Load the program, push the RUN and RETURN keys, and follow the directions subsequently displayed. Either the CRT or the printer can be selected as the output device.

A user-supplied scale factor is used to convert from square inches on the map to the desired area unit.

Areas are computed for regions with up to 99 arcs. If a region has more than 99 arcs, the area given for it by the program is incorrect, and a warning message is printed. Each map data file may contain up to 200 regions. For each region, the area, number of arcs, and a "closure check" figure are displayed. The meaning of the closure check is explained below.

A procedure for incorporating the calculated areas into a corresponding numerical data file is available. If this option is selected, the original numerical file is mounted in cassette unit 10A, and the new file, in which areas are included, is written in unit 10B.

The name of the variable that contains the region identification numbers in the file and the variable name for the areas must be specified. If regions exist in the numerical file that do not appear in the map, appropriate messages are printed, and the corresponding areas are set equal to zero.

Because of the limited resolution of the digitizer and errors made in tracing the arcs, the polygons that delimit the regions will not usually close exactly. A partial correction for this is made by calculating for each region, the net horizontal closure error where the net horizontal closure error is the algebraic sum of the projections of the gaps on the x-axis (with appropriate signs). The correction that is subtracted from the polygon area is the area of a column with base equal to the net horizontal closure error and height equal to the average y-coordinate for the end points of all the arcs for the region. The net horizontal closure error, in hundredths of an inch, is printed along with the areas, in the column labeled "closure check". Values much in excess of the digitizer's resolution (2.5) indicate gross errors in the map data. An addition check is provided by the fact that the sum of the areas over all regions should be reasonably small. (The area of the exterior "region" is negative, so that without errors the sum should

PROGRAM "SELECT"

PURPOSE.

Creates a list of the observations in a numerical data file that satisfy some specified logical condition concerning the values of the variables. When used with a numerical data file that has a corresponding map data file, SELECT can identify the regions that satisfy a given logical condition. Program SELECT2 can then be executed to produce a map showing these regions.

USAGE.

Basic operating instructions.

Load the program, push the RUN and RETURN keys, and follow the directions subsequently displayed. If a map is to be produced using SELECT2, the program cassette should remain mounted in unit 10A until SELECT2 has been loaded. Either the CRT or the printer can be selected as the output device.

The name of the variable that identifies the observations (for a map, the variable that contains the region numbers) must be specified. The values of this variable for the observations selected will be listed. Only the integer part of each value is displayed, and it should not have more than 7 digits.

The condition under which the observations (or regions) are selected is specified by an expression that may contain names of variables in the

file, constants, parentheses, and arithmetic, relational and logical operators. The available operators are:

Arithmetic: +, -, *, /, \uparrow (exponentiation).

Relational: <, <= (less than or equal to), =, >= (greater than or equal to), >, <> (not equal).

Logical: #AND#, #OR#, #NOT#.

Parentheses and blanks may be used freely. The expression must fit in one line, and produce a logical true or false value. The observations for which the expression is true are selected. The order in which the operations are performed, within the same level of parentheses, is equivalent to that in BASIC, FORTRAN, or ordinary algebra, with one exception: a sequence of exponentiations is evaluated from left to right. Errors of syntax are, in most cases, detected and indicated by the program. No more than 720 observations can be selected.

If the numerical data file being processed contains map data, the map processing option can be specified, if desired, after the list of selected regions is displayed. If this selection is made, program SELECT2 is automatically loaded and executed. Otherwise, execution is terminated.

The program can be terminated at any time by pressing "EXIT" (special function key 2).

Additional information.

In BASIC, FORTRAN, and many other high-level computer languages, the operators are grouped in hierarchical levels with respect to the order of operations, and within each level computation is performed from left to right (except for a sequence of exponentiations, where the

order is from right to left). In this program, all the operators in the selection condition have a different priority. However, the only resulting difference likely to affect the user is in the order of a sequence of exponentiations which will here be executed from left to right. The order of priorities, from higher to lower, is as follows: ~ (unary), †, /, *, - (binary), +, =, <>, >=, <=, >, <, #NOT, #AND#, #OR#. Sequences of the same operator are evaluated from left to right. The unary + is not a permissible operator.

The logical values, true and false are represented by 1 and 0, respectively, in evaluating the selection condition, so that variables with values 0 or 1 could be used as arguments for the logical operators and arithmetic operations could be performed on the results of comparisons or logical operations. The result of evaluating the logical condition must always be 0 or 1. If it is not, an error message is displayed.

