INTERACTIVE ANALYTICAL DISPLAYS FOR SPATIAL DECISION SUPPORT SYSTEMS

Marc P. Armstrong
Panagiotis Lolonis

Department of Geography
The University of Iowa
Iowa City, IA 52242
BLAMMGPD@UIAMVS.BITNET

INTRODUCTION

Geographic information system technology is now being placed in the hands of decision-makers who often have little or no cartographic training. Although it is possible that these users will glean insight from the displays that they generate, map readers are able to recover more information from a carefully designed display, than from a poorly designed display. In our application, we are attempting to integrate spatial modelling techniques (e.g. location-allocation models) with the spatial data handling and display capabilities of GIS, to improve the process of spatial decision-making. Although it is important that decision-makers be provided with a capability of viewing the results of spatial models, we wish to isolate users from technical, mundane details of generating the display. To achieve this goal we have developed techniques that allow an expert system to assume responsibility for some aspects of thematic map layout and design.

PREVIOUS WORK

Computer-Assisted Map Design

Monmonier has described a general system used to monitor the process of layout for the National Atlas of the United States (Monmonier, 1982:159). In that system, predetermined configurations are used to guide page design. Each configuration is assigned a code, and pages are built from composites of these codes. Monmonier (1982: 159-163) also describes several other areas of cartographic production that lend themselves to automation: aggregation, data reduction to control display of attribute data, and the general problem of map layout. In his discussion, Monmonier distinguishes movable and fixed map components. In our application, we expand this notion to develop rules for determining the motility of map components.

Broome, Beard, and Martinez (1987) describe an approach to determining the conditions under which an inset map should be produced to enhance legibility, and then to decide where the inset should be placed. The system attempts to mimic the process through which a cartographic designer would arrive at the same answer. The map is divided into cells, and the number of cartographic objects in each cell is tabulated, and smoothed using a 9 cell window. This notion of feature density is important in our application, but in a slightly different way because we are concerned with finding empty areas, rather than areas with high feature density.

Artifical Intelligence Approaches

Several researchers have been involved in exploring the application of artificial intelligence principles to cartographic problem solving.
Maggio (1987) suggests an approach in which a GIS becomes a part of an expert system. Morse (1987) developed an approach in which existing GIS software (MOSS) is augmented by an expert system shell to facilitate the construction of a rule-based forest management system. A general approach to developing an expert system for map design is described by Robinson and Jackson (1985). Follow-up work on a related project by Mackaness and Fisher (1986) shows how some problems of map design can be resolved, and also points out the great complexity involved in the process. Nevertheless, although Fisher and Mackaness (1986) conclude that map design expert systems are possible to construct, they stress that cartographic expertise is both difficult to elicit and specify.

APPROACH TO SYSTEM DEVELOPMENT

In this paper we allocate map components to areas of a viewport in such a way to lead to a maximum, balanced filling of space. To accomplish this task, we use production rules to guide placement of map components (e.g. legends and titles). The production rules are derived from two general sources: cartography textbooks that purport to deal with design, and personal experiences of the authors. From the texts, we incorporated ideas about framing, and allocation of space using the general strategy of "thumbnail sketches" advocated by several authors (Robinson, et al. 1984; Cuff and Mattson, 1982; Dent, 1985). To make the problem manageable each map component is delineated by its extent (bounding rectangle). This approach allows us to deal with the size and location of elements without the excessive amount of computation required for irregularly shaped objects.

We have adopted a three stage approach to system development. The first stage transforms the study area into an abstract representation which can be easily manipulated by declarative programming languages. The second stage arranges the map components within the viewport using a set of rules designed to meet cartographic requirements while avoiding conflict in displaying the map components. The third stage consists of the transformation of the map components from the their abstract representation into a displayable format.

Input to the Process

Two types of input are required at this stage. The first is a set of vector chains (Figure 1) which define the perimeter of the area that must not be overlapped by any other map component (e.g. legend). Each straight line segment is represented as a fact in the knowledge base. The second type of input consists of information about the number, size, and shape of the remaining map components (e.g. legend). To restrict the problem, we assume that the minimum space requirements of each component has been determined at an earlier stage of the layout and design process.

