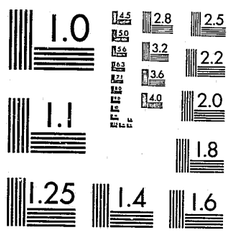


A STRUCTURED GEOGRAPHIC DATA BASE DESIGN TO SUPPORT THE RESOURCE ALLOCATION PROBLEM FOR POLICE DEPARTMENTS

Grant No. 67850274
National Institute of Justice

Robert W. Claire
2/10/82

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Computer usage trends
1.2 Complexities of patrol resource allocation
1.3 Purpose

2 DIGITAL REPRESENTATION OF SPATIAL DATA

2.1 Image data
2.2 Topological data
2.3 Measurement data
2.4 Attribute data

3 DATA BASE DESCRIPTION

3.1 Overview
3.2 Image data
3.3 Files for individual overlays
3.3.1 Node files
3.3.2 Arc files
3.3.3 Polygon files
3.4 Police-related data set
3.4.1 CFS-related files
3.4.2 Patrol unit-related files
3.5 Summary

4 ALGORITHMS FOR FUNDAMENTAL DATA MANIPULATIONS

4.1 Retrieving image data
4.2 Distance between two points
4.3 Arc length
4.4 Rectangle search
4.5 Intersection between two line segments
4.6 Area of polygon
4.7 Point in polygon
4.8 Arc coding of events by address
4.9 Polygon coding of arcs

5 SUBSCHEMA FOR APPLICATIONS PROGRAMMING

5.1 CFS-related processing
5.2 Hypercube queuing model
5.3 Patrol car allocation model
5.4 Extended applications
5.4.1 Graphic display
5.4.2 Network analysis
5.4.3 Statistical analysis

6 CONCLUDING REMARKS

REFERENCES

APPENDIX A: Computer usage by police departments
APPENDIX B: Spatio-temporal variations in offense levels
APPENDIX C: Overview of the DIME system
APPENDIX D: File structure for GIRAS
APPENDIX E: Overview of Patrol Car Allocation Model
APPENDIX F: Overview of Hypercube Queuing Model

1 INTRODUCTION

1.1 COMPUTER USAGE TRENDS

Surveys of computer usage by police departments have monitored past developments and have attempted to project trends for the future. As might be expected, the once manual activities that incorporated 'cut and dried' decision rules and operations were the first to be adopted to the algorithmic environment of data processing. Colton [9,10,11], in compiling time series data on police computer trends, shows that initially usage was biased strongly toward the more structured or more routine applications areas. (See Appendix A)

The activities identified as routine applications included real-time inquiries of people and property; automated record keeping related to traffic accidents, citations, and parking violations; and the processing of personnel records, payrolls, budgets, and inventories. To the extent that crime statistical analysis is used to generate and report standard statistics, it also can be considered routine. However, activities such as computer-aided dispatching, criminal investigations, and resource allocation are considered non-routine. No well defined set of procedures is available for their implementation. The computer, in respect to non-routine application areas, can only support the decision maker by automating the straightforward and iterative aspects of the operation.

Routine application areas continue to dominate the computer usage pattern. With these activities in place, however, research and development is shifting toward the non-routine application areas. One non-routine application area of particular interest is resource allocation. Separate surveys in 1971 and 1974 indicate that resource allocation was considered by police departments to be the most important application area for computer resources. Moreover, it was the only application area where developments by 1974 exceeded predictions of earlier years.

Increased interest in resource allocation is understandable in light of the fiscal situation confronting agencies of local government. Police departments are faced with increasing demands for their services with resources equal to, if not less than, those of previous years. The costs of fielding an emergency response unit on a 24-hour basis is over $100,000 per year. Increasing manpower levels in response to increasing demand levels is becomming less and less of an alternative.

1.2 COMPLEXITIES OF PATROL RESOURCE ALLOCATION

Resource allocation, in a general form, is the matching of a supply of a good or service with its demand according to some optimization function. In the context of police operations, resource allocation generally refers to the determination of how many patrol units to assign to what time periods and to what subareas of the city. The task of allocating patrol units is complicated by objectives that do not lend themselves to a single optimization function and by the dynamics that characterize demand.

Demand for patrol resources is both time and space dependent. A primary function of the patrol unit is to respond to calls-for-service (cfs) made to the department. Cfs levels are not uniform over time; rather cyclic patterns are apparent where the level of a particular offense type is relatively high for some time periods and lower for others. Nor are cfs levels uniform over space. City-wide distributions of offense types yield different spatial patterns. Certain subareas of the city tend to experience a disproportionate share of the offense activity, while other areas are characterized by comparatively low levels. (The spatial and temporal dynamics of cfs activity for an urban area are considered further in Appendix B.)

Fundamental to the patrol service delivery system is the decentralization of resources. The city is subdivided into divisions, which are comprised of precincts. Precincts, in turn, are subdivided into patrol areas. Units are detailed to areas for portions of the day, or tours of duty. The terminology may vary, but the construct is common. Flexibility is introduced by assigning overlay tours and special purpose units to subareas of the city, and by altering the extent of the area covered by individual units.

Consider the sequence of events that are associated with the servicing of a call-for-service (figure 1.01). At T(0) a call is reported to the police. Departmental staff obtains relevant information and alerts the dispatcher. The dispatcher determines which patrol unit is responsible and its disposition. If the primary unit is available, it is dispatched to the scene at T(d). At T(a), the unit arrives at the scene and, at T(b), it reports back-in-service.

If the primary responding unit is unavailable, the dispatcher may refer the cfs to an available unit in an adjacent patrol area. Alternatively, the cfs can be placed in a queue at T(q), and dispatched when the unit becomes available. The preferred action requires a comparison of the expected queuing delay and travel time of the primary unit with the expected increased travel time of the secondary unit. Dispatch policies vary among departments and, to a large extent, are dependent upon the nature of the individual situation.

Response time refers to the interval between the time that a cfs has been processed by the department and the time that the unit arrives on the scene. Two components comprise response time-- the time the cfs may sit in a queue and the time required for the unit to travel to the scene. The interval from the point that it arrives on scene to the point it reports back in service is referred to as on-scene time. The interval comprised of the travel time and the on-scene time is termed the service time.

The spatial configuration of the delivery system has a direct bearing upon these temporal components. All else being equal, the more compact the patrol area, the lesser the expected average travel time.

Travel time is directly related to travel distance by the elementary laws of physics. And average travel distance has been empirically shown to increase with incresed service area size. A good approximation of this relationship is given by the square root law [4,16,17].

A reduction in expected travel time will yield a decrease in expected in-service time. Consequently, the patrol unit is more available, and the likelihood of a cfs being placed in queue is less.

Unfortunately, all else is rarely equal. Consider the hypothetical situation in figure 1.02 where the assumption of a uniform cfs distribution is relaxed. Delineating the region into two patrol areas as shown in figure 1.02(a) may yield equal travel times in each area. However, one area experiences a higher cfs rate; and the partol unit in that area is busier and less likely to be available. Delineating the region as shown in figure 1.02(b) may yield equal workloads, but travel time in the larger area is greater. In a sense, the 'safer' area is being penalized.

Thus far, the discussion has focused upon the reactive aspect of patrol services. The patrol unit has a dual function; besides responding to cfs's, the unit serves a proactive role by patrolling an area as a deterrent to potential criminal activity. Generally, when it is not servicing a cfs (and not out-of-operation), a unit is assumed to be in a preventative patrol mode. Thus any increase in a unit's in-service time has a negative effect upon its preventative patrol capabilities.

Various perfomance measures are employed to evaluate the characteristics of police patrol delivery systems; these relate to workload levels, response times, queuing delays, and the like. Strong interdependencies exist between the measures and often the desire to optimize one measure has a negative effect on others.

It is the role of the administrator to consider equity versus efficiency issues, various tradeoffs among performance measures for alternative configurations, and draw up a plan that is appropriate for the jurisdiction. The researcher's contribution is the development of resource allocation programs that effectively model the real world, with all its complexities and interdependencies, such that the impact of contingency plans on various aspects of the delivery system can be evaluated.

1.3 PURPOSE

Past developments have seen police resource allocation models increasing in sophistication. The oversimplistic hazzard formula approach has given way to measures of performance characteristics of the delivery system based upon stochastic processes, queuing theory, and the like. The role of the computer is to support the analytical manipulations of large volumes of data which characterize existing models and which is expected to characterize models developed in the future.

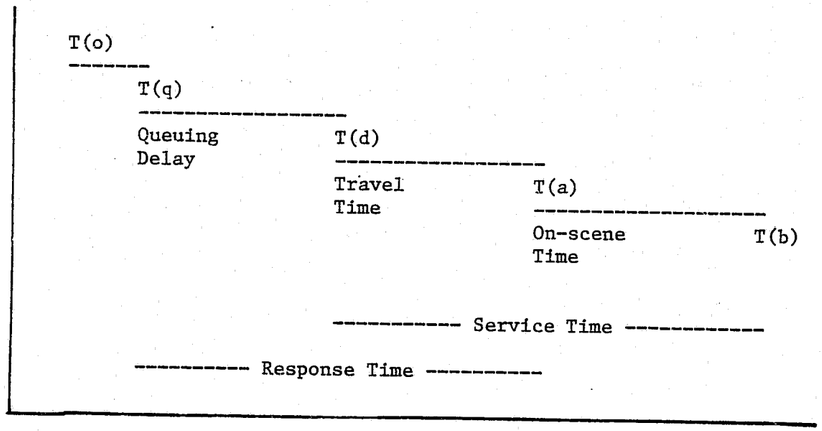
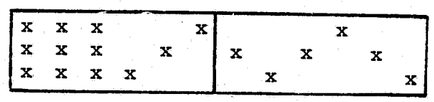
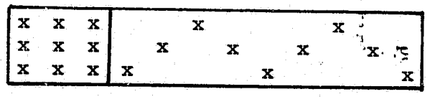


Figure 1.01



(a)



(b)

Figure 1.02

In addition, the computer can provide the access to the wide number of variables inherent in resource allocation. With this increased sophistication in resource allocation models is an increase in data requirements. Required data elements include not only police-related data, but extends to information on the demographic, socio-economic, geographic, and physical characteristics of the urban area.

Without an effective data base structure, in conjunction with software for retrieval and management, weeks or months may be devoted to generating the input requirements to support a model that runs in minutes. Data files currently retained by police departments are primarily event-specific. An offense file, for example, contains records that record attributes of individual offenses. In contrast, input data requirements for resource allocation models are, to a large degree, summary or aggregate in nature. To compile say 'average on-scene time' for cfs's located in a particular patrol area may involve the processing of thousands of cfs-specific records. Computing 'average on scene time' by patrol area and hour, for each 24-hour day in the input file, compounds the amount of preprocessing substantially.

Moreover, files that are comprised of aggregate or summary-type data elements are 'locked' to the parameters employed in their generation. Changes in any one of the parameters may render files generated after the change incompatible with previously generated files. This problem becomes painfully obvious to a planner when attempting a comparative analysis of data sets compiled for different patrol area configurations, workshift structures, cfs classification schemes, and the like. The planner is confronted with the alternatives of recompiling the source data to a common framework, or dispensing with the analysis.

The purpose of this report is to propose a geographic data base design that will support much of the data processing requirements necessary for resource allocation programming. The geographic base file provides a common spatial referencing scheme for the various data sets that may serve as source information to the analysis. The locational aspect of the data is explicitly encoded, from which spatial relationships among data elements may be derived and employed to accommodate queries and processing based upon geographic parameters. Such capabilities are fundamental to resource allocation modelling— the compilation of statistics for areal units, the accessibility between locations, the comparison of activity levels for different areas, the relating of offense activity to other attributes of an area, and the like.

The geographic data base design is an important step for the consideration of a data base management system in police operations. With the automated capability for deriving input requirements for applications programs (eg resource allocation models), the need is lessened for storing multiple input files that represent various contingencies (and weeks of compilation). Prior to running the model, control parameters are employed to retrieve the prerequisite source data and derive the input requirements in the form of a subschema. The control parameters define the spatial, temporal, and typological ranges for the data to be included in the analysis at hand. It is this type of efficient, ad hoc data access that affords the flexibility to run the model for alternative situations.

Also lessened is the 'domino effect' of updating, where editing the source data yields embedded inconsistencies in archived aggregate data files. Since the source data files are referenced prior to any model run, any modifications are reflected in the analysis.

Software development is made easier. The burden of retrieving data elements in a particular form is turned over the management software. The applications program need only focus upon the manipulations directly related to the analysis. Moreover, the added layer of insulation represented by the management software ensures data independence. Thus changes can be made to the data base or the applications software without effecting major modifications in the other.

The geographic data base design draws from existing technology; primarily the DIME system of the Bureau of the Census and GIRAS of the U.S. Geological Survey. The DIME system is based upon the network structure that characterizes a city's street system. The street system is an instrumental component in the police delivery system in more than just a transportation sense. The street addressing scheme is the basic manner for referencing location within the urban area. The DIME system serves as an instrumental tool for referencing address encoded data in a form compatible with mathematical manipulation and automated processing. GIRAS, which adopts many of the constructs of DIME, is a polygon based system. To the extent that resource allocation models reference data aggregated by areal units, many of the underpinnings of GIRAS are applicable. (DIME and GIRAS are discussed further in Appendicies C and D.)

For the purposes of this report, resource allocation programming is represented by the Patrol Car Allocation Program (PCAM) and the Hypercube Queuing Model. PCAM addresses the issue of determining the number of units to assign to a region, whereas Hypercube is concerned with delineating patrol area boundaries within a region. They represent the most recent developments in resource allocation modeling and incorporate many of the factors discussed above. (PCAM and Hypercube are descrived in more detail in Appendicies E and F.) The capabilities of the geographic data base to support subschema for resource allocation applications is demonstrated for PCAM and Hypercube. The system is not limited to these two models, and extended applications are described.

The geographic data base design, together with the discussion of access and manipulation algorithms, should provide police departments with valuable insights to data base management systems for police planning.

The description of the data base design requires some understanding of basic terms and constructs of spatial data. These are outlined in the following section, the digital representation of spatial data. With this foundation, the description of the data base design is presented in Section 3.