If an error is detected when interpreting or when trying to evaluate the selection condition, a "SYNTAX ERROR" message is displayed, and the condition can be reentered. In addition, if the error was found in the interpretation step, a "^" symbol is placed under the character where the error was detected.

The list of identification values for the observations selected is stored in a common area in core where it can be used by program SELECT2. This list is stored by columns in the 240 x 3 alphanumeric matrix T8\$, with elements of four characters each and according to format (+#####).

Program SELECT2 should be located after SELECT in the cassette containing the programs.

METHOD.

Two subroutines are used to evaluate the selection condition. The first subroutine, which is executed only once, interprets the alpha-numerically expressed condition to produce a numerical vector that represents the condition in reverse Polish notation. The second subroutine uses this numerical vector to evaluate the condition for each observation.

The procedure used in the interpreter subroutine to produce a Polish string is based on techniques described by Maginnis (3). Operators can be ranked according to increasing priority of evaluation in a nonparenthesized expression, and consecutive integer values can be associated with them for representing this hierarchy. The effect of parentheses in an expression is to increase the hierarchy-rank values that determine the order of evaluation of the operators. A variable Q is used for keeping track of the different levels of parentheses, and is added to the operator ranks affected by parentheses to obtain the augmented operator values. Q changes from one level of parentheses to the next by an amount that can be any number greater than the maximum rank. In his description of the algorithm, Maginnis changes Q by 10, since he considers only arithmetic operators, and uses two "piles" in which operands and operators can be stacked. The algorithm given by Maginnis is as follows: (It is assumed that the expression being scanned is delimited by dummy operators with rank zero.)

1. Set Q to zero. Scan left to right incrementing Q by 10 for each left parenthesis and decrementing by 10 for each right parenthesis. Use Q and a table of operator ranks to maintain

- a list of augmented operator values.
2. Whenever an operand is encountered, place it in pile 1.
Whenever an operator is encountered, place it in pile 2. Do not move any parentheses.
 3. Whenever an encountered operator has a rank less than or equal to the value of the operator on top of pile 2, move the operator from pile 2 to pile 1. Continue this as long as the encountered operator has a rank less than the new top of pile 2.
 4. Terminate at the end of the scan. Pile 1, reading from bottom up, will be the reverse Polish string.

In program SELECT, the operators are numbered from 3 to 17, in order of increasing ranks. These numbers represent the operators in the Polish string, and at the same time they are used for comparing ranks in the interpreter subroutine. Constants are represented by a number 1 followed by the value, and variables by a 2 followed by the index of the variable in the numerical file. A schematic flowchart for the interpreter subroutine in SELECT is shown in Figure 5. Pile P contains, on return from the subroutine, the numerical vector representing the reverse Polish string, while pile T is a temporary storage vector.

The subroutine that evaluates the condition for each observation works as follows. Scan the Polish string from left to right. Start with an empty pile T. Whenever a constant or variable is encountered, place the value on top of T. Whenever an operator is encountered, replace the two values at the top of T by the result of performing the operation between them. (Take only one value if the operator is a unary one.) At the end of the scan, the value remaining in T is the desired result.