Rasterization of Map Boundaries

Vector format data are not well suited for detecting and evaluating the size and shape of empty spaces where map components can be placed. The use of vector data causes an additional problem because our processes are formulated in a declarative environment. In addition, the chains comprising the boundary of the study area might have superfluous sinuosity; cartographers consider only the general shape and not the details of objects when they make decisions about map layout (Cuff and Mattson, 1982, p. 75). Unnecessary detail can also exhibit more virulent effects associated with increased memory requirements and decreased processing speed. For these reasons, we transform chains to a coarse raster format (Figure 1). The size of the grid depends on the range of the coordinate values and the dimensions of the display medium, and should neither introduce unnecessary detail, nor substantially distort the area to be mapped.

When the cell size has been specified, then determining the cells which correspond to the endpoints of each straight line segment is straightforward. Specifically, the row and column number of each endpoint cell on a staight line segment can be computed using the following formulae:

$$Xci = (int (Xi) \text{ DIV } d) + 1 \qquad (1)$$

$$Yci = (int (Yi) \text{ DIV } d) + 1 \qquad (2)$$

where Xci : the column number of the cell for endpoint i
Yci : the row number of the cell for endpoint i
Xi : the x coordinate of endpoint i
Yi : the y coordinate of endpoint i
d : integer denoting the cell size in the same units as Xi and Yi.

The determination of row and column numbers is iterative, and can be implemented in PROLOG using backtracking and the fail predicate. The results of this process are asserted as facts in the database. Each fact stores the column and row numbers of the endpoints of the corresponding segment. Figure 2 displays how the areas shown in Figure 1 are transformed after the application of this process.

Rasterization of Segments

In order to determine the empty space around the map, all grid cells comprising the boundary of the study area are defined. At this point although we know the grid cells that terminate each straight line segment, the remaining cells along the segment are calculated in a vector to raster conversion. From the various algorithms which rasterize line segments, Bresenham's was chosen because it is efficient and easy to implement. Bresenham's algorithm uses the row and column numbers of the end points of a straight line segment and returns the row and column numbers of the set of grid cells which most closely approximate the line segment (Foley and Van Dam, 1982:432).

We implemented Bresenham's algorithm in PROLOG. The program uses a set of facts which define the endpoints of border segments of the study area, and returns another set of facts which describe the complete set of grid cells bounding the study area. The structure of the new facts is shown in Figure 3, where each fact corresponds to a cell defined by its column (X) and row (Y) number. In addition, each pixel has an attribute indicating whether the corresponding cell is

part of the border of the area to be mapped. All remaining cells are labeled "blank". If, for example, the data in Figure 2 were processed using the algorithm, the outcome would appear as Figure 4.

Layout to Fill the Display Frame

The next steps eliminate empty space surrounding the mapped area, and determine the frame dimensions such that the frame proportionally fits the display area. To achieve this goal it is necessary to determine the extent of the area, which is used as a "core" for any further manipulations (e.g. translation, scaling). If the extent fits in the display frame, and the remaining map components can be arranged in the remaining empty space without violating any cartographic standards, then a satisfactory solution has been identified. Any attempt to further increase the scale of the map would result in crossing the display border and in hiding part of the information from the user. This suggests that it is promising to use the extent of the study area as a starting point for solving the problem of allocating map components in the display.

Given that the area to be mapped has been rasterized, and that the cells defining the border of the area are given as facts (Figure 3), the extent of a picture is determined by scanning all facts which have "pixel" as a functor and "map" as an attribute value and then finding the minimum and maximum row and column number. In Figure 4, for example, the extent is defined by rectangle with corners (2,2) (9,9). Row and column one and ten of that figure do not have shaded cells, and thus can be eliminated without any loss of information (Figure 5). Notice that the remaining rows and columns have been renumbered for convenience. Determining the extent of an object is a double loop iterative process which is implemented in PROLOG by using the fail predicate and backtracking.