Section 5 describes the manner in which the data base supports the input requirements for patrol resource allocation models. In addition, some

logical extensions to applications programming are discussed.

The generation of input requirements descibed in section 5 make references to a set of algorithms described in section 4, which are considered to be fundamental and common to various data manipulations.

2 DIGITAL REPRESENTATION OF SPATIAL DATA

What characterizes data as 'spatial' is the integration of location identifiers to support a two-dimensional model of reality. The basic problem is that of defining " a two dimensional continuous view of reality within the discrete and sequential computer storage" [28]. Four types of location identifiers have been defined [29] which represent different approaches to the problem (Figure 2.01):

(a) external indexing
(b) coordinate reference
(c) arbitrary grid
(d) explicit boundary

External indexing is a nominal type coding scheme that labels a feature according to some external referencing system. The code serves only to link the feature to its representation on a separate source document (i.e. base map). Other than grouping elements, the external index has little manipulative capabilities. PCAM, for instance, makes use of an external indexing scheme (precinct id) to group records in the input file structure and tabular output. External indexing is easy to implement, which is the reason it is employed by many police resource allocation programs; but it is very limited for supporting queries on relative location and other spatial relations.

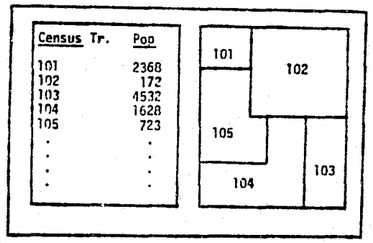
The Hypercube model also employs an external referencing scheme in the form of id's for geographic atoms, but extends the input requirements to include coordinates for internal points. The representation of areas by internal points is a form of coordinate referencing which Hypercube uses to compute inter-atom distances. Coordinate referencing allows for limited manipulation, but, as with external indexing, requires referencing a base map to detemine area boundaries.

A common method for explicitly encoding the spatial extent of areal distributions is the arbitrary grid. Grid cells of some appropriate size and orientation are superimposed over the distribution and coded accordingly. Because of the uniformity in the grid configuration, it is efficient for the discrete and sequential characteristic of both computer storage and processing. Moreover, implicit in grid indexing is relative location. Depending upon relative grid size, the arbitrary grid method is a generalization of the areal distribution. Boundaries are not explicitly encoded, but spatial extent is represented.

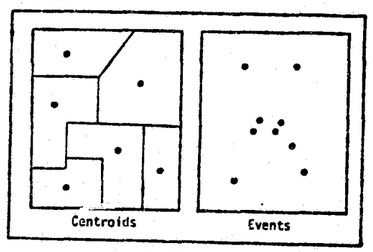
Addressed in detail in this report is the explicit boundary approach. It is the most demanding of the approaches in terms of data requirements and structuring, but offers the most accurate representation and greatest potential for supporting spatial processing. The explicit boundary approach is expanded beyond image data to include discussions of topological relationships, measurement data, and attribute data.

2.1 IMAGE DATA

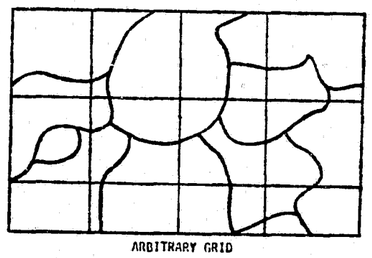
All the features on a map are reducible to basic geometric elements--



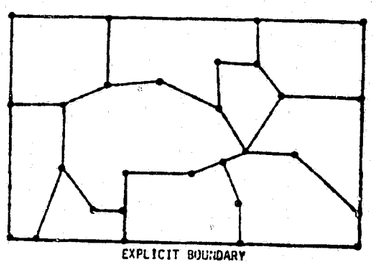
EXTERNAL INDEX



COORDINATE REFERENCE



ARBITRARY GRID



EXPLICIT BOUNDARY

Figure 2.01

LOCATION IDENTIFIERS FOR GEOGRAPHICAL INFORMATION SYSTEMS

Source: Tomlinson, R.F., (ed.), 1972. Geographic Data Handling, Vol. 1, I.G.U. Commission on Geographical Data Sensing and Processing, Ottawa, Canada.

points, lines, and areas. And it is this decomposition that is fundamental to converting a map image to machine-readable form. Geometry provides the elements and also a referencing scheme for defining location.

A map of a city can be treated as a cartesian plane with two orthogonal axies (X and Y). Select an arbitrary point on the map, and its location can be precisely defined by a pair of (X,Y) coordinates. (Figure 2.02)

A line segment, in theory, consists of an infinite number of points that run between two end points. Its location can be specified by recording the coordinates for the end points, and assuming the intermediate points. Given any value for one coordinate, the value of the other coordinate is easily computed from the equation for a line. (Figure 2.03)

Consider now the linear feature in figure 2.04 . Recording just the end points yields one image that does not resemble the image of the original feature. As is easily recognized, however, the feature can be decomposed into smaller straight line segments. By recording the coordinates for the vertices along the feature, and assuming intermediate points, the feature's location is defined.

In some situations, the location of a linear feature cannot be defined precisely, only approximated. Figure 2.05 illustrates just such a case. There are no obvious vertices or line segments about which to decompose the image. Figures 2.05 and 2.06 illustrate the digital approximation of the linear feature as first it is treated as two segments, then four, and so on. Regardless of the number of segments in which the linear feature is subdivided, the result is only an approximation; although the approximation increases as the number of intermediate points increase. This problem is resolved by accepting some level of approximation or resolution as adequate for the purposes at hand.

Areal features are characterized on maps by some indication of the area's spatial extent. This may be a boundary line, a break in color or shading pattern, or the like. The boundary of an area is actually a linear feature, and its location can be defined in a similar manner.

Because the areal unit is a bounded region, its boundary is closed; and any arbitrary end points will coincide. (Figure 2.07)

A boundary actually subdivides a region into two areas. For referencing purposes, some inidication is needed to specify which area is being addressed. This is commonly accomplished by recording the coordinates of an arbitrary internal point or 'centroid'.

Consider now the areal configuration in figure 2.08. Also suppose that individual areas are represented in digital form by single closed arcs representing each area's boundary. The hazards of such an approach are both data redundancy and possible image noncompliance.

All of the image data is essentially stored twice in the digital file. Coordinates for each boundary segment between two areas appear in the

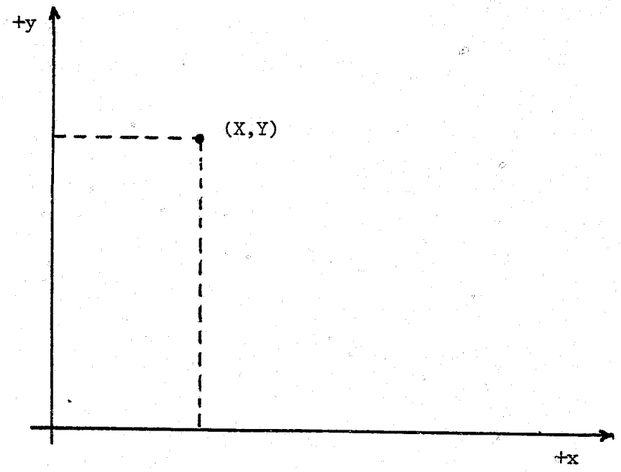


Figure 2.02

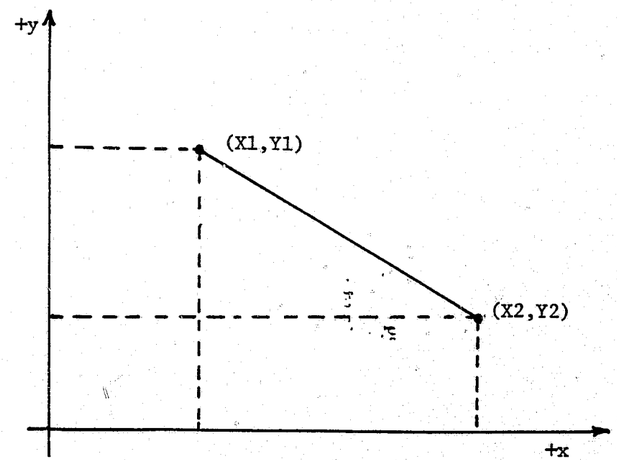


Figure 2.03

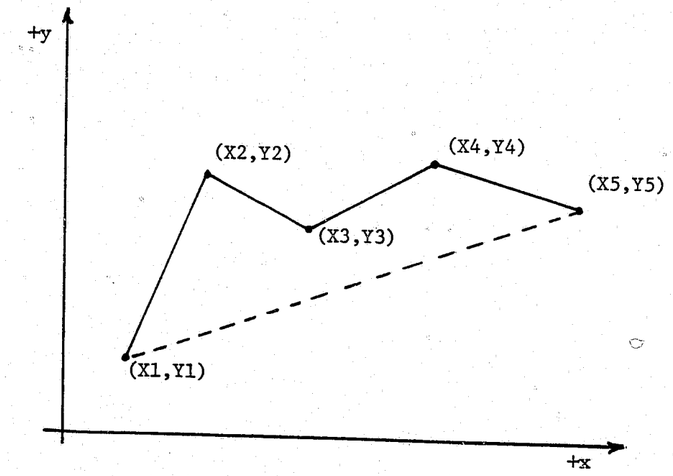


Figure 2.04

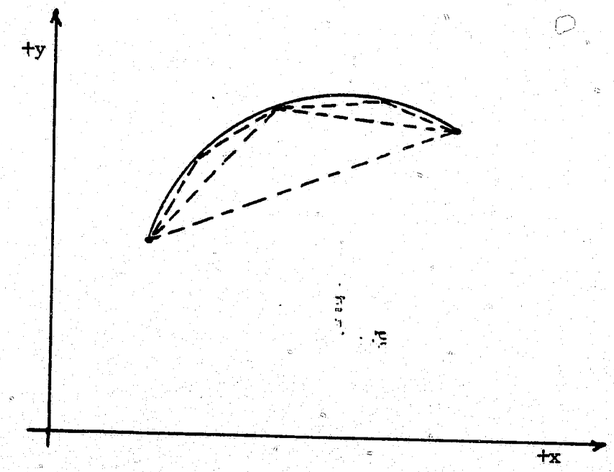


Figure 2.05

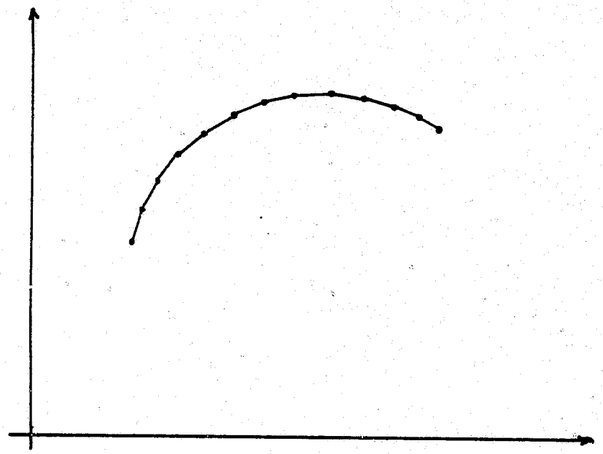


Figure 2.06

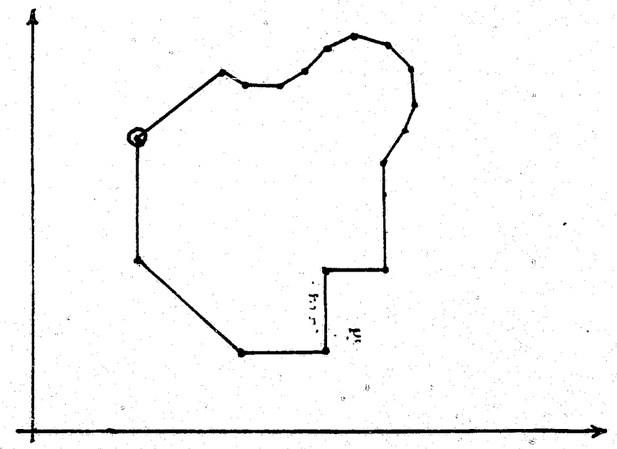


Figure 2.07

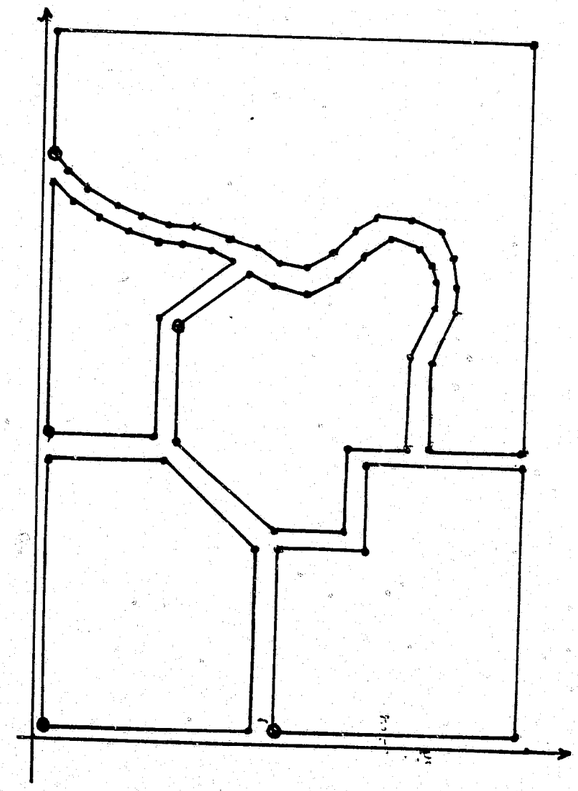


Figure 2.08

coordinate strings for each of the areas.