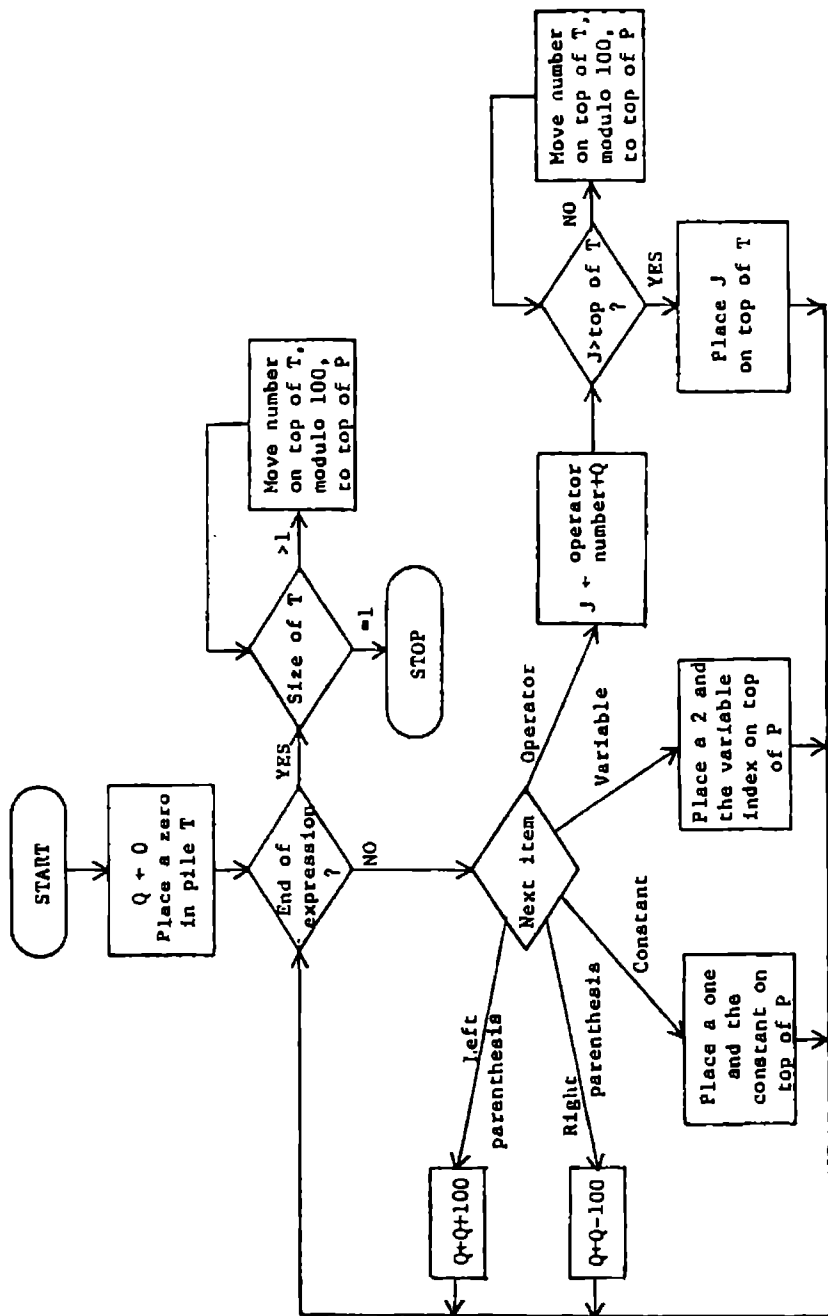


Figure 5. Interpreter Subroutine of Program SELECT

PROGRAM "SELECT2"

PURPOSE.

Produces a map showing a subset of the regions in the source map, and computes the approximate combined area of these regions. The selection of the subset is made by program SELECT or SELIST.

USAGE.

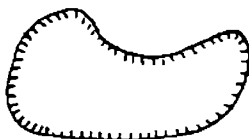
Basic operating instructions.

This program is automatically loaded and executed by programs SELECT or SELIST and uses the list of selected regions stored in core by one of these programs. The output map may be displayed at the CRT and/or stored in a cassette. If cassette output is selected, a name and optional comments for the new file must be provided. Inclusion of internal boundaries (i.e., the boundaries between the regions in the selected subset) is optional.

Additional information.

When the output map is stored in a cassette, the line code for the subset external boundaries in the output file is changed to +7 if the subset lies to the right of the arc and to -7 if it lies to the left. The line codes for internal boundaries are not changed. Line codes are used for selecting different line types when the map is drawn by a plotter. In the current implementation, the line code is used for

selecting different pens in the CALCOMP plotter so that different colors or line thicknesses are produced (see program PLOT). In the future, the capability of drawing dashed lines and other types of lines could be added. The sign in the line code produced by this program could be useful for using "oriented" types of lines—for example, a line with "shading" toward the interior of the selected subset, as shown below.



The procedure for computing the area and correcting for closure errors is the same as described for program AREA.

PROGRAM "TRANS"

PURPOSE.

In combination with program TRANS2, TRANS produces a new map generated from a source map by changing the region identification numbers. The new region numbers are computed from data in a numerical file corresponding to the source map by a user supplied subroutine. TRANS first computes a table containing the old and new identification numbers and then loads and executes TRANS2 which uses the table to produce the new map.

USAGE.

Basic operating instructions.