After determining the extent of the area, the next step is to place the extent within a frame with dimensions proportional to the dimensions of the viewport. If the dimensions of the extent are not proportional to the viewport, columns or rows must be added to the extent to achieve proportionality. The resulting frame is called the adjusted extent. If required, then the number of columns that need to be added can be computed using the formula:

$$X = (L / W) * w - l \qquad (3)$$

where:

X : is the horizontal extension in grid cell units
L : is the horizontal dimension of the viewport
W : is the vertical dimension of the viewport
w : the number of rows of the extent (vertical)
l : the number of columns of the extent (horizontal)

If X is non-integer the number of columns to be added should be equal to the next largest integer. If X is negative then this means that rows must be added to the extent to achieve proportional dimensions. A similar formula provides the magnitude of the extension along the vertical dimension.

Determination of Space along the Border of the Extent

Up to this point we have transformed the study area into a format which can be easily handled by a declarative language. The study

area also is at the largest scale that the display area permits, and, finally, the area in the configuration is balanced. The next step of the process consists of detecting the empty space surrounding the study area into which the remaining map components can be placed. That empty space is represented in a data structure such that:

-no useful empty space is undetected,
-the identification of size and shape of blocks of empty space is easily determined, and finally,
-translations and scale changes can be made easily using a declarative language.

The detection of useful empty space can be made if we scan the rectangle, which results from the adjusted extent in the four cardinal directions. For each row of a given scan direction, the run length is determined. The run length is defined as the number of empty cells from the frame to the first non-empty cell which is met in the direction of scanning. Scanning Figure 5, for example, from left to right would result in run lengths of: 3 for row 1, 3 for row 2, 3 for row 3, and zero for the remaining rows. Notice that a left-to-right scan does not detect all empty cells (e.g. 2,4). These cells, however, can be detected when the extent is scanned from another direction. Continuing with the previous example, and scanning from bottom to top, the corresponding run lengths are : 3 for column 1, 4 for column 2, 5 for column 3, and zero for all other columns. The empty space at the top and right sides of the extent are detected when Figure 5 is scanned from top to bottom and from right to left. The only empty space which is outside the border of the study area, and is not detected by any scan, is cell (6,4). Such empty space is unlikely to be useful for placing map components.

Since we have identified a way for detecting the location, size, and shape of useful empty space, the next problem that must be addressed is the efficient representation of that information for subsequent manipulations. This problem is solved by using a data structure to store the direction and run length of each swath along each scan direction. A swath is either a row or a column of the adjusted extent depending on the direction of the scan. An abstract data structure which enables the representation of all information related to a scan direction is shown in Figure 6. Specifically, empty space is represented as a compound object, which contains the scan origin, the scan destination, and a list of objects containing information about the swath number and its run length. Figure 6 also displays how the data structure can be implemented as a PROLOG fact. Tables 1.1, 1.2, 1.3, and 1.4 show the information stored in the database if our procedures are applied to the area shown in Figure 5. Each table is a separate fact, and these four facts suffice to store all information needed for evaluating the useful empty space of a configuration.

Geometrical Transformations using the Data Structure

In this section it will be demonstrated how placement of map components can be made using scale change and translation operations and the data structure shown in Figure 6.

Change in scale. In this case we deal only with scale reductions with respect to the frame of an area. Although enlargement can be

treated in a similar fashion we assume that we start with the largest possible scale of the study area and reduce it, until we achieve a satisfactory allocation of map components. The input of the scale change procedure are four facts describing the empty space around the border of the study area. The output of the procedure consists of four new facts describing the empty space at the new scale. Conceptually, reduction with respect to the frame can be made by adding an appropriate number of rows and columns around the adjusted extent. This increase in the frame size allows us to reduce the scale, and helps to avoid difficulties resulting from the raster format of the data.

To keep the dimensions of the frame proportional to the dimensions of the display, we determine the number of rows and columns added at each scale change. If L and W are the horizontal and vertical dimensions of the final display, l and w are the horizontal and vertical dimensions of the adjusted extent, and X and Y are the number of columns and rows that must be added in the frame to reduce scale, then the equation relating X to Y is:

$$X = (L / W) * Y = (l/w) * Y \qquad (4)$$

To overcome problems resulting when X (or Y) is non-integer, it is suggested that Y be chosen first if W <= L. If X is non-integer then the next largest integer is chosen. If, on the other hand, W > L then the value of X should be chosen first. To keep the whole image balanced after the addition of new columns and rows, half of the rows and columns are distributed at the bottom and left sides and the other half at the top and right sides of the image. An illustration of this operation is shown in Figure 7.