Moreover, if the boundaries were digitized independently for each area, there is likely, because of limitations on digitizing procedures, to be gaps, slivers, or overlaps in the final image. (Figure 2.09)

This problem of data redundancy and noncompliance can be resolved by utilizing the hierarchial structure of geometric elements. That is, points define linear features, and linear features define area boundaries. To implement this structure, the following terms are defined:

node : any point of intersection or termination
arc : the linear feature running between two nodes
polygon: the uninterrupted area bounded by arcs

Figure 2.10 illustrates the configuration in figure 2.08 expressed in terms of nodes, arcs, and polygons. Three files can now be defined to capture the configuration in digital form. A node file that stores a node id and the (x,y) coordinates for the node. An arc file that stores an arc id, the number of points in the arc, and the coordinate string definining the arc's location. And a polygon file that stores a polygon id, the number of arcs comprising its boundary, and a list of the respective arc id's. Note that the image is stored once.

2.2 TOPOLOGICAL DATA

Whereas image data is concerned with the absolute location of features, topological data explicitly encode the location of a feature relative to other types of features. The utility of topological data was illustrated above. The polygon file is a topological file that defines the relationship between a polygon and adjacent arcs. Topologically structuring spatial data serves many purposes-- ensuring data independence, reducing data redundancy, supporting automated quality control checks, and allowing for spatial entities to be addressed directly and manipulated.

Topological data elements can be defined for all the interrelationships between nodes, arcs, and polygons:

- Which arcs comprise the boundary of a polygon.
- Which nodes comprise the boundary of a polygon.
- Which arcs adjoin a node.
- Which polygons adjoin a node.
- Which nodes provide end-points to an arc.
- Which polygons are separated by an arc.

The nature of data processing intended for the data should define which topological data elements are required and which are extraneous.

2.3 MEASUREMENT DATA

Measurement data are those data elements whose values are in some way

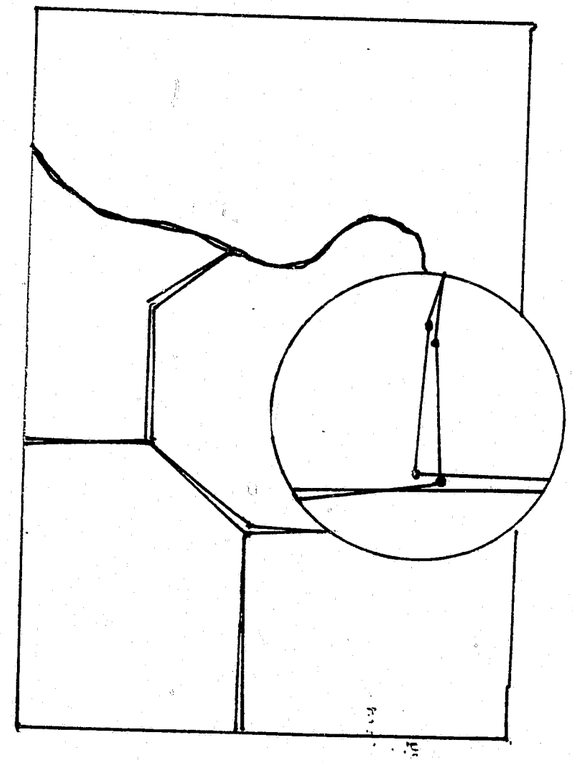


Figure 2.09

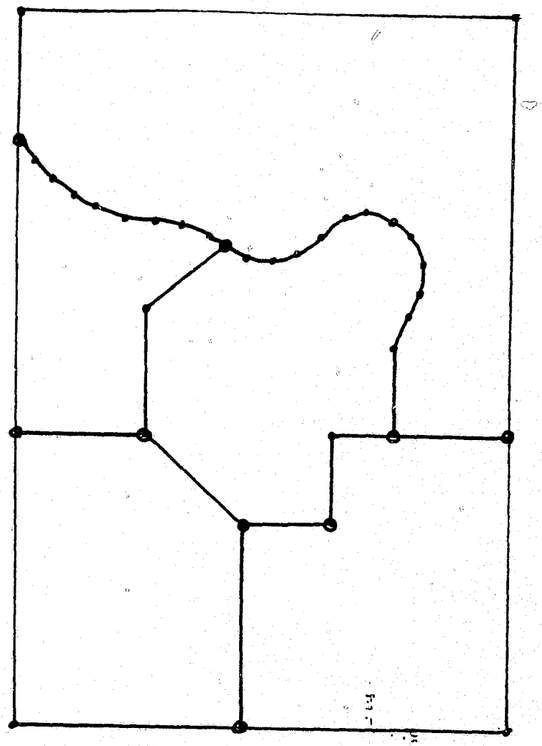


Figure 2.10

derived from the image data, and because they are referenced so frequently, are stored explicitly.

For instance, values for areal units (eg population, cfs levels, etc.) are often expressed in terms of density (eg population per square mile). Density measures require repeated computations involving area. Given the coordinates for a polygon boundary, the area of the polygon can be computed. But rather than make this computation each time an area value is required, it may be more efficient to compute the area once and store it as a permanent data element.

Another common measurement data element is bounding rectangle (Figure 2.11), which is instrumental in various spatial search routines. That is, the minimum and maximum X and Y coordinates for a feature, which define a rectangular hull about the feature. These data elements can be derived once from the coordinate string associated with a feature, and referenced repeatedly for spatial search operations.

2.4 ATTRIBUTE DATA

Attribute data are those elements associated with nodes, arcs, or polygons, that characterize the feature in a qualitative or quantitative way, and that are obtained from an information source external to the spatial data. For example, a digital representation of patrol areas would include a file(s) indicating the patrol area id, together with statistics (eg cfs levels) compiled externally for the polygon. It is the attribute data that draws the association between the digital map data as an abstract and reality. Without it, nodes, arcs, and polygons are without meaning or relevance.

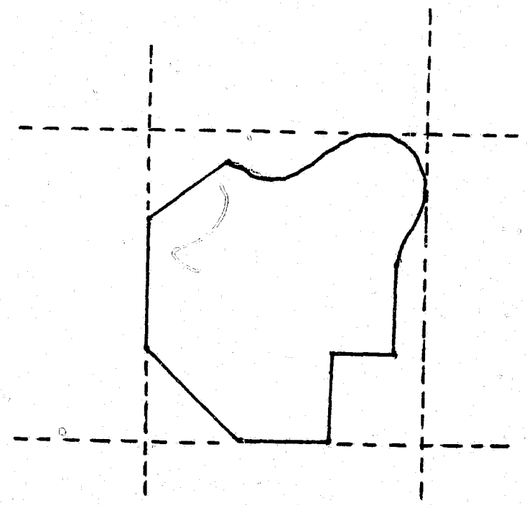


Figure 2.11

3 DATA BASE DESCRIPTION

3.1 OVERVIEW

Presented here is a data base design to support resource allocation programming by police departments. The data base approach represents an alternative to the generation of input files for single purpose runs, that will reduce data redundancy and ensure greater consistency in results. Moreover the data base design incorporates a structured geographic base data set (gbds) that explicitly handles spatial data. Given this gbds, a large degree of flexibility is possible for all phases of the analyses-- the generation of subschema for applications programming, the scope of the analysis, and the manner in which the output is presented. The structure of the data base is not trivial and would require a reasonable investment to implement; but its advantages to resource allocation, and to a broader range of potential applications, both within and outside the police department, make it worth considering.

The data base is comprised to two components-- a police related data set that retains information concerning calls-for-service (cfs) and patrol units, and the gbds which provides a spatial structure with which to locate and manipulate the police related data (figure 3.01). It is possible to integrate other data sets (eg census data, land use data, etc.), and this would seem highly desireable; but the design process here focuses upon the primary objective of supporting patrol resource allocation models.

The gbds is comprised of a set of quasi-autonomous overlays representing different categories of information. Those described at length are the city's street system and administrative areas used by the police department. Each overlay represents an integrated set of arc, node, and polygon files that retain image, topological, measurement, and attribute data. Data files for individual overlays do not contain image data per se, rather they contain pointers to a single image data set.

The police-related data set is restricted to data elements fundamental to the supported resource allocation models. It is recognized that additional data elements may be available.

File structure designs to store cfs-related and patrol unit-related information vary from department to department. The file structure as described in this report for police-related data is intended for reference purposes, and is not critical to the overall gbds design. That is, the processing of police related data is done independently of the gbds. What is critical is some structure that retains the specified data elements and supports the processing capabilities outlined.

3.2 IMAGE DATA

The image data for all overlays in the gbds are stored in a single set of strand and vertex files. Strands and verticies are comparable to

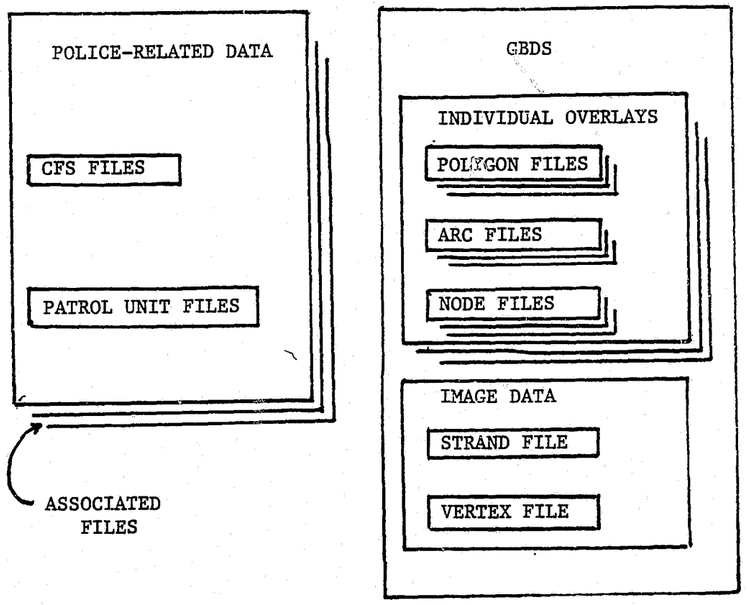


Figure 3.01

OVERVIEW OF DATA BASE

arcs and nodes within an overlay, but the distinction is made for three reason: their primary purpose to retain a single copy of the image data for multiple overlays; their independence from higher topological, measurement, and attribute constructs defined for inidividual overlays; and their representation as a common image element for locationally coincident features.

Consider a base map upon which all the overlays are compiled. For our purposes here, the base map portrays the street network together with the boundaries for police administrative areas at some level. (Figure 3.02)

All the points of intersections and terminal points are assigned vertex identification numbers for internal processing. Similarly, all the line segments between verticies are assigned identification numbers. From this composite image configuration, a strand and vertex file are defined.

Records in the vertex file simply contain the vertex id and its absolute location:

```
VID X Y

where  VID : vertex id
       X,Y : coordinates for the vertex
```

Records in the strand file contain the number of points defining the location of the strand followed by the actual coordinate string. In addition, the id's for the beginning and end verticies are specified. The format is given as follows:

```
SID BV EV NC XY(1) XY(2)....XY(NC)

where  SID : strand id
         BV : vertex id for beginning of strand
         EV : vertex id for end of strand
         NC : number of points in coordinate string
      XY(i) : ith coordinate pair in string
```

The specification of BV and EV give the strand an implicit direction, which represents the direction in which the strand was digitized. Although this direction is arbitrary, it is instrumental for internal processing, as will be shown in later sections.

3.3 FILES FOR INDIVIDUAL OVERLAYS

Each overlay (ie street system, police districts, precincts, patrol areas, and reporting areas) employs an integrated set of arc, node, and polygon files to store appropriate topological, measurement, and attribute data. What specific feature an arc, node, or polygon represents depends upon the individual overlay. Just as was done for the composite image data, consider now a base map for a particular overlay. Nodes represent the points of intersection and terminal points. Arcs represent the linear segments that run between nodes.

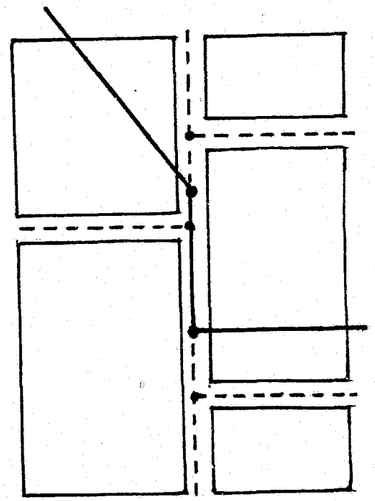


Figure 3.02

(Figure 3.03)

3.3.1 NODE FILES

The only node related data required are the absolute location of the nodes. This information, however, is available in the vertex file of the image data. Thus, a file could be proposed specifying node id's and corresponding vertex id's. Such a file would be extraneous since the vertex id's can be referenced directly. Thus wherever node id's are referenced, the value can be that of the corresponding vertex id.

3.3.2 ARC FILES

The arc files store topological, measurement, and attribute data relating to arcs. All overlays have similar topological file; the need or nature of measurement and attribute files, however, depends upon the individual overlay.

Pointers defining the topological relationship between arcs and relevant nodes and polygons is given in the arc.topo file, which has the following format:

```
AID BN EN LP RP

where  AID : arc id
        BN : id for beginning node
        EN : id for end node
        LP : id for polygon to left of arc
        RP : id for polygon to right of arc
```

As with strands in the image data, the arc assumes an implicit direction by the specification of a beginning/end and left/right terms. The direction is purely arbitrary, but, as will be shown, is instrumental in internal processing.

Measurement data is stored in an arc.meas file. Arc measurement data is consists of two items-- arc length and bounding rectangle coordinates. The following format is given:

```
AID ALNGTH  XMIN YMIN XMAX YMAX

where  AID : arc id
    ALNGTH : length of the arc
      XMIN,YMIN,XMAX,YMAX:  coordinates for bounding rectangle
```

Attribute data for arcs is required only for the street overlay. An arc.addr file is defined which provides an instrumental link between the topological representation of the street network and the city's addressing scheme. The format of the arc.addr file is given as follows:

```
AID STNAME LOLFT HILFT LORGT HIRGT

where  AID : arc id
```

OVERLAY FILES

PID	NA	AID's ....

PID	PAREA	XMIN	YMIN	XMAX	YMAX

PID	PCODE

PID	(X,Y)

AID	BN	EN	LP	RP

AID	ALN	XMIN	YMIN	XMAX	YMAX

AID	STNAME	LOLFT	HILFT	LORGT	HIRGT

AID	NS	SID's ....