Load the program, press the RUN and RETURN keys, and follow the directions subsequently displayed. The cassette with the programs should be mounted in unit 10A and must not be removed until TRANS2 has been loaded.

Either the CRT or the printer can be used to produce written output.

The program provides an opportunity for input of a user-supplied subroutine and provides a message at the appropriate time for entry. A subroutine could also be loaded after loading the main program but before executing it. The subroutine must be unlabeled (without DEFFN' statement), and must begin with statement number 1000. The numerical file variables should be referenced as elements of a vector X. The new

region number must be given in variable N (the default is $N=X(1)$). Variable O must contain the old region number (default $O=X(1)$). Exit from the subroutine must be through a RETURN statement. Execution of the program is resumed by pressing the "ENTER" key (special function key 0 which prints "RUN 280"), and the RETURN key.

Up to 400 changes of region number can be made.

When end-of-file is encountered on the input cassette, an appropriate message is displayed and program TRANS2 is loaded and executed.

Execution can be terminated at any time by pressing "EXIT" (special function key 2).

Additional information.

BASIC language variable designators with 8 as the second character (e.g., A8, B8, A8\$) are reserved for the program, and should not be altered in the user-supplied subroutine. Tape commands addressing cassette units 10A or 10B should not be used. With these exceptions, any valid BASIC statements may be used. PRINT statements may be useful to display the region number changes made by the subroutine.

The pairs O,N for each change of region number are packed in format (+#####) and stored in a common area in core where they can be used by program TRANS2. They are stored by columns in the 200 x 2 alphanumeric matrix T8\$ with 8 character elements. Pairs with $O=N$ are not considered.

Program TRANS2 should be located after TRANS in the program cassette.

PROGRAM "PROMAP"

PURPOSE.

PROMAP is a flexible, general purpose procedure for processing data in a map data file. Through the inclusion of appropriate user-supplied subroutines, PROMAP can be used to delete arcs, generate new arcs, or transform the existing data. The program can also be used to concatenate two or more map files and to change line codes.

USAGE.

Basic operating instructions.

After loading the program, push the RUN and RETURN keys and follow the instructions subsequently displayed. Available options provide for:

1. Selection of the CRT or the printer to produce written output.
2. Specification of whether the output map is to be displayed on the CRT, stored on cassette tape, or both.

When cassette output is requested, the output file is written in cassette unit 10B. A new file identification name and any desired optional comments must be provided when requested. If desired, a new line code can be specified for inclusion with all arcs of the tape cassette output. The new code must be a one- to three-digit positive or negative integer. The original signs of the line codes are preserved. (If a negative code is specified, they are reversed.) If the word SAME is entered in place of a new line code, the codes remain

unchanged (unless modified by the user's subroutine).

The program provides an opportunity for input of a user-supplied subroutine. (No subroutine is needed if the program is being used only for concatenating files without modification of the data involved.) A subroutine can also be loaded after loading the main program but before executing it. The subroutine must be unlabeled (without DEFFN' statement) and must begin with statement number 1000. The arc information is made available to the subroutine in packed format in an alphanumeric variable A\$ of length 16, and an alphanumeric vector B\$ with 4 elements of length 58. Variable A\$ contains the following information, from left to right: left and right region numbers (arc identification), each in format (+#####); area under the arc, in square inches $\times 10^4$, format (+#####); line code, in format (+####); number of coordinates (twice the number of points), in format (####). Vector B\$ contains the coordinates, in hundredths of an inch, in the order $x_1, y_1, x_2, y_2, \dots$, and format (####). Exit from the subroutine must be through a RETURN statement. Execution of the program is resumed by pressing the "ENTER" key (special function key 0, which prints "RUN 300"), and the RETURN key.

When processing of the file is complete, a message appears asking if there is another file. If the answer is NO, program execution terminates. If the reply is YES, the next file is processed in the same way as the previous one, and the cassette output, if any, is saved as continuation of any previous output. A different line code can be specified. Through this procedure, two or more maps can be combined into a single new map.

The program can be terminated at any time by pressing the "EXIT" key (special function key 2). The new file is then closed in the normal way.

Additional information.

BASIC language variable designators containing 8 as the second character (e.g., A8, B8, A8\$) are reserved for the program, and should not be altered in the user's subroutine (except for D8 and H8, which are discussed below). Tape commands addressing cassette units 10A or 10B should not be used. With these exceptions, any valid BASIC statement may be used in user-supplied subroutines. For example, the use of PRINT and INPUT statements can provide for user interaction in the processing of individual arcs.