After the addition of new rows and columns the empty space and thus the run length of each swath has changed. If XL, XR, YB, YT indicate the number of swaths added to the left, right, bottom, and top sides of the adjusted extend respectively then, for the case of left to right scan, the following rules determine the numbering and the run length of each swath. Rules for other direction scans are specified in a similar fashion.

Left to right scan:

- Add YB elements at the head and YT elements to the tail of the list describing the empty space of the swaths. The swath numbers of the new elements are integers and are selected such that the elements of the list are in ascending order with respect to the swath numbers. The run length of each new swath is set to be equal to XL + l + XR where l is the horizontal length of the frame of the study area before the addition of new swaths.

-For every other element of the list :

If the run length of that element is equal to l
then set the run length equal to XL+l+XR
otherwise set the run length equal to XL+RL
where RL is the old value of run length for
that swath.

If we choose XL=1, XR=1, YT=1, YB=1, l=8, w=8 and we apply the previous rules to Table 1.1, we will get Table 2.1. Applying similar rules corresponding to the other three scan directions to Tables 1.2, 1.3, and 1.4, we obtain Tables 2.2, 2.3, and 2.4. Notice that the last tables describe the useful empty space of the picture displayed in Figure 7.

Translation to the right. Conceptually, translation of the study area to the right with respect of its frame can be accomplished by adding a number of columns at the left and deleting an equal number of columns from the right side of the current frame (see Figures 7 and 8). This operation simply changes the relative position of the objects with respect to the frame. If XL is the number of columns which are added to the left side of the frame and the other variable names have the same interpretation as in 3.6.1, then the rules determining values for the new empty space are:

a Left to right scan.

- For each element of the swath list
  If the run length of the swath is less than 1
  then the new run length is RL + XL
  otherwise the run length remains the same.

b Top to bottom scan.

- Add XL elements with appropriate swath numbers at the head of the list containing information for the run length of each swath. The run length of each new element is w.

- Delete XL elements from the end of the list. Only elements which have run length equal to w are allowed to be deleted. Elements with run length less than w can not be deleted because the corresponding swath contains shaded cells and thus it is part of the boundary of the study area.

Rules applied to the right to left and bottom to top scan are analogous to a and b respectively. Applying such rules to Tables 2.1, 2.2, 2.3, 2.4 and taking XL=1 and w=10, Tables 3.1, 3.2, 3.3, 3.4 are derived. Those tables represent the useful empty space of Figure 8.

Empty Block Size Determination

Information stored in the data structures representing empty space can be manipulated to determine if a map component fits there. If, for example, a 3 by 3 legend must be placed in Figure 5, the solution is to put that legend at the bottom left. That block of empty space can be identified if the values of Tables 1.1 through 1.4 are examined. If the algorithm searches the swaths of Table 1.1 it will identify that there are three consecutive swaths which have run length equal to three -swaths 1, 2, and 3. It then can infer that there is a 3 by 3 square of suitable empty space at the left side of those swaths. If, on the other hand, the dimensions of the legend were 3 by 4, the algorithm will fail to find a solution by traversing Tables 1.1 through 1.4. Since no translation can be performed in Figure 5, the next step is to do a scale reduction. Figure 7 shows the result of such a reduction, and Tables 2.1 through 2.4 represent the empty space of that figure. Searching the swaths in Table 1.1, the algorithm can now identify four consecutive swaths with run length greater than or equal to 3 -swaths 0, 1, 2, 3- and it will come up with an answer. Similar reasoning can be applied for cases where more than one map component must be allocated. In addition to performing list searches, translations, and scale changes, the program keeps track of space reserved for components that have already been allocated.

CONCLUSIONS

In this paper the problem of allocating map components within a viewport has been explored. A three stage approach is adopted to determine the final layout for a simplified study area. The preprocessing stage was the principal topic of our discussion. Data representations and procedures, which allow satisfactory allocation of map components using rules, were described. The effectiveness of the chosen representations is illustrated using examples. The second stage of the approach, which consists of the determination of rules guiding the placement of map components, is currently a focus of our work.

REFERENCES

Broome, F.R., Beard, C., and Martinez, A.A. 1987. Automated map inset determination. Proceedings, Auto-Carto 8, pp. 466-470.