NID	VID

IMAGE FILES

SID	BV	EV	(X,Y)'s ....

VID	(X,Y)

Figure 3.03

STNAME : street name
LOLFT,HILFT : address range for left side of arc
LORGT,HIRGT : address range for right side of arc

The arc.addr file is similar to the DIME structure and provides the interface between the gbds and the police related data. (See Appendix C)

3.3.3 POLYGON FILES

As with arcs, polygon files store topological, measurement, and attribute data relating to areas.

A poly.arc file is defined for each overlay which specifies the arcs that comprise the boundary of each polygon (similar to GIRAS, see Appendix D). Its format is as follows:

PID NA AID(1) AID(2) ... AID(NA)

where PID : polygon id
NA : number of arcs that comprise the boundary
AID(i) : the ith arc id in the polygon list

The AID's are ordered such that they chain sequentually about the polygon. A '+' or '-' indicates whether the direction of the chaining corresponds to the direction of the arc.

A poly.meas file is defined for each overlay which specifies the area of the polygon and its bounding rectangle. The format of the poly.meas file is given as follows:

PID AREA XMIN YMIN XMAX YMAX

where PID : polygon id
AREA : area of the polygon
XMIN,YMIN : minimum X and Y coordinate values
XMAX,YMAX : maximum X and Y coordinate values

A poly.attr file is defined for each overlay relating to police administrative areas which specifies the department's internal identification code for the area (eg patrol area id, district name). Its format is given as follows:

PID PCODE

where PID : polygon id
PCODE : department's respective identification

3.4 POLICE RELATED DATA

The police related data is basically comprised of two sets of data files— those files relating to calls-for-service (cfs), and those files

relating to patrol units (figure 3.04).

3.4.1 CFS RELATED FILES

CFS files specify fundamental attributes for individual calls; this includes the cfs type, the location of the cfs, when it occured, and temporal characteristics of responding units.

A cfs.addr file is defined to specify the location of the cfs. The format is given as follows:

CID ADDR STNAME

where CID : cfs id
ADDR : house number portion of address
STNAME : street name portion of address

A cfs.type file is defined to specify the type of call (eg armed robbery, aggravated assault, auto theft, etc.). Its format is simply given:

CID TYPE

where CID : cfs id
TYPE : cfs type code

Temporal data associated with a call is given in the cfs.time file, which has the following format:

CID DATE DOW TOCCR TCALL

where CID : cfs id
DATE : date of call
DOW : day of week code
TOCCR : time of day cfs estimated to occur
TCALL : time of day cfs reported

The cfs.resp file specifies the temporal characteristics of responding units, and is given in the following format:

CID NUR UID(1) TDISP(1) TARRV(1) TBIS(1) UID(2).....
....UID(NUR) TDISP(NUR) TARRV(NUR) TBIS(NUR)

where CID : cfs id
NUR : number of responding units
UID(i) : id of ith responding unit
TDISP(i) : time of day ith unit was dispatched
TARRV(i) : time of day ith unit arrived on scene
TBIS(i) : time of day ith unit was back in service

The cfs.resp file represents a key link between the cfs files and the unit file.

3.4.2 PATROL UNIT FILES

CFS-RELATED FILES

CID	ADDR	STNAME

CID	TYPE

CID	DATE	DOW	TOCCR	TCALL

CID	NUR	UID	TDISP	TARRV	TBIS

PATROL UNIT-RELATED FILES

UID	PA	BDATE	EDATE

UID	SUT	MOT	TUT	WET	THT	FRT	SAT	BDATE	EDATE

UID	NRA	RA	%PT	BDATE	EDATE

UID	DATE	ACCT	HR ....

Figure 3.04

Patrol unit files define required data elements for individual units.

A unit.area file specifies the patrol area id assigned to each patrol unit, and is given in the following format:

UID PA BDATE EDATE

where UID : unit id
PA : patrol area id
BDATE : beginning date when the assignment was operative
EDATE : ending date when the assignment was operative (if not current)

A unit.shift file defines the work shifts to which a unit is assigned, and has the following format:

UID SUT MOT TUT WET THT FRT SAT BDATE EDATE

where UID : unit id
SUT-SAT : shift assignment ids for days of week
BDATE,EDATE : operative dates for assignment

The unit.shift file defined above is based upon a weekly cycle. This may vary by department, but can be accomodated in a similar structure.

The spatial pattern of preventative patrol operations is defined in a unit.prev file, with the following format:

UID NRA RA(1) %PT(1) RA(2)....RA(NRA) %T(NRA)

where UID : unit id
NRA : number of reporting areas patrolled by unit
RA(i) : id of ith reporting area
%PT(i) : measure of percent of nonbusy time spent in reporting area
BDATE,EDATE : operative dates for assignment

Information is required as to the types of work in which individual patrol units spend their time. This is ultimately used to determine the effective number (vs actual) number of units. A unit.accnt file is defined here with a similar format to a time card:

UID DATE NAC ACCT(1) HR(1) ACCT(2) HR(2)....ACCT(NAC) HR(NAC)

where UID : unit id
DATE : date for reported activity distribution
NAC : number of account codes referenced
ACCT(i) : ith account code
HR(i) : number of time units devoted to account i

It is assumed that the account code typology is sufficient to handle the suppressible and busy/non-busy classification necessary for PCAM.

3.5 SUMMARY

Figures 3.05 to 3.08 summarize the interrelationships among data elements in the data base.

Polygon and arc files are internally linked by feature id (figure 3.05). Similarly for cfs-related and patrol unit-related files (figure 3.06).

The topological relationships provide inter-feature pointers among files in the gbds (figure 3.07). Polygon records point to arc records that comprise the boundaries of individual polygons. Arc records, in turn, point to the polygons to the left and right of each arc. Arc records also point to the beginning and end nodes that terminate each arc. A second arc file points to the strands that comprise the arc.

A strand file includes pointers to the vertex file for beginning and end vertex id's.

The cfs-related files are linked to the patrol unit-related files via pointers in the cfs.resp file (figure 3.08).

Finally, the interface between the police-related data and the gbds is provided by the cfs.addr file and arc.addr file (figure 3.09).

OVERLAY FILES

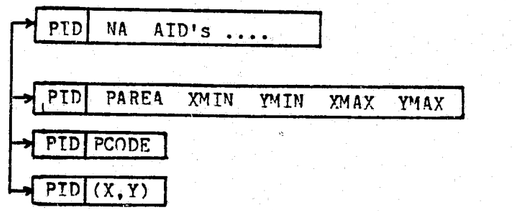
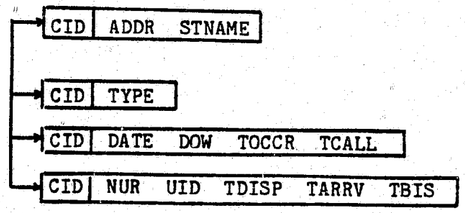


IMAGE FILES

Figure 3.05

GBDS: INTRA-FEATURE POINTERS

CFS-RELATED FILES



PATROL UNIT-RELATED FILES

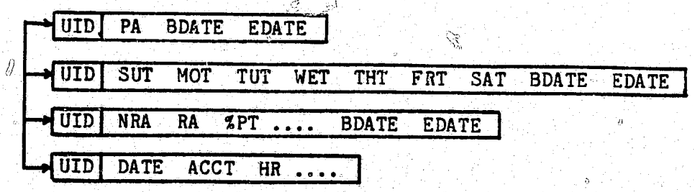


Figure 3.06

POLICE-RELATED DATA: INTRA-ELEMENT POINTERS

OVERLAY FILES

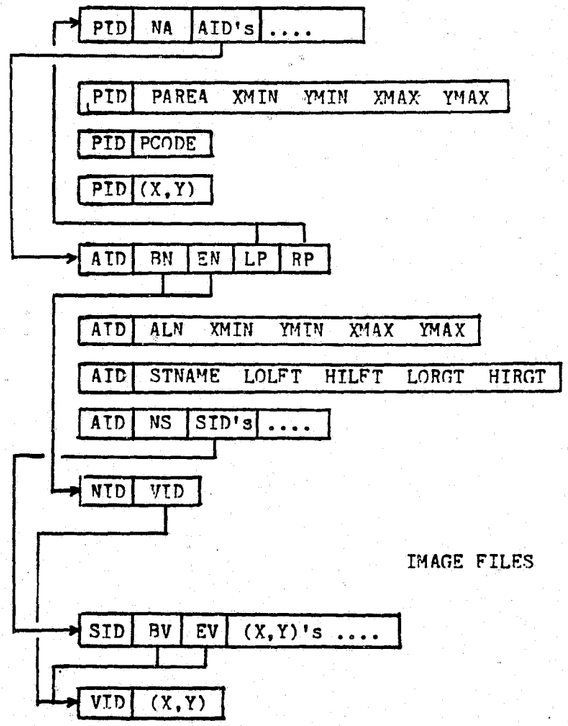
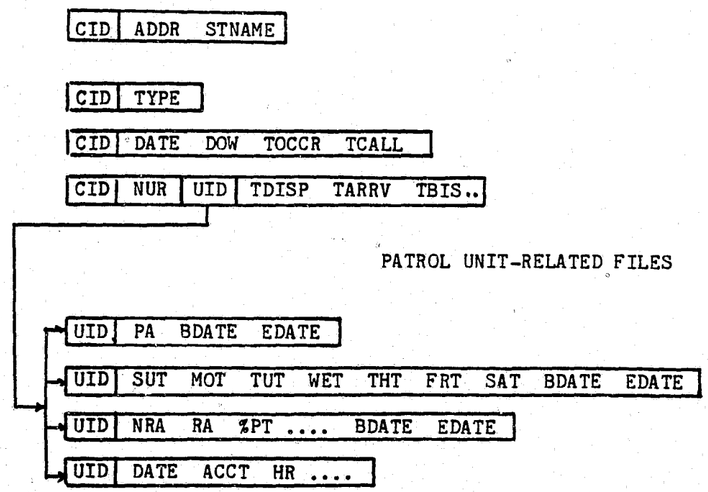


IMAGE FILES

Figure 3.07

GBDS: INTER-FEATURE POINTERS

CFS-RELATED FILES



PATROL UNIT-RELATED FILES

Figure 3.08

POLICE-RELATED DATA: INTER-ELEMENT POINTERS

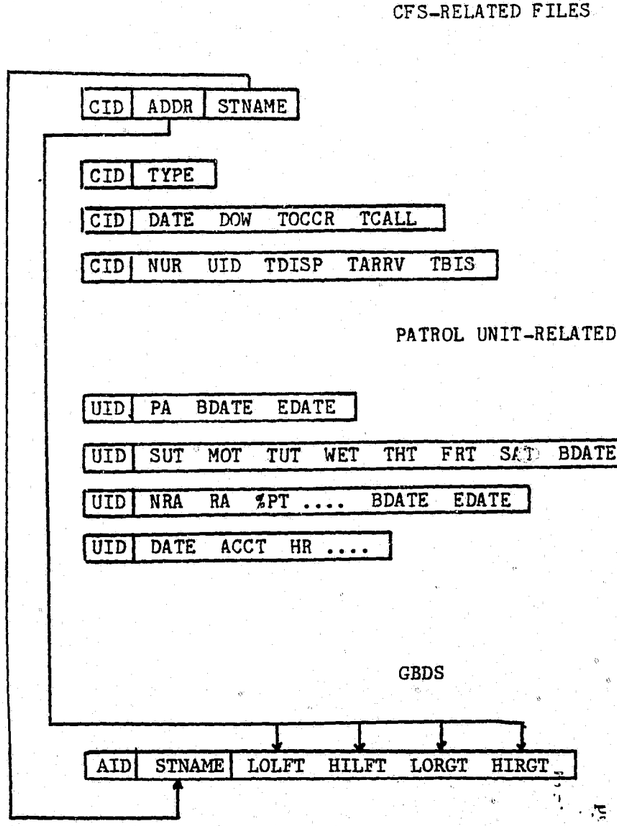


Figure 3.09

4 ALGORITHMS FOR FUNDAMENTAL DATA MANIPULATION

In this section, a set of procedures are outlined which support the more fundamental types of manipulation for generating subschema required by resource allocation models. Because of their utility in numerous situations, they are invoked repeatedly and are designed in a general form to handle different cases. This set of procedures in no way is intended to be exhaustive, only illustrative of required procedures for supporting the patrol unit allocation models, and their referencing of data elements in the gbds.

4.1 RETRIEVING IMAGE DATA

In situations where it is necessary to obtain image data for arcs, a procedure IMARC is defined. Given an arc id, the (x,y) coordinate string delineating the arc is returned. The arc.strand record lists the strand id's comprising the arc. The respective records in the image.strand file are referenced to obtain the actual coordinates, and the implicit direction of the strand is checked to determine the order in which the coordinates are to be returned.

Procedure IMARC:

Passed: id of (object) arc for which image data is required
Returned: coordinate string and number of points in string

1 Initialize point counter and point array.

2 In arc.strand file, find record for object arc to obtain strand id list.

3 The arc.strand record is comprised of strand id's. For each strand id in the list:

3.1 In image.strand file, find record for current strand.

3.2 Check the sign of the strand id as it appears in the arc.strand record (of step 3).

3.2.1 If the sign is '+', then add the coordinate string, as it appears, to point array.

3.2.2 If the sign is '-', then reverse the order of the coordinate string before adding it to point array.

3.3 Increment point counters by count specified in image.strand record.

4 Go on to next stand id in list, and return to step 3.1 .

5 Return point counter and point array.

A procedure IMPOL is defined for retrieving image data delineating the boundary of a polygon. Given a polygon id, the coordinate string that locates the polygon's boundary is returned. The pol.arc file indicates which arcs comprise the polygon boundary. With the arc id's, the procedures is similar to that for arcs, with an additional check for relative directions. Within the arc.topo file, the arc is assigned a direction which may or may not be the same. Moreover, the strands that comprise the arc have their own direction. All these assigned directions need to be accounted for to ensure a continuous coordinate string.