In the normal sequence of processing arcs the data for one arc are loaded from the original file in unit 10A. The user supplied subroutine is then called to operate on A\$ and B\$. On return from the subroutine the line code is changed (unless SAME has been specified), the new A\$ and B\$ are saved in the new file in unit 10B, and/or the modified arc is displayed, depending on the option selected. This sequence can be altered using the "delete" and/or "hold" options in the subroutine.

Delete option. The variable D8 is set to 0 when the subroutine is called. If it is set to 1 by the subroutine, the arc being processed will not be saved in the new file. This provides the user with a procedure for deleting unwanted arcs and selecting specific subsets from a file.

Hold option. The variable H8 is set to 0 when the subroutine is called. If H8 is set to 1 by the subroutine, no new arc will be loaded

from the old file. This feature of the program can be used to generate several new arcs from a single original arc.

Normal operating sequence can also be altered by using some specified logical condition to activate the command GOSUB'2 which will close the new file and terminate execution.

As an example of a subroutine for use with PROMAP, the following is a subroutine for changing the origin and scale of a map.

```

1000 IF F=1 THEN 1500
1010 F=1: DIM Z(58,2)
1020 INPUT "NEW ORIGIN (INCHES)",X0,Y0
1030 INPUT "SCALE FACTOR",S
1040 X0=100*X0: Y0=100*Y0: S2=S*S
1500 UNPACK(####) STR(A$,15) TO N
1510 N=.5*N: MAT REDIM Z(N,2)
1520 UNPACK(####) BS() TO Z()
1530 FOR I=1 TO N
1540 Z(I,1)=S*(Z(I,1)-X0)
1550 Z(I,2)=S*(Z(I,2)-Y0)
1560 NEXT I
1570 PACK(###) BS() FROM Z()
1580 UNPACK(+#####) STR(A$,9,4) TO A
1590 A=S2*A+.5: IF ABS(A)<1E7 THEN 1600: A=0
1600 PACK(+#####) STR(A$,9,4) FROM A
1610 RETURN

```

PROGRAM "PLOT"

PURPOSE.

Communicates with the University of Georgia 360-370 IBM system to prepare maps on the CALCOMP plotter. The use of this program is considered as a temporary expedient pending acquisition of a dedicated plotter. When this program is run, the Wang 2200 is essentially operating as a terminal under the Time Sharing Option (TSO) of the IBM system.

USAGE.

Basic operating instructions.

Load the program, press the RUN and RETURN keys, and follow the directions subsequently displayed. This program transmits map data files through a telephone line to the IBM system where they are stored in a previously defined data set. It also controls the execution, in the IBM system, of a FORTRAN program which takes the map data from the data set and produces an instruction tape for driving the CALCOMP plotter.

To establish the connection with the IBM system, disconnect the printer, connect input-output channel 019-01D to the acoustical coupler and establish a telephone connection between the coupler and the IBM system.

An appropriate TSO user identification code ("LOGON ID") should be

APPENDIX 2

OVERLAYS

Forest resource managers often refer to several maps of the same geographic area with each map containing somewhat different information. One map, for example, might delineate areas which have homogeneous timber stand conditions while another might show areas of homogeneous soil conditions as map regions. In such multimap situations, each source map is referred to as a layer. A commonly used technique in multimap situations is the process of overlaying whereby two layers are superposed to form a new map in which the regions are formed as intersections of the regions in the original layers. MAPS currently contains no provision for automatic preparation of overlays. Maps can be superposed with program PROMAP, but the combined map cannot be further processed as such since the original arcs continue to be the basic units and the coordinates of the intersections have not been identified. If the regions formed by the intersection of layers are to be recognized in the computer, the intersections of arcs from the various layers must be recognized at input time so that arcs of the composite map are individually identified. This can be done by entering the already combined map or, alternatively, by using a light table to find and mark the intersections of arcs on each layer. Region identification numbers must be assigned to the regions formed by intersection of the regions in the layers. Once a map is entered in this way, it is

not possible to subsequently add additional layers.