Cuff, D.J., and Mattson, M.T. 1982. Thematic Maps: Their Design and Production. New York: Methuen.

Dent, B.D. 1985. Principles of Thematic Map Design. Reading: Addison-Wesley.

Fisher, P.F., and Mackaness, W.A. 1987. Are cartographic expert systems possible? Proceedings, Auto-Carto 8, pp. 530-534.

Foley, J.D., and Van Dam, A. 1982. Fundamentals of Interactive Computer Graphics. Reading, MA: Addison-Wesley.

Mackaness, W.A. and Fisher, P.F. 1987. Automatic recognition and resolution of spatial conflicts in cartographic symbolisation. Proceedings, Auto-Carto 8, pp. 709-718.

Maggio, R.C. 1987. The role of the geographic information systems in the expert system. Proceedings, GIS '87, pp. 685-692.

Monmonier, M.S. 1982. Computer-Assisted Cartography: Principles and Prospects. Englewood Cliffs: Prentice-Hall.

Morse, B.W. 1987. Expert system interface to a geographic information system. Proceedings, Auto-Carto 8, pp. 535-541.

Robinson, A.H., Sale, R.D., Morrison, J.L., and Muehrcke, P.C. 1984. Elements of Cartography (5th ed.). New York: John Wiley.

Robinson, G. and Jackson, M. 1985. Expert systems in map design. Proceedings, Auto-Carto 7, pp. 430-439.

Figure 1 Study area example

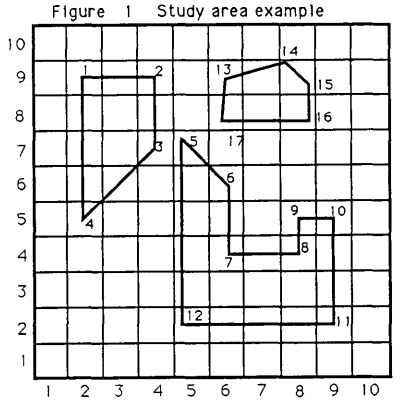


Figure 2 Endpoints of line segments

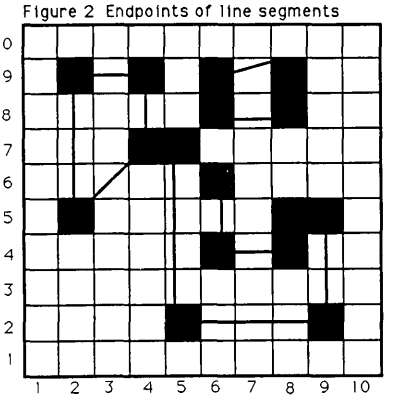
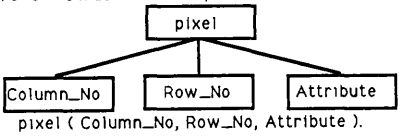


Figure 3 Structure for representation of cells



pixel (Column_No, Row_No, Attribute).

Figure 4 Border of the study area

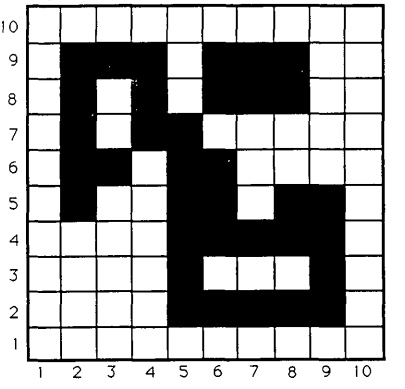


Figure 5 Extent of the area

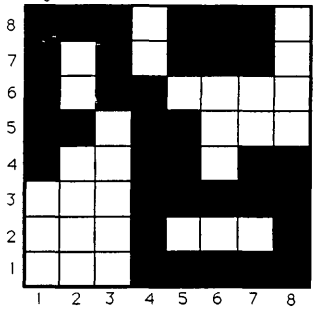


Table 1.1

Scan origin		left							
Scan destination		right							
Swath	Swath_no	1	2	3	4	5	6	7	8
	Run_length	3	3	3	0	0	0	0	0