Procedure IMPOL:

Passed: id of (object) polygon for which image data is required
Returned: coordinate string and point counter

1 Initialize point counter and point array.

2 In poly.arc file, find record for object polygon.

3 The poly.arc record contains an arc list. For each arc id in the list:

3.1 Invoke IMARC to obtain coordinate string and point counter for current arc.

3.2 Check sign of current arc id.

3.2.1 If sign is '+', add coordinate string point arrayas it was returned in step 3.1 .

3.2.2 If sign is '-', reverse order of coordinate string before adding to point array.

3.3 Increment point counter by count (returned in step 3.1).

4 Go on to next arc id, and return to step 3.1 .

5 Return point counter and point array.

4.2 DISTANCE BETWEEN TWO POINTS

Given the coordinates for two points, the distance between them may be expressed in two ways. Euclidean distance is the lenght of a straight line between the two points. Right angle distance is the sum of the orthoganol line segments between the two points. Unless otherwise specified, distance computations in this report refer to the Euclidean method. The right-angle method is also given because of periodic references to it for approximating street distance (eg Hypercube).

Procedure EDIST:

Passed: coordinates for two points
Returned: straight-line distance

1 Given coordinates for two points ((X1,Y1),(X2,Y2)), distance is computed by the following:

D = ((X1 - X2)**2 + (Y1 - Y2)**2)**1/2

2 Return D .

Procedure RDIST:

Passed: coordinates for two points
Returned: right-angle distance

1 Given coordinates for two points ((X1,Y1),(X2,Y2)), distance is computed by the following:

D = /X1 - X2/ + /Y1 - Y2/

2 Return D .

4.3 ARC LENGTH

Computing the length of arcs may be required for initially generating an arc.meas file or for some ad hoc manipulation. Given an arc id, the length of the arc is returned. Computing arc length simply consists of accumulating the distances betwween points comprising the arc.

Procedure ALNGTH:

Passed: image data for (object) arc (or some portion)
Returned: length of (object) arc (or some portion)

1 Initalize LNGTH .

2 The passed image data consists of a coodinate string. For each sequential pair of points in string:

2.1 Invoke EDIST, returning D .

2.2 Increment LNGTH by D .

3 Go on to next pair of points, and return to step 2.1 .

4 Return LNGTH .

4.4 RECTANGLE SEARCH

Rectangle searching represents a 'quick and dirty' method for determining whether two features (points, arcs, or polygons) possibly intersect. A rectangle search is typically invoked prior to determining actual points of intersection. If the possibility of intersecting can be ruled-out, the costly procedure for determining actual points of intersection can be avoided.

Rectangle search utilizes the bounding rectangle coordinates stored in the approriate arc.meas file and pol.meas files (designated *.meas file). The underlying method is illustrated in figure 4.01 . Assuming quadrant 'E' represents the bounding rectangle for one feature, if the bounding rectangle of the second feature lies entirely in quadrant sets (A,B,C), (A,D,G), (C,F,I), or (G,H,I), then there is no possibility for intersection. Intersecting bounding rectangles does not necessarily imply intersecting features, only the possibility. A single, binary variable is returned by procedure RSRCH indicating the possibility of intersection.

Procedure RSRCH:

Passed: coordinates for bounding rectangles
Returned: binary variable indicating overlap possibility

1 Given coordinates for bounding rectangles of two features (XMIN,YMIN,XMAX,YMAX for i and j), determine whether the following boolean expression is true or false:

```
(XMIN(i).GT.XMAX(j)) or
(XMAX(i).LT.XMIN(j)) or
(YMIN(i).GT.YMAX(j)) or
(YMAX(i).LT.YMIN(j))
```

2 Set binary variable accordingly.

3 Return binary variable.

4.5 INTERSECTION BETWEEN TWO LINE SEGMENTS

Procedure INSECT is defined to determine the intersection between two line segments. Given the two pairs of coordinates for each line segment, the procedure returns (x,y) of the (virtual) point of intersection and a binary code indicating whether the intersection is virtual or actual. An intersection is not actual when the point of intersection is beyond the endpoints of one of the line segments.

Procedure INSECT first determines the intersection by converting the coordinate pairs to the general line equation form; then invokes a bounding rectangle search to determine if the point is a virtual or actual intersection.

Procedure INSECT:

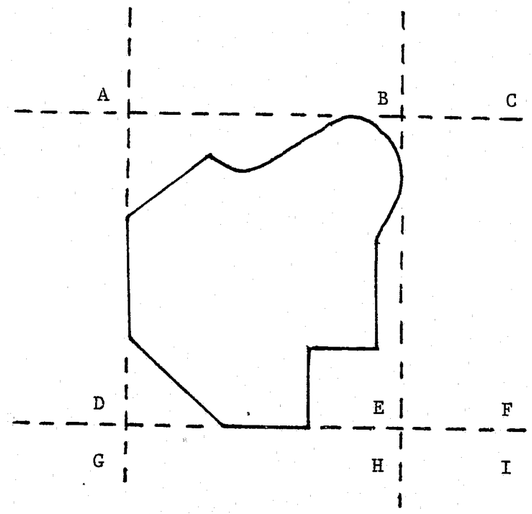


Figure 4.01

Passed: coordinates for two line segments
Returned: coordinates of intersection, and binary code inidicating actual or virtual

1 Given end points for two line segments ((X1,Y1),(X2,Y2) and (X3,Y3),(X4,Y4)), convert to general form by computing the following terms:

A = Y1-Y2 B = X2-X1 C = -A(X2)-B(Y2)

P = Y3-Y4 Q = Y4-Y3 R = -P(X4)-Q(Y4)

2 Determine virtual intersection by computing the following:

D = A*Q-B*P

X=(B*R-C*Q)/D

Y= (C*P-A*R)/D

3 Invoke RSRCH for (X,Y) (from step 2) and one of the original line segments, returning RCODE.

4 Return (X,Y) and RCODE.

4.6 AREA OF POLYGON

Given the coordinate string delineating the boundary for a polygon, a procedure PAREA is defined to return a value for the area.

Several methods exist for computing polygon area. The method employed in procedure PAREA is described in [1] and illustrated in figure 4.02. The area is computed for a rectangle which is the equal area representation of the trapezoid below a polygon boundary. This is done for consequtive pairs of points in the boundary. The rectangle's area is added to a subtotal when moving in a '+x' direction, and subtracted when moving in a '-x' direction.

Procedure PAREA:

Passed: Image data for (object) polygon
Returned: area of (object) polygon

1 Initialize AVAL .

2 For each sequential pair of points in the coordinate string for the polygon (X(i),Y(i)),(X(j),Y(j)):

2.1 Compute the following terms:

DX = X(i)-X(j)

DY = Y(i)+Y(j)

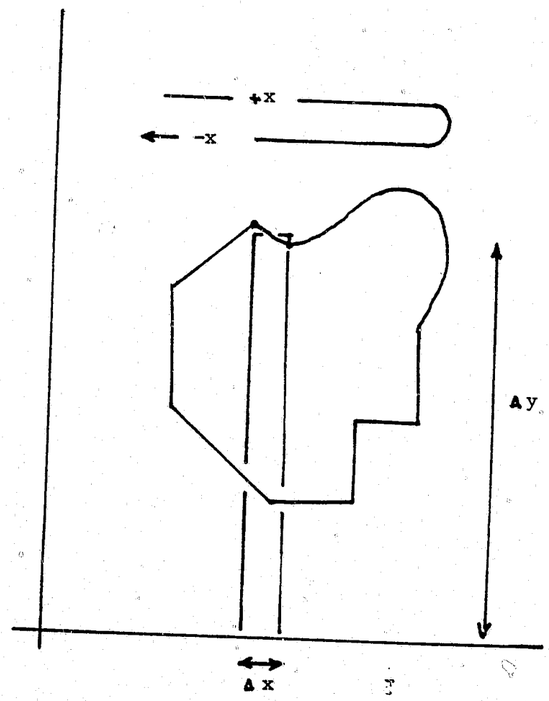


Figure 4.02

DZ = DX*DY

2.2 Increment AVAL by DZ.

3 Go on to next pair of points and return to step 2.1 .

4 AREA = AVAL / 2 .

5 Return AREA .

4.7 POINT IN POLYGON

Several methods exist for determining if a point is located within a specified polygon [1]. In the procedure presented below, the dimension of the object point is extended to infinity in a +x direction (figure 4.03). The underlying logic for the algorithm is this: If the extended point intersects the polygon boundary an odd number of times, the the point is located within the polygon; if even, then the point is outside. Cases where the extended line is tangent to the boundary are not considered intersections.

The first test in the procedure determines if the individual arcs of the polygon have the possibility of intersecting the extended point. If yes, the procedure accesses the respective image data for further analysis.

Procedure PIPOL:

Passed: coordinates for point, and id for polygon
Returned: logical variable which is true if the point is located within polygon, false if outside

1 Initialize tangent flag.

2 In *.pol.arc file, find record for object polygon to obtain arc id list.

3 For each arc id in list:

3.1 In *.arc.meas file, find record for current arc to obtain bounding rectangle coordinates.

3.2 Use a form of the rectangle search to determine if current arc has possibility of intersecting extended point:

3.2.1 If XMAX is greater than X for point, OR
if YMIN is greater than Y for point, OR
if YMAX is less than Y for point, go to step 4.

3.3 Invoke IMARC to obtain image data for current arc.

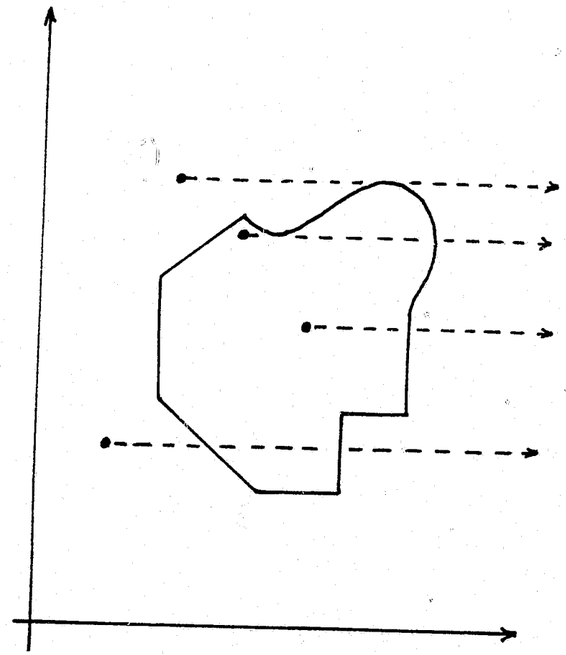


Figure 4.03

3.4 For each line segment (pair of sequential points):

3.4.1 Invoke INSECT, returning point of intersection between current arc and extended point, and RCODE1 indicating if intersection is 'actual'.

3.4.2 If intersection is actual, then:

3.4.2.1 If intersection is not an end point for line segment, then increment counter, and go to step 3.5 .

3.4.2.2 If intersection is end point for line segment, then check tangent flag:

3.4.2.2.1 If tangent flag is off, store end points, and go to step 3.5 .

3.4.2.2.2 If tangent flag is on, then:

3.4.2.2.2.1 If Y coordinates of end points of previous and current line segment are either all greater or all less than Y coordinate for object point, then increment intersection counter.

3.4.2.2.2.2 If not, do not increment counter.

3.4.2.2.2.3 Reset tangent flag.

3.5 Go on to next line segment and return to step 3.4.1 .

4 Go on to next arc and return to step 3.1 .

5 Inspect intersection counter. If 'even', set logical variable to 'true'; if 'odd', set to 'false'.

6 Return logical variable.

4.8 ARC CODING OF EVENTS BY ADDRESS

Because address is a common and available form of designating location for events that may serve as input for the data base, capabilities are required to code events in a form comprehendable to the gbds. This encoding capability is based upon the street overlay (see also DIME description in appendix C). The event's house number and street name are compared to records in the street.arc.addr file which specifies street names and address ranges for each side of the street segment. If the check is successful, the arc id is assigned to the event; a '+' or '-' is used to indicate left and right sides of the street segment.

Procedure ARCADDR:

Passed: address of an event
Returned: arc id for street.arc overlay with '+' or '-' indicating left or right side of arc

1 In street.arc.addr file, retrieve all records which have same street segment name as that specified for the event.

2 For each record retrieved in step 1, compare the event's address with address ranges for the segment:

2.1 If the event's address falls within the range specified for the left side of the arc, arc id equals that of the current arc record with a '+' sign; and go to step 4 .

2.2 If the event's address falls within the range specified for the right side of the arc, arc id equals that of the current arc record with a '-' sign; and go to step 4 .

3 Go on to next record of those considered in step 2 .

4 Return arc id with sign.

4.9 POLYGON CODING OF ARCS

Situations commonly arrise where arc coded data have to be compiled by polygons. The obvious case in police resource allocation is that of cfs's. CFS events by arc (street segment) are available from procedure ARCADDR. The resource allocation models, however, require cfs levels by geographic atom, precincts, and the like. Three procedures (POLARCS, POLARCC, and POLARCD) are defined to handle increasingly complex situations of converting street.arc data to polygon encoded form.

The simplest situation (figure 4.04) is where the areas represent polygons as they exist in the street overlay (ie city blocks). A department, for example, may define their reporting areas (small areas for statistical analysis) to be city blocks. A procedure POLARCS is defined to handle the case where a one-to-one correspondence exists between arcs that comprise polygon boundaries and street segments. Given a street.arc id, POLARCS returns the street.polygon id's associated with the left and right sides.