It is not a difficult task, conceptually, to describe a computerized mapping system that would possess an automatic overlay capability. First of all, some additional hardware beyond that used by MAPS would be desirable, since overlaying at a reasonable speed with the Wang 2200 would require a magnetic disk unit or some other type of direct-access mass storage device. One fundamental change that would affect the entire system would be a modification of the procedure for identifying regions. In the present version map regions are identified by a single number. In a revised version with overlay capability, region identification should probably be a vector with one element for each original source layer. This vector would contain the region identifiers from the various layers used in producing a composite map.

In the system being considered, some modifications would be required to guarantee closure in the computer representations of source-map polygons. With the present version of MAPS, the strings of coordinate pairs that represent polygon boundaries usually do not close exactly because of the limited resolution of the digitizer and the difficulty of exactly tracing the polygon boundaries. However, the existence of such closure errors would be an obstacle to operation of an effective automatic overlay algorithm. The proposed procedure for doing this is described below, and implies ordering the arcs in the boundary of each region in either a clockwise or counterclockwise direction. This ordering is used by one of the overlay procedures subsequently described and could also be useful in further processing of the maps, since it completely defines the relative positions and the interconnections of all the arcs contained in the map. This structure can

be recorded by including, with each arc, four pointers which contain the locations in the file of the preceding and following arcs in clockwise order for the boundaries of the regions at the left and at the right of the arc. These pointers will be identified as the structural pointers of the arc.

In preparation of a computer overlay, the following steps would be involved.

1. Concatenate the two source-map files to form a single combined-map file with the second source-file pointers revised to reflect the new storage locations.
2. Find all intersections between arcs of the first source map and those of the second source map. In general, each intersection transforms two arcs into four.
3. For each intersection found, replace the two old arcs by two of the new ones, add the other two at the end of the file, store the appropriate structural pointers for all four new arcs, and revise other pointers that referred to the original intersecting arcs.
4. Determine the arc identifications, or labels, for those four arcs in the new map. Each label contains the identification vectors for the regions at the left and at the right of the corresponding arc in the combined map.
5. Establish and store new labels for each non-intersected original arc.

An algorithm for finding intersections and two procedures for updating the labels of non-intersected arcs will be subsequently described. It should be noted that the above-described procedure could be iterated

to produce composite maps containing any number of original layers.

Operational efficiency of the overlay procedure described above could almost certainly be enhanced by providing "preprocessing" calculations designed to simplify subsequent intersection tests. Loomis (2), in connection with a procedure for determining inclusion of points and lines in regions of a map, has suggested one such procedure whereby the original arcs are subdivided into monotonic segments. This decomposition technique operates by breaking the arc at points which are local extremes (maxima or minima) in either their x- or y-coordinate values so that the new arcs obtained are non-increasing or non-decreasing in both coordinates. (The type of monotonicity need not be the same for both coordinates.) For each arc, a rectangle is defined with sides parallel to the x and y axes and opposite vertices at the end points of the arc. It follows from the monotonicity conditions that the arc lies entirely on or within the rectangle. Hence, two arcs can intersect only if their rectangles have a non-empty intersection. A test for intersection between two arcs can therefore be made by first checking whether the rectangles associated with the arcs overlap (this is made by comparing the coordinates of the end points of the arcs) and applying the intersection algorithm only if the rectangles overlap.

An alternative to the Loomis decomposition involves retaining the original arc and finding the smallest rectangle (with sides parallel to the axes) that contains the arc. The rectangles so obtained can then be used to screen the pairs of arcs to be examined for intersections. This alternative has the advantage of avoiding an increase in the number of arcs which, in turn, causes increased storage requirements and com-

puting time in all the processing programs. On the other hand, Loomis' decomposition simplifies the search for intersections and does not require updating of the rectangles after each overlay operation. Furthermore, the Loomis decomposition is applied only once to each layer. (In subsequently derived composite maps, the rectangles containing the arcs will always be defined by the end point of the arcs.) It is not immediately obvious which of these two alternatives would be more efficient, but some method of reducing the number of pairs of arcs to be examined in detail should improve efficiency.

It should be noted that most of the processing described above makes no use of the interior points of the arcs. It could thus be advantageous to use a separate file for storage of the interior points. The first file would then contain, for each arc, the arc identification (region vectors), number of points, structural pointers, the end points, the pair of points defining the arc rectangle (if Loomis' decomposition is not used), and a pointer indicating the location of the interior points in the second file. In addition to reducing access time, this file organization could also save storage space by permitting the use of fixed length records in the first file and variable length records or sequential access in the second file.