Table 1.2

Scan origin		top							
Scan destination		bottom							
Swath	Swath_no	1	2	3	4	5	6	7	8
	Run_length	0	0	0	2	0	0	0	4

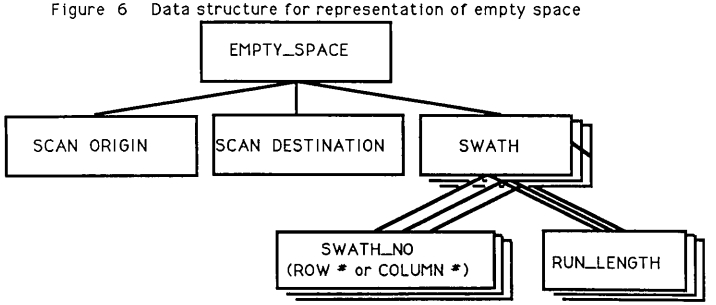
Table 1.3

Scan origin		right							
Scan destination		left							
Swath	Swath_no	1	2	3	4	5	6	7	8
	Run_length	0	0	0	0	3	4	1	1

Table 1.4

Scan origin		bottom							
Scan destination		top							
Swath	Swath_no	1	2	3	4	5	6	7	8
	Run_length	3	4	5	0	0	0	0	0

Figure 6 Data structure for representation of empty space



empty_space(Scan_origin,Scan_destination,[swath(Swath_No, Run_Length)])

Figure 7 Change in scale

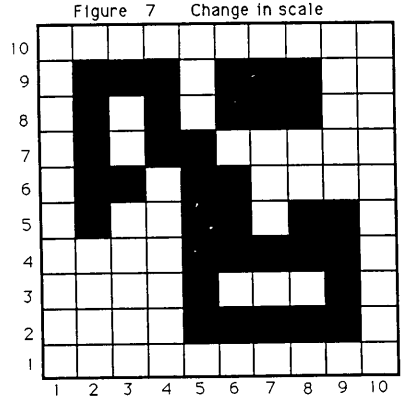


Table 21

Scan origin		left									
Scan destination		right									
Swath	Swath_no	0	1	2	3	4	5	6	7	8	9
	Run_length	10	4	4	4	1	1	1	1	1	10

Table 22

Scan origin		top									
Scan destination		bottom									
Swath	Swath_no	0	1	2	3	4	5	6	7	8	9
	Run_length	10	1	1	1	3	1	1	1	5	10

Table 23

Scan origin		right									
Scan destination		left									
Swath	Swath_no	0	1	2	3	4	5	6	7	8	9
	Run_length	10	1	1	1	1	4	5	2	2	10

Table 24

Scan origin		bottom									
Scan destination		top									
Swath	Swath_no	0	1	2	3	4	5	6	7	8	9
	Run_length	10	4	5	6	1	1	1	1	1	10

Figure 8 Translation to the right

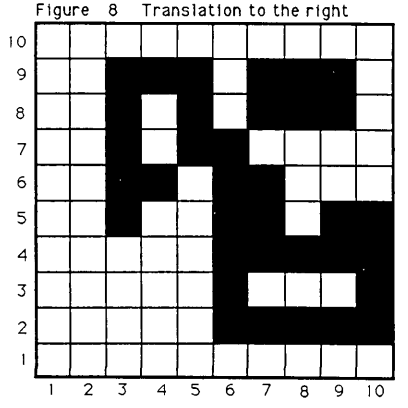


Table 31

Scan origin		left									
Scan destination		right									
Swath	Swath_no	0	1	2	3	4	5	6	7	8	9
	Run_length	10	5	5	5	2	2	2	2	2	10

Table 32

Scan origin		top									
Scan destination		bottom									
Swath	Swath_no	-1	0	1	2	3	4	5	6	7	8
	Run_length	10	10	1	1	1	3	1	1	1	5

Table 33

Scan origin		right									
Scan destination		left									
Swath	Swath_no	0	1	2	3	4	5	6	7	8	9
	Run_length	10	0	0	0	0	3	4	1	1	10

Table 34

Scan origin		bottom									
Scan destination		top									
Swath	Swath_no	-1	0	1	2	3	4	5	6	7	8
	Run_length	10	10	4	5	6	1	1	1	1	1