Procedure POLARCS:

Passed: (object) arc id for event with sign to indicate left/right
Returned: polygon id for event

1 In street.arc.topo file, find record for object arc to obtain polygon-left and polygon-right codes.

2 Using the sign of the (object) arc id, make the

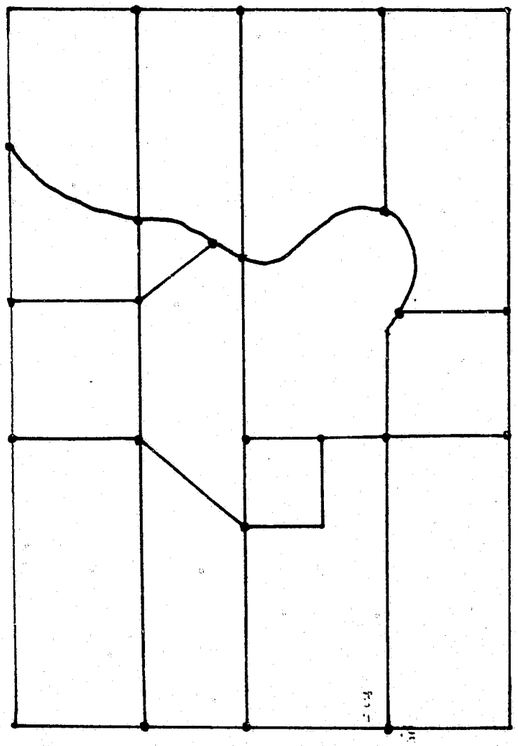


Figure 4.04

appropriate assignment:

2.1 If the sign is '+', then the polygon-left code is assigned to the object arc.

2.2 If the sign is '-', then the polygon-right code is assigned to the object arc.

3 Return assigned polygon id.

Consider now the situation (figure 4.05) where the polygons, which are defined for some overlay other than the street overlay, may comprise an area larger than a city block but its boundary is known to follow street segments. Procedure POLARCC handles this situation.

Given a street.arc id, POLARCC employs a bounding rectangle search to generate a list of polygons that may envelop or adjoin the arc. The street.arc can either lie on the boundary between two polygons or fall entirely within a polygon. The boundary case is checked first. By accessing respective arc.strand records for the street.arc and arcs comprising the polygon boundary, strand id strings are compared. If successful, left and right determinations are made and the respective polygon id returned. If unsuccessful, a point-in-polygon procedure is invoked for a representative point on the arc.

In the procedure description, the symbol '*' is used to refer to an overlay (and associated files) representing the areas for which events are to be coded.

Procedure POLARCC:

Passed: (object) arc id of event with sign to indicate left or right
Returned: polygon id for event

1 Initialize polygon candidate list.

2 In street.arc.meas file, find record for object arc to obtain coordinates for bounding rectangle.

3 For each polygon in * overlay:

3.1 Find record in *.poly.meas file to obtain bounding rectangle coordinates.

3.2 Invoke RSRCH for object arc (from step 2) and current polygon (from step 3); returning RCODE1.

3.3 If RCODE1 indicates possible overlap, add current polygon id to polygon candidate list A.

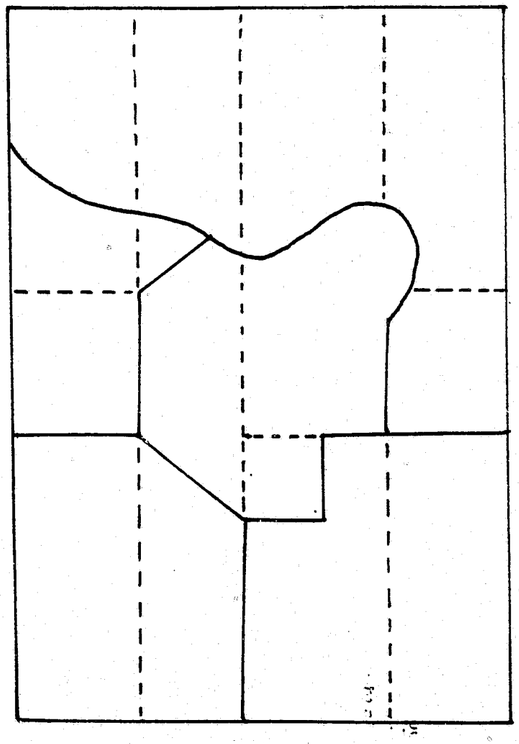


Figure 4.05

4 Go on to next polygon, and return to step 3.1 .

5 In street.arc.strand file, find record for object arc to obtain strand id list.

6 For each polygon id on polygon candidate list A (step 3.3):

6.1 In poly.arc file, find record for that of current polygon (from step 6), and obtain arc id list.

6.2 For each arc id in list (from step 6.1):

6.2.1 In arc.strand file, find record for current arc id (from step 6.2), and obtain strand id list.

6.2.2 Compare strand id lists associated with current polygon boundary (from step 6.2.1) with that of object arc (from step 5).

6.2.2.1 If the strand id lists match and are in the same order, then check sign of arc of event to make proper assignment:

6.2.2.1.1 If sign is '+', then set polygon id of event to polygon-left code of current arc (from step 6.2); and go to step 11.

6.2.2.1.2 If sign is '-', then set polygon id of event to polygon-right code of current arc (from step 6.2); and go to step 11.

6.2.2.2 If strand id lists match, but order is reverse, then use following assignments based on sign for object arc:

6.2.2.2.1 If sign is '+', then set polygon id of event to polygon-right code of current arc (from step 6.2); and got to step 11.

6.2.2.2.2 If sign is '-', then set polygon id of event to polygon-left code of current arc (from step 6.2); and go to step 11.

6.3 Go on to next arc id in list, and return to step 6.2.1 .

7 Go on to next polygon in candidate list, and return to step 6.1 .

8 Obtain a representative point for the arc of the event by following method:

8.1 From strand id list associated with object arc of the event (from step 4), select arbitrary strand id.

8.2 In strand.image file, find record for selected strand id (from step 8.1) to obtain coordinate string.

8.3 From coordinate string, select arbitrary intermediate point. (If string is comprised of 2 points, take average.)

9 For each polygon in candidate list A (from step 3.3):

9.1 Invoke PIPOL for polygon id and representative point for arc (from step 8.3); returning RCODE2.

9.2 If RCODE2 is 'TRUE', assign id of current polygon (from step 9) to object arc; and go to step 11.

10 Go on to next polygon in candidate list, and return to step 9.1 .

11 Return polygon id assigned to object arc.

The most complex situation (figure 4.06) is where the polygon's boundary may or may not coincide with street.arcs; and procedure POLARCD is defined to handle this case. Given a street.arc id, one or more polygon id's are returned coupled with the proportion of the arc relating to the respective polygons. All the proportion values sum to 1.0 .

POLARCD first determines if the object arc falls entirely within a single polygon. If not, the set of polygons to consider in subsequent analyses is reduced to those that are in the neighborhood of the object arc. The possibility of the object arc (or portions of it) falling on the boundary between polygons is considered next. The boundary case is handled by referencing id's for segments of the arc (strands), and where successful, eliminates the more costly processing of coordinate data. Where unsuccessful, representative points for strands are tested to determine in which polygon the segment is located. Given the definition of a strand, if one point is within a polygon, the entire strand is located within it. The procedure's structure is ordered such that, only when a simpler test proves unsuccessful are more complex tests applied.

In the procedure description, the symbol '*' is used to refer to an overlay (and associated files) representing the areas by which events are to be coded.

Also, references are made to 'an assignment array'. This is intended to be an N by 3 array where N is the number of strands that comprise the object arc. The three columns represent the strand id, the id of the polygon to which the strand is assigned, and the length of the strand.

Procedure POLYARCD:

Passed: (object) arc id of event with sign to indicate left/right
Returned: polygon count, polygon id string, and respective proportions for arc

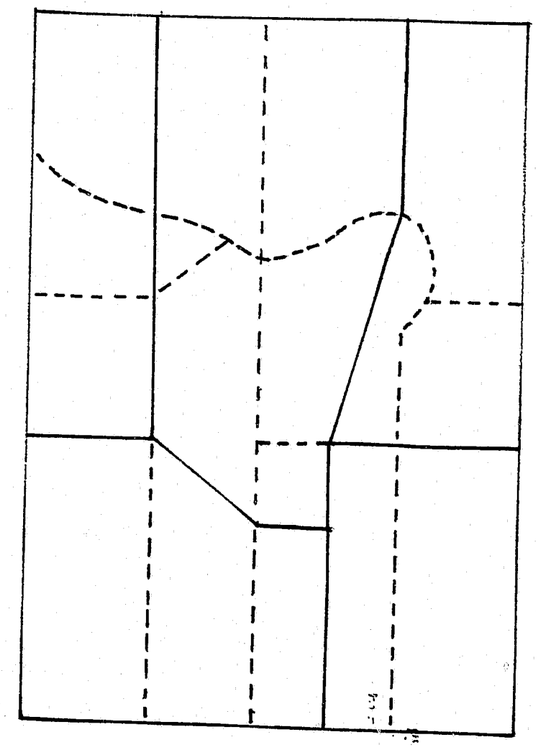


Figure 4.06

1 Initialize assignment arrays, candidate lists.

2 Determine which polygons are candidates for intersection by the following method:

2.1 In street.arc.measure file, find record for object arc to obtain coordinates for bounding rectangle.

2.2 For each polygon id in * overlay:

2.2.1 In *.polygon.meas file, find record for current polygon to obtain coordinates for bounding rectangle.

2.2.2 Invoke RSRCH for bounding rectangles of current polygon (from step 2.2) and object arc (from step 2.1); returning RCODE1.

2.2.3 If RCODE1 indicates possibility of intersection, add current polygon id to candidate array A.

2.3 Go on to next polygon, and return to step 2.2.1 .

2.4 If number of polygons in candidate array A is one, assign entry to object arc with a proportion of 1.0 and go to step 6 .

3 Given multiple polygons to consider, make polygon assignments for portions of object arc that may fall on polygon boundaries, using the following method:

3.1 In street.arc.strand file, find record for object arc to obtain strand id list.

3.2 For each polygon id in candidate list A:

3.2.1 In *.pol.arc file, find record for current polygon to obtain arc id list.

3.2.2 For each arc in arc id list:

3.2.2.1 In *.arc.meas file, find record for current arc to obtain coordinates for bounding rectangle.

3.2.2.2 Invoke RSRCH for bounding rectangles of current arc and object arc (from step 2.1); returning RCODE2.

3.2.2.3 If RCODE2 does not indicate the possibility of intersection, go to step 3.2.3 .

3.2.2.4 In *.arc.strand file, find record for current arc to obtain strand id list.

3.2.2.5 Compare entries in strand id list for current arc and object arc (from step 3.1):

3.2.2.5.1 If no common strand id's, go to step 3.2.3 .

3.3.3.5.2 If common strands exist, then find record in *.arc.topo file for current arc to obtain polygon-left and polygon-right entries.

3.2.2.6 Fo each common strand id in lists for current arc and object arc:

3.2.2.6.1 If signs for strand id's in both lists are same, then:

3.2.2.6.1.1 If sign for object arc is '+', assign strand polygon-left code for current arc (from step 3.2.2).

3.2.2.6.1.2 If sign for object arc is '-', assign strand polygon-right code for current arc (from step 3.2.2).

3.2.2.6.2 If signs for strand id's in both lists differ, then:

3.2.2.6.2.1 If sign for object arc is '+', assign strand polygon-right code for current arc (from step 3.2.2).

3.2.2.6.2.2 If sign for object arc is '-', assign strand polygon-left code for current arc (from step 3.2.2).

3.2.2.7 Go on to next pair of common strand id's, and return to step 3.2.2.6.1 .

3.2.2.8 Flag common strand id's in assignment array to eliminate them from further consideration.

3.2.3 Go on to next arc, and return to step 3.2.2.1 .

3.2.4 Flag arc id's considered in previous loop to prevent redundant consideration in subsequent loops.

3.3 Go on to next polygon in candidate list A, and return to step 3.2.1 .

4 Inspect assignment array to determine if all strands for object arc are assigned a polygon id. If not, remaining strands must fall within polygons (as opposed to on the boundary), and are assigned by the following method:

4.1 For each unassigned strand in assignment array:

4.1.1 Get representative point by the following:

4.1.1.1 In strand.image file, find record for current strand to obtain intermediate point. (If strand consists of 2 points, take average.)

4.1.2 For each polygon in candidate array A (from step 2.2.3):

4.1.2.1 Invoke RSRCH for bounding rectangles of current polygon (from step 2.2.1) and representative point for strand; returning RCODE3.

4.1.2.2 If RCODE3 indicates possibility of intersection, add current polygon id to candidate list B.

4.1.3 Go on to next polygon, and return to step 4.1.2.1 .

4.1.4 If candidate list B contains just one polygon id, assign that entry to strand, and go to step 4.2 .

4.1.5 Given multiple polygon candidates, make appropriate assignment by following method:

4.1.5.1 For each polygon on candidate list B:

4.1.5.1.1 Invoke PIPOL for current polygon and representative point for strand (from step 4.1.1.1); returning RCODE4.

4.1.5.1.2 If RCODE4 indicates that point is in polygon, assign strand id of current polygon and go to step 4.2 .

4.1.5.2 Go on to next polygon, and return to step 4.1.5.1.1 .

4.2 Go on to next unassigned strand in assignment array, and return to step 4.1.1 .

5 At this point, all strands of the object arc should be assigned a polygon id. Proportionate distributions are determined by the following method:

5.1 Invoke ALNGTH to get lengths for individual strands in assignment array.

5.2 Sum strand lengths for entries in assignment arrays to get total length of object arc.

5.3 For each polygon in assignment array, sum entries for strand lengths and divide by object arc length to get proportion.

6 Return number of polygon assignments with respective polygon id's and proportions from assignment array.

5 SUBSCHEMA FOR RESOURCE ALLOCATION MODELS

A number of models have been developed to assist the police planner in patrol unit resource allocation. In this section, two such models are considered which represent the dual nature of the problem. The problem of allocating patrol units to subareas of the city is addressed by the Patrol Car Allocation Program (PCAM). The Hypercube Queuing Model addresses the problem of delineating patrol area boundaries within the subareas. Described below is the methodology for referencing the data base to generate subschema that support the Hypercube Model, PCAM, and related applications programming.

5.1 CFS RELATED PROCESSING

Implicit in the compilation of input data for Hypercube, PCAM, or most any type of analysis, is some selectivity process as to which cfs's to include in the analysis. It is unlikely that when the input data requirements call for cfs level by some area, all cfs's that are located in the area are to be included. Explicit in the input file structure for PCAM is a temporal stratification of the data. Data is not only compiled by precinct, but also by day and hour of the day for each precinct.

In running the Hypercube model, temporal stratification is implicit. The data has to be representative of some time span, whether it is 1 month, 6 months, or whatever.