Some specific algorithms which could be used in a system with automatic overlay capability are described below.

A Gap-Closing and File-Structuring Algorithm

This algorithm would close the small gaps in region boundaries created by input errors and, optionally, create and store the structural pointers which describe the basic structure of the map. The regions

contained in the map would be processed as follows:

1. Find the smallest region number— N .*
2. Search the file. Copy into core the arcs in the boundary of region N (i.e., those arcs containing N in their identification) and their file addresses. At the same time, find N' —the smallest region number greater than N .
3. Find the clockwise order for the arcs of region N and close any existing gaps (see below). Set the appropriate structural pointers.
4. Return the modified arcs to the file.
5. If no N' was found in step 2, terminate. Otherwise, set $N = N'$ and go to step 2.

Step 3 is performed with the arcs stored in core memory. (Actually, only the end points of the arcs are used.) It should be noted that in the detailed description of step 3 which follows, the terms initial point and terminal point of an arc apply in relation to a clockwise boundary description of region N . Thus, the initial point is the first point of the arc if N is the second number in the arc identification, and is the last point if N is the first number in the identification. Analogously, the terminal point is the first or last point of the arc depending on whether N is the first or the second region number in the identification. The following procedure produces a clockwise sequence of the arcs of region N and closes any gaps existing in this sequence as called for in step 3 above:

*It is assumed here, for the sake of simplicity, that regions are identified by single numbers. If vector identifications are used, some additional rules must be used.

- a. Take any arc as the first arc in the sequence.
- b. If there are more arcs remaining, continue with step c. Otherwise, check whether the terminal point of the last arc coincides with the initial point of the first arc. If not, close the gap, as described in step d, and stop.
- c. Compute the distances between the terminal point of the last arc in the sequence and the initial points of all the arcs not in the sequence. The arc for which this distance is a minimum is added to the sequence.
- d. If the initial point of the arc just added does not coincide with the terminal point of the previous arc, make them equal by changing one of them. Change the point contained in the arc with the largest region number in its ID. Go to step b.

As a distance, either the square of the Euclidean metric, $(x_i - x_j)^2 + (y_i - y_j)^2$, or the metric $|x_i - x_j| + |y_i - y_j|$ could be used. The latter should be more efficient in terms of computing time. It might also be desirable to have a warning message displayed if, at any time, the minimum distance exceeds some predefined tolerance value.

The Intersection Algorithm

Whenever the rectangles of two arcs overlap, a search procedure must be applied to establish any intersections that may occur. The algorithm assumes that each arc is either non-increasing or non-decreasing in its y-coordinates. This will automatically be true if the Loomis decomposition procedure is used. Alternatively, the arcs can be subdivided into monotonic parts (this can be done concurrently with the intersection search) with each part of one arc being checked

for intersections with each part of the other.

The algorithm proposed here is based upon the following observation. Consider two continuous curves, each of which is either monotonically increasing or monotonically decreasing in y and locate a point on each curve such that the y -coordinates of the two points are equal. Move the points along the curves in a way that maintains the equality of their y -coordinates and observe the difference in x -coordinates between the two points. Intersections between the two curves occur at, and only at, those points where the difference in x changes sign.

The principle just noted would be implemented in practice by making use of the fact that the arcs are composed of points connected by straight line segments. It is therefore possible to move along the arcs in discrete steps until the segments which intersect are located and the intersection point computed. Details of the algorithm are summarized below (with the search proceeding upward along the arcs):

1. Let P be the higher of the initial (bottom end) points of both arcs. Let S be the straight line segment of the other arc that contains the y -coordinate of P . Find the sign of the horizontal distance between P and S . In the calculation of these signs, the first element of the difference must always come from the same arc.
2. Let P' be the point in the P -arc that follows P . If P' is lower than the upper end of S , make $P = P'$. Otherwise, take as P the upper end point of S and let the segment P, P' be S .
3. Find the sign of the horizontal distance between P and S . If it is different from the previous one, go to step 4. If P is

- the last point of the arc, terminate. Otherwise, go to step 2.
4. Compute the intersection between S and the segment for which P is the upper end point.
 5. Repeat the entire procedure, starting again at 1 with the points immediately above the intersection serving as the initial points.