Moreover, there are some types of cfs's that are not related to patrol unit activities, and subsequently excluded from the analysis.

Given a time frame for the analysis, the data elements in the cfs.temp file are inspected to flag those cfs's which fall within the temporal domain. For instance, the analysis is to focus on the first 6 months of the year, the value for DATE is checked to see if it is between 1/1/81 and 6/30/61. If the analysis is focused futher upon say a single work shift, the TCALL (or TOCCR, whichever is more appropriate) is subjected to a range check representing the beginning and ending times for the shift.

Similarly, with the type of cfs. It may be desireable to exclude some types of cfs's which do not really impact patrol activities. Given the set of codes to be included (or excluded), the cfs.type file may be inspected and CID's flagged that satisfy the condition.

In describing the generation of subschemas for resource allocation models, it is assumed that the cfs data was preprocessed to select those cfs's relevant to the analysis, and that a list of the respective CID's are available in a cfs.input file.

5.2 HYPERCUBE QUEUING MODEL

The input data required by the Hypercube queuing model consists of the following data elements:

number of response units
number of geographic atoms
number of runs of the model to be made
exact/approximate model indicator
title for the run
relative cfs levels for each atom
relative patrol time levels by unit and atom
average response speed
coordinates for internal point of each atom
dispatch procedure indicator
que capacity indicator
average service time
average number of cfs's per hour

Several of these data elements are simple control parameters or identification type information that can easily be supplied at run time and need not be discussed here. These include the title and indicators for model type, dispatch procedure, and que capacity. Other elements represent simple counts that are readily available-- number of response units, geographic atoms, and cfs levels to input, and these are assumed to be available.

First, consider those elements which are not linked to the gbds-- average response speed, average service time, and average number of cfs's per hour. Speed is a function of time and distance. Average response distance for a region can be estimated by the square-root law, which has been shown to be fairly accurate in test cases. Response times for individual calls can be computed from the cfs.unit call. The difference between the time a unit was dispatched and the time it arrived on scene are simply averaged for each unit and for each cfs.

Average service time is also derived from the cfs.unit file. Averaged here is the difference between the time a unit was dispatched (TDISP) and the time it reported back in service (TBIS).

The average number of cfs's per hour is a single entry for the entire region. This average is easily computed by dividing the number of cfs's in the cfs.input file by the number of hours in the time frame represented by the cfs.input file.

The input data elements for the Hypercube Model that are linked to the gbds include geographical atom level data for relative cfs levels, for relative patrol time levels, and coordinates for internal points.

In the gbds, it is assumed that image data and overlay files are generated for police administrative areas. Geographical atoms represent one such overlay, although in the above disucssion they were referred to as reporting areas. Whatever the term, they represent a small area configuration for compiling statistics, and for the Hypercube Model, they represent the level at which the model operates.

The coordinates for atoms is directly available from the poly.cord file as defined in the gbds. Internal point coordinates served as an indicator of the inside area (vs outside) of a closed boundary, and now serves in the Hypercube Model to determine inter-atom distances.

The user is given the alternative of supplying empirically measured data of the distances between atoms. The gbds is supportive in this mode also. The street overlay is essentially a graph theoretic representation of the city's street system, from which internodal distances computations may be derived. (See section 5.4.2 for further discussion.)

The compilation of cfs's by geographic atoms is a particular application of the ARCADDR and POLARCS, POLARCC, or POLARCD intrinsic procedures.

For any cfs in the cfs.input file, a corresponding record in the cfs.addr file can be found. Procedure ARCADDR finds the street segment (arc) where the cfs is located, and indicates whether it is located on the left or right side of the street segment.

A street.arc.attr file can be defined which, for each arc, keeps counts of the number of cfs's on the left and right sides of the street segment.

In cases where geographic atoms correspond to city blocks, procedure POLARCS will convert the file of arc coded counts to polygon tabulations. A polygon's cfs level is directly gotten by summing the appropriate left and right arc counts for arcs comprising the atom's boundary.

In applications where geographic atoms represent areas that may encompass many city blocks, one of two procedures is used-- POLARCC or POLARCD.

POLARCC is for cases where the boundaries of atoms are known to follow street segments. An atom's cfs level is gotten in part from counts for street segments that comprise the atom's boundary in a manner as described for POLARCS. In addition, the cfs counts for street segments located within the atom are added to the atom's total.

The third case is where the boundaries for atoms may follow street segments, but not necessarily. Procedure POLARCD handles this problem in the general form. POLARCD operates in a manner similar to POLARCS for cases where the street segment falls either entirely on an atom's boundary or entirely within the atom. POLARCD, however, is expanded to handle situations where an atom's boundary intersects a street segment at an interim point, perhaps follows the street segment, and continues on. It does this by dropping to the strand level.

POLARCD divides a street segment, in these more complicated cases, into three components-- that portion that lies entirely within a particular polygon, that portion that comprises the boundary of a particular polygon, and that portion that is located outside the polygon. Based upon total length of the street segment and its component parts, POLARCD generates respective component percentages of the count for the arc. The right and left weights associated with the component entirely within the atom are used to increment the atom's own weight. For the component

coinciding with the atom's boundary (if any), the weight for the left or right side is used to increment the atom's weight; which depends upon the directions of the strand, the street arc, and the arc of the atom's boundary.

The final data element required for the Hypercube Queuing Model is the proportion of preventative patrol time spent by units in each of the atoms. There are several ways in which this type of data may be compiled. The weights may be estimated from the judgement and experience of officers, or derived from a departmental preventative patrol policy. The gbds can also serve as an input.

There are those types of cfs activity that are impacted by preventative patrol activity (eg auto theft vs white collar crime). In PCAM, these are referred to as 'suppressible crimes'. The distinction of suppressible crimes is assumed for the departments cfs type classification system. The cfs.type file can be scanned for those CID's associated with type codes that are considered suppressible. Given this cfs.input file, associated records in the cfs.addr file can be referenced, and a street.arc.attr file can be generated which records the suppressible cfs levels for right and left sides. Procedures POLARCS, POLARCC, or POLARCD can be invoked to determine relative preventative patrol demand at the atom level.

5.3 PATROL CAR ALLOCATION PROGRAM

Input requirements for PCAM are stratified in both a spatial and temporal sense. Many data elements represent tabulation for each precinct by day and hours within each day. Described below is a method in which the gbds design supports the generation of data elements required by PCAM.

Eliminating those elements which are trivial to supply, it is the following set which represents some level of processing:

For each precinct:

- Precinct area
- Lenght of streets in precinct
- Unavailability parameters

Day-specific data for each precinct:

- CFS parameter
- Service time parameter

Hour-specific data for each day and precinct:

- CFS factors
- Service time factors

Shift-specific data for each day and precinct:

- Average number of units on duty
- Average response speed
- Average patrol speed
- Percent cfs's priority 1
- Percent cfs's priority 2

Time block-specific data for each day and precinct:

- Average number of suppressible cfs's

Precinct Area:

The precinct configuration represents one of the overlays defined for the gbds. Consequently precinct area can be directly gotten from the respective prec.pol.meas file, or derived by the method that generated the file orginally. That is, invoking IMPOL to retrieve the image data for the polygon boundary, and PAREA to compute the area of the polygon. A scale factor converts the area value from internal file units to square miles.

Total Length of Streets in Precinct:

The total length of streets in each precinct is computed by comparing arcs in the street overlay with polygons in the precinct overlay. The lenght of individual street segments (or some portion thereof) is added to a subtotal for the precinct in which it is located. POLARCD is invoked to assign a street segment (or portion) to individual precincts, and the results are easily tabulated.

The output of the POLARCD procedure may be used to generate a cross reference file that is instrumental in the subsequent generation of cfs-related data elements for PCAM. The file, termed here ref.street.prec file, is given the following format:

```
AID NPR  PID(1) %LN(1) PID(2) %LN(2) .... PID(NPR) %LN(NPR)

where  AID : arc id in street overlay
       NPR : number of precincts in which street segment is
             located
    PID(i) : ith polygon id in precinct overlay
             (+,- indicate left/right sides of street segment)
    %LN(i) : proportion of street segment length in precinct
```

If precinct boundaries are known to coincide with street segments, POLARC may be invoked, and the ref.street.prec file is reduced to:

```
AID  +PID -PID
```

Cfs-related Data:

CFS related data are those elements that require processing of the cfs

files, and are identified here to be: the cfs and service time parameters and factors; the average response speed; the percentages for priority 1 and 2 cfs's; and the average number of suppressible cfs's.

The generation of these elements for the required areal units and time units represents a sizeable amount of processing. However, by referencing the ref.street.prec file defined above, and employing an effective selectivity search on the cfs files, the size of the task is reduced considerably.

A temporary working file, cfs.pcam, is defined to store codes and values relevant to the analysis. Its format is given as follows:

CID AID DAY TIME PRIOR1 PRIOR2 SUPP RTIME STIME

where CID : id for cfs
AID : arc id indicating street segment
DAY : id of day for input to PCAM
TIME : hour of day
PRIOR1 : priority 1 indicator (0/1)
PRIOR2 : priority 2 indicator (0/1)
SUPP : suppressible cfs indicator (0/1)
RTIME : travel time
STIME : on-scene time

For each cfs, the respective records in the cfs files are inspected to derive appropriate values.

AID is gotten directly from the cfs.street file.

The respective cfs.time record is inspected for DATE and TCALL and used to determine DAY and TIME.

PRIOR1, PRIOR2, and SUPP codes are derived from the cfs.type file which specifies the type of offense represented by the call. The ref.type file is referenced to determine the type code's priority, and suppressibility.

In the respective record of the cfs.unit file, the times that a unit(s) was dispatched (TDISP), arrived on scene (TARRV), and reported back in service (TBIS) are stored. Travel time and on-scene time represent the differences between (TDISP and TARRV) and (TARRV and TBIS) respectively. These can be averaged for all responding units to get a single representative value for the cfs.

Now consider, for each precinct, a three-dimensional array (fig 5.01), the cells of which represent the selective attributes of cfs's. One axis indexes the individual days included in the analysis. A second axis indexes the hours of the day. A third axis is defined to consist of the following elements:

(1) cfs level
(2) total travel time
(3) total on-scene time

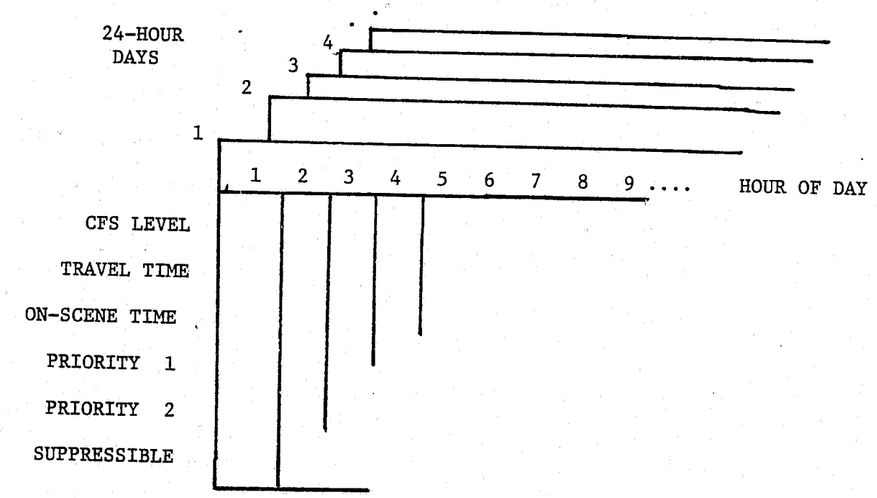


Figure 5.01

(4) total priority 1 level
(5) total priority 2 level
(6) suppressible cfs level

Given a record form the cfs.pcam file and the street.prec file, the appropriate cells of the precinct's array can be incremented. From the cfs.pcam record, AID points to a record in the street.prec file that indicates which precinct(s) the arc is located. Assume, for simplicity, that the street segment is located entirely within a precint.

The DAY and TIME codes (say 'i' and 'j') provide indicies for the X and Y axies in the array. Cell (i,j,1) is incremented by 1 to indicate that an additional cfs occurred in that precinct on that day for that hour. Cells (i,j,2) and (i,j,3) are incremented by the values of RTIME and STIME (travel time and on-scene time) respectively.

PRIOR1, PRIOR2, and SUPP are simply binary variables (0/1) indicating whether or not the cfs if of priority 1, of priority 2, and suppressible. These are added to cells (i,j,4), (i,j,5), and (i,j,6) respectively; a zero value will not affect the increment.

After processing all records in the cfs.pcam file, the cfs-related data elements required by PCAM are directly available or deriveable from the precinct arrays.

Precinct-specific cfs levels for hour 'j' of day 'i' are found in cells (i,j,1). Summing over 'i' gives a cfs level for day 'i'.

Precinct-specific on-scene time and travel time totals by day and hour are stored in cells (i,j,2) and (i,j,3) respectively. Dividing these by corresponding entries in cells(i,j,1) gives average on-scene and travel times by day and hour. Just as in the processing of the requirements for Hypercube, average travel speed is derived from an estimate of average travel distance (square root law) and average travel times. For, PCAM, however, average travel speed is computed for individual precinct by day and shift.

Proportion of cfs's that are priority 1 by day and hour is obtained by dividing cells (i,j,4) by corresponding (i,j,1) cells. The priority 2 proportion is derived in the same manner.

Average number of suppressible cfs's by day and hour is gotten directly from cells (i,j,6).

For those elements that need to be expressed in terms of shift or time block, the values in the cells (i,j,k) are summed for j = q to r where q and r are the indicies for the beginning and ending hours of the time unit.

UNIT RELATED DATA:

The unit related data required by PCAM include the following elements:

Unavailability parameters by precinct
Average number of units on duty by precinct, day, and shift
Average patrol speed by precinct, day, and shift

The unavailability parameters represent a precinct-specific linear regression analysis for observations of the percentage time spent by units on cfs and non-cfs related activities. The results are used in determining effective versus actual number of units in PCAM.