It should be noted that the sign of the horizontal distance between P and S can, in most cases, be determined by simply comparing P with the end points of S. Only if the x-coordinate of P is between the x-coordinates of the end points of S is it necessary to find the point on S at the same height as P. Efficiency can also be improved by remembering that, with Loomis' decomposition, two arcs with inverse monotonicity intersect in at most one point so that, in this case, the search can be terminated as soon as an intersection is found.

Algorithms for Labeling Non-Intersected Arcs

When a map A is overlaid with a map B to form a new map C, the new arc labels, that is, the updated identification vectors for the regions to the left and right of intersecting arcs, can easily be generated from the original identifications of the intersecting arcs and the configuration of the intersection. Although arcs of A that do not intersect with arcs of B remain unchanged in map C, their labels must still be updated by finding the regions of B in which they are contained.

One straightforward procedure for labeling an arc of A that does not intersect with B involves selection of any point on the arc and use of an algorithm for finding the region of B in which the point is

contained. This process would have to be carried out for every non-intersecting arc. An efficient algorithm for determining the region that contains a given point on a map operates as follows. Consider a ray with origin at the given point (and parallel to the x- or y-axis, to simplify the computations). Find all the arcs intersecting the ray an odd number of times. That region appearing an odd number of times in the ID's of these arcs is the region containing the point. Some care is necessary for handling the case where the ray passes through end points of arcs. The general form of this algorithm appears to have been independently identified by several authors (1,6,8). (See also Loomis (2).) It should be noted that this method does not require use of the structural pointers.

A second method for labeling the non-intersecting arcs is based on the observation that the label of an arc can be completed from knowledge of the label of any of the arcs indicated by its structural pointers. In applying this method, the arcs generated by the intersections would be labeled and the file would then be searched for unlabeled arcs. For each unlabeled arc, the arcs indicated by its structural pointers would be examined. If any of these were labeled, the arc could be labeled. If the arcs of the map formed a connected network (i.e., no "islands" are present), all the arcs could be labeled by repeating this process a finite number of times (usually one or two). Dummy arcs could be used to guarantee that each layer constituted a connected network. Alternatively, the first method could be used to label those arcs that could not be labeled with the second procedure.

This second method is probably less time consuming than the first

but it requires storing the structural pointers and updating them in the overlaying process. On the other hand, the structural pointers could be useful for other purposes.

Although the approach and procedures described above cover the main problems encountered in designing a map processing system with automatic overlay capability, it is obvious that implementing such a system would not be a trivial matter. Many decisions would be involved in attempting to reach the proper balance between computing time, storage requirements, system complexity, and coding efficiency. It is also possible that better algorithms could be found. Anyone attempting implementation of such a system should be interested in the pertinent literature review by Rosenfeld (7) and the references to other similar systems given by Massam (5).

BIBLIOGRAPHY

- (1) Jacobsen, J. D. 1968. Geometric relationships for retrieval of geographic information. IBM Systems Journal, 7(3&4): 331-341.
- (2) Loomis, R. G. 1965. Boundary networks. Comm. of the ACM, 8(1): 44-48.
- (3) Maginnis, J. B. 1972. Elements of compiler construction. Appleton-Century-Crofts. p. 90.
- (4) Martin, W. A. 1971. Sorting. Computing Surveys, 3(4): 147-174.
- (5) Massam, B. H. 1975. An evaluation of geo-information systems with special reference to the Canada Geographical Information System: The user's viewpoint. Urban Planning, McGill University. February 1975.
- (6) Nordbeck, S., and B. Rystedt. 1967. Computer cartography--point-in-polygon programs. BIT, 7: 30-64.
- (7) Rosenfeld, A. 1973. Progress in picture processing: 1969-1971. Computing Surveys, 5(2): 81-108.
- (8) Shimrat, M. 1962. Position of point relative to polygon. Algorithm 112. Comm. of the ACM, 5(8): 434. See also: Hacker, R. Certification of Algorithm 112. Comm. of the ACM, 5(12): 606.
- (9) Wang Laboratories, Inc. 1974. System 2200 Programming Tools. January 10, 1974 - Revision 2.

77-12,381

GARCÍA VIDAL, Oscar Pablo, 1945-
PROCESSING OF MAP INFORMATION
IN A MINICOMPUTER.

University of Georgia, Ph.D., 1976
Computer Science

Xerox University Microfilms, Ann Arbor, Michigan 48106