Records in the unit.accnt file indicate the time distribution of units for different activities. The ref.accnt file can be referenced to determine which accounts in the record are considered cfs-related and non-cfs related. These are merely summed and expressed as percentages for each unit.

The patrol area to which a unit is assigned is available in the unit.pa file. This, together with the ref.prec.pa file, indicate the precinct assignment for the unit. Given a unit's precinct assignment, and percentage for cfs and non-cfs time, the regression analysis can be employed to detemine unavailability parameters.

Average number of units on duty by precinct by day by shift is derived by first searching on DATE in the unit.accnt file for dates associated with days in the PCAM input. The found records are then sorted by precinct by referencing the unit.pa and pa.prec files and merely counting to get unit levels.

5.4 EXTENSIONS TO APPLICATIONS PROGRAMMING

The primary purpose of the gbds design exercise was to develop a structured geographic data base that supports the input requirements of police resource allocation models (represented by PCAM and Hypercube). The potential applications of the gbds are much greater, and some of the logical extensions to applications programming are discussed below. These include automated graphic display, network analysis, and statistical analysis of spatial data.

5.4.1 AUTOMATED GRAPHIC DISPLAY

Automated graphic display refers to the capability of the computer to generate charts or maps from encoded data. Maps are an effective medium for conveying information, and have been a tool for the criminologist and police planner long before computer processing. Perhaps suprising is that more has not been done to integrate graphics into resource allocation models; particularly given the spatial nature of the task. And, in fact, the reams of tabular output from multiple model runs have been pointed to as elements in the lack of success for some implementations [11].

There is nothing magic about the way in which graphics software works. In a vector mode, it is much like the way in which a person draws. Simply stated, a plotting device linked to the computer is instructed to begin at some intial location. It may then be instructed to lower its pen-like device on the medium and move to a second point, thereby drawing a line segment. It can then be instructed to move onto a third point, extending the line; or lift the pen device and move to another point to begin plotting.

Fundamental to driving a plotting device are the coordinates of the points defining the movement of the pen and indications of when to lift and lower the pen. In the gbds, coordinate data is available in the image data files. By selectively referencing the type of feature to be plotted (eg street system, patrol areas), the respective image data for individual arcs and polygons can be retrieved by procedures IMARC and IMPOL.

5.4.2 NETWORK ANALYSIS

A wide and varied set of algorithms, relevant to resource allocation, have been developed about network representations of space. Minimum path routines could be useful in planning primary response paths between areas. Optimal routing-type algorithms could assist the planner in developing preventative patrol plans. And discrete space location-allocation models could provide a basis for locating police facilities, delineating service areas, and the like. Network analysis operates on a graph theoretic representation of a delivery or transportation system, with appropriate weights and constraints placed upon individual features (nodes, arcs).

The street overlay, to a large extent, provides a graph theoretic configuration for the city's street system. Connectivity, in a cartographic sense, is specified in the street.arc.topo file. Modifications are required to this overlay to indicated connectivity in a transportation sense (eg overpass vs an intersection). Perhaps a street.node.attr file with a binary (0/1) indication of transportation type intersection. Given these modifications, respective street.arc.attr and street.node.attr files storing appropriate weights and constraints would support network analyses.

5.4.3 STATISTICAL ANALYSIS

Statistical analysis encompasses a wide range of quantitative techniques that are employed to reduce a large collection of observations to a smaller set of statistics that summarize basic trends or relationships. Some of the more commonly used statistical techniques are listed below with sample applications related to police resource allocation:

Univariate statistics: are the descriptive statistics for one-way frequency distributions, and include mean, mode, standard deviation, and the like.

Correlation analysis: summarizes the degree of association between two variables. For instance, correlation coefficients may be used to measure the association between robbery and burglary levels by patrol areas. That is, are areas of high robbery activity also areas of high burglary activity, and vice versa.

Regression analysis: allows for the projection of one variable as a function of one or more independent variables. Regression can be instrumental in prediction type models where say cfs levels by patrol area are to be estimated from a set of socio-economic variables for the area.

Factor analysis: is useful for reducing a large number of attributes for a large number of observation to a set of basic trends that manifest groupings of very positive or very negative correlation. For example, time of day-specific cfs levels for a number of crime types, by patrol areas can be subjected to factor analysis to determine regions of homogeneous diurnal crime patterns.

The subschema for statistical analyses are fairly uniform-- a set of observations or events, and values for attributes of the events. If the observation represents areal units or other spatial features, the analysis acquires a spatial nature.

The gbds can be instrumental in supporting spatial statistical analyses. The compilation of address-coded events (and their attributes) by arc and polygon features has been demonstrated for cfs's. If, rather than address, events are digitized, a point-in-polygon procedure can be invoked to convert the point distribution to areal tabulations.

6 CONCLUDING REMARKS

Resource allocation is becoming an application area of increasing importance to police departments. To support this activity, departments will need to develop effective methods for storing, retrieving, and manipulating spatial data. Presented above is one approach to a geographic data base design. The design employs a topologically-structured, vector representation of various categories of data associated with patrol resource allocation.

Generating the data specified for the design, together with the editing and maintaining it once in place, represents a sizeable task. The street overlay alone for a large city would include something on the order of 45k polygon records and 30k records each for arc and node files. Given the elements specified for the different files, this one overlay, although relatively complex, could require 4.5Mbytes to 5.0Mbytes of storage. A conservative estimate for the image data would be 5.0Mbytes; realistic estimates, however, would depend upon the complexity of the line work to be encoded, desired resolution, and the degree of coincidence among linear features. Add to this, files for other overlays, control data, data directories and data dictionaries. The data volumes are substantial, even by current standards.

Moreover, data integrity has to be maintained to ensure the logical consistency of the geographic data base. Throughout the geographic data base, an extensive set of pointers are employed to define the interrelationships among spatial elements. The integration of additional overlays and/or the modifications to existing overlays requires extensive updating of the pointer system. The nature of the changes, because of the data base's logical underpinnings, are tractible, and must to a large degree be automated in order to be feasible.

Also described in this report was the manner in which the geographic data base design supports resource allocation models. Related applications were also discussed-- graphic display, network analysis, and statistical analysis. Additional activities within the department may be linked directly or indirectly to the geographic data base. By considering the wider applications of the system, the substantial investment in implementing the system can be shared by a larger number of benefactors. And often, this represents the difference between a system which is feasible and one that is not.

REFERENCES

(1) Baxter, Richard. S., 1976. Computer and Statistical Techniques for Planners, Methuen & Co., Ltd., London, England.

(2) Bureau of the Census. The DIME Geocoding System, Washington, D.C., Census Use Study Report No. 4.

(3) Chaiken, J., 1971. Allocation of Emergency Units: Response Areas, The New York City Rand Institute, P-4745.

(4) Chaiken, J., and P. Dormont, 1975. Patrol Car Allocation Model: Users Manual, The Rand Corp., Santa Monica, Calif., R-1786/2-HUD/DOJ.

(5) Chaiken, J., and P. Dormont, 1975. Patrol Car Allocation Model: Program Description, The New York City Rand Institute, R-1786/3-HUD/DOJ.

(6) Chaiken, J., and R.C. Larson, 1971. Methods for Allocating Urban Emergency Units: A Survey, The New York City Rand Institute, P-4719.

(7) Chelst, K., 1974. An Interactive Approach to Police Sector Design: Innovative Resource Planning in Urban Public Safety Systems, Cambridge, Mass.: MIT, WP-03-74.

(8) Chelst, K., 1975. Implementing the Hypercube Queuing Model in the New Haven Department of Police Services: A Case Study of Technology Transfer, The Rand Corp., Santa Monica, Calif., R-1566/HUD.

(9) Colton, K. 1973. "Computers and Police: Patterns of Success and Failure", Journal of the ACM.

(10) Colton, K., 1977. "Police and Computers- The Use, Implementation, and Impact of Information in U.S. Police Departments", Proceedings of the Workshop on Mapping and Related Application of Computers to Canadian Police Work, National Research Council, Ottawa, Canada.

(11) Colton, K., ed., 1978. Police Computer Technology, Urban Public Safety Systems, Vol. 3, D.C. Heath & Co., Lexington, Mass.

(12) Gass, S., 1968. "On the Division of Police Districts into Patrol Beats", Proceedings of the 23d ACM National Conference.

(13) Internation Association of Chiefs of Police, 1975. Geographic Base Files for Law Enforcement, Research Division - Operations Research Section, IACP.

(14) International Association of Chiefs of Police, 1976. Geographic Base Files: Administrative Overview, Gaithersburg, Md.

(15) Jarvis, J., and M. McKnew, 1975. Data Collection and Computer Analysis for Police Manpower Allocations, Innovative Resource Planning in Urban Public Safety Systems, MIT.

(16) Kolesar, P., and E. Blum, 1973. "Square Root Laws for Fire Engine Response Distances", Management Science, V.19, p.1368-1378.

(17) Larson, R.C., 1972. Urban Police Patrol Analysis, MIT Press, Cambridge, Mass.

(18) Larson, R., 1975. Hypercube Queuing Model: Users Manual, The Rand Corp., Santa Monica, Calif., R-1688/2-HUD.

(19) Larson, R., and A. Odoni, 1981. Urban Operations Research, Prentice Hall, Englewood Cliffs, N.J.

(20) Martin, J., 1975. Computer Data Base Organization, Prentice Hall, Englewood Cliffs, N.J.

(21) Martin, J., 1976. Principles of Data-Base Management, Prentice Hall, Englewood Cliffs, N.J.

(22) Mitchell, W.B., et al., 1977. "GIRAS: A Geographic Information Retrieval and Analysis System for Handling Land Use and Land Cover Data", Geological Survey Professional Paper 1059, U.S. Geological Survey.

(23) Peucker, T.K., and N.Chrisman, 1975. "Cartographic Data Structures", The American Cartographer, V.2, No.1.

(24) Ross, R.G., 1978. Data Base Systems: Design, Implementation, and Management, AMACOM, New York, N.Y.

(25) Stevens, J., and B. Strahan, 1975. "Geo-Processing Systems and their Impact on Crime Analysis and Manpower Allocation", Paper presented in St. Louis, Mo.

(26) Strahan, B., 1975. "Future Directions for Law Enforcement Systems Development", Proceedings, URISA.

(27) Strahan, B. "A Case for Delivering Police Services Using the GBF/DIME File", U.S. Bureau of the Census.

(28) Tamminen, M., 1980. Management of Spatially Referenced Data, Report-HTKK-TKO-B23, Helsinki University of Technology, Laboratory of Information Processing Science, Finland.

(29) Tomlinson, R.F. (ed.), 1972. Geographic Data Handling, Vol. 1 and 2, I.G.U. Commission on Geographical Data Sensing and Processing, Ottawa, Canada.

APPENDIX A: COMPUTER USAGE BY POLICE DEPARTMENTS

SURVEYS OF COMPUTER USAGE IN POLICE DEPARTMENTS

The following table was derived from information presented in [11] concerning the pattern of computer use by police departments.

Application Area	Percent of Total Computer Use			Average Ranking of Importance*	
	1966	1971	1974	1971	1974
Police Administration	29.9	21.2	18.5	1.2	3.8
Traffic	24.4	17.9	17.4	6.7	7.3
Crime Statistical Files	19.7	19.5	19.5	33.7	36.7
Police patrol and inquirey	12.6	19.9	17.4	27.3	31.7
Miscellaneous operations	4.0	4.4	5.7	1.5	6.5
Resource allocation	8.7	12.2	16.0	45.0	42.0
Computer-aided dispatch	0.0	3.8	0.9	20.0	26.0
Criminal investigation	0.7	1.1	4.7	10.3	8.7

*Ranking is based on the average number of times applications were selected by police departments as one of their most important applications.

APPENDIX B: SPATIO-TEMPORAL VARIATIONS IN OFFENSE LEVELS

SPATIAL AND TEMPORAL VARIATIONS IN OFFENSE LEVELS - A CASE STUDY

Presented in part below are the results of a study* addressing the spatial and temporal variations in offense levels for an urban area (Buffalo, New York). Data representing a period of one year were tabulated for eight serious offense types-- (a) aggravated assault and homicide, (b) rape, (c) arson, (d) residential burglary, (e) non-residential burglary, (f) armed robbery, (g) strong-armed robbery, and (h) grand larceny. Tabulation of the offense data by time periods of the 24-hour day (table 3 and figure 4) indicates a range from a low of 7% for the 6am-9am period to over 2-1/2 times that level (18%) for the 3pm-6pm time period.

Crosstabulations by offense type reveal the composition and individual temporal patterns in the offense activity. The burglary and robbery-related offenses are predominant; accounting for over 70% of the serious offense activity. Residential and non-residential burglary levels fluctuate about the 'work-time' portion of the day. Above average levels for residential burglary occur between noon and 7pm; with levels peaking between 3pm and 6pm. Below average levels occur between midnight and 6am. In contrast, non-residential burglary levels are above average between 5pm and 6am, and below average during normal working hours.

Levels for robbery and other victim-related offenses increase steadily during the day with above average levels from 3pm to a peak between 9pm and midnight. After 12am, levels drop to below average levels between 3am and 8am.

Serious offense data were further stratified by subarea of the city; in this case census tracts. The result is a 3-dimensional array, 8 offense types by 8 time periods by some 70 census tracts. The mere presentation, not to mention interpretation, of such an array would be difficult. Thus, the array was subjected to factor analysis to reveal the basic trends in the data. The result (table 5) indicates which types of offense activity for individual time periods have similar spatial patterns.

Whereas arson and rape activity tend to be unstructured in both a time and space-dependent sense, the other offense types tended to group about three basic patterns: A pattern based upon residential burglary, a pattern based around commercial-related offenses, and a pattern based upon victim-type offenses. Apparent from the pattern of factor loadings is a consistency to the temporal variation. That is, an area's activity level relative to other areas is consistent, although the absolute level varies throughout the 24-hour day. Moreover, the mapping of the factor scores (not presented here) yields three separate spatial patterns. The residential burglary factor focused upon an inner ring of residential areas adjacent to the CBD. The commerical factor focused primarily upon the CBD, adjacent industrial areas, and scattered business and educational sections throughout the city. The victim-related factor focused on lower income residential areas adjacent to the CBD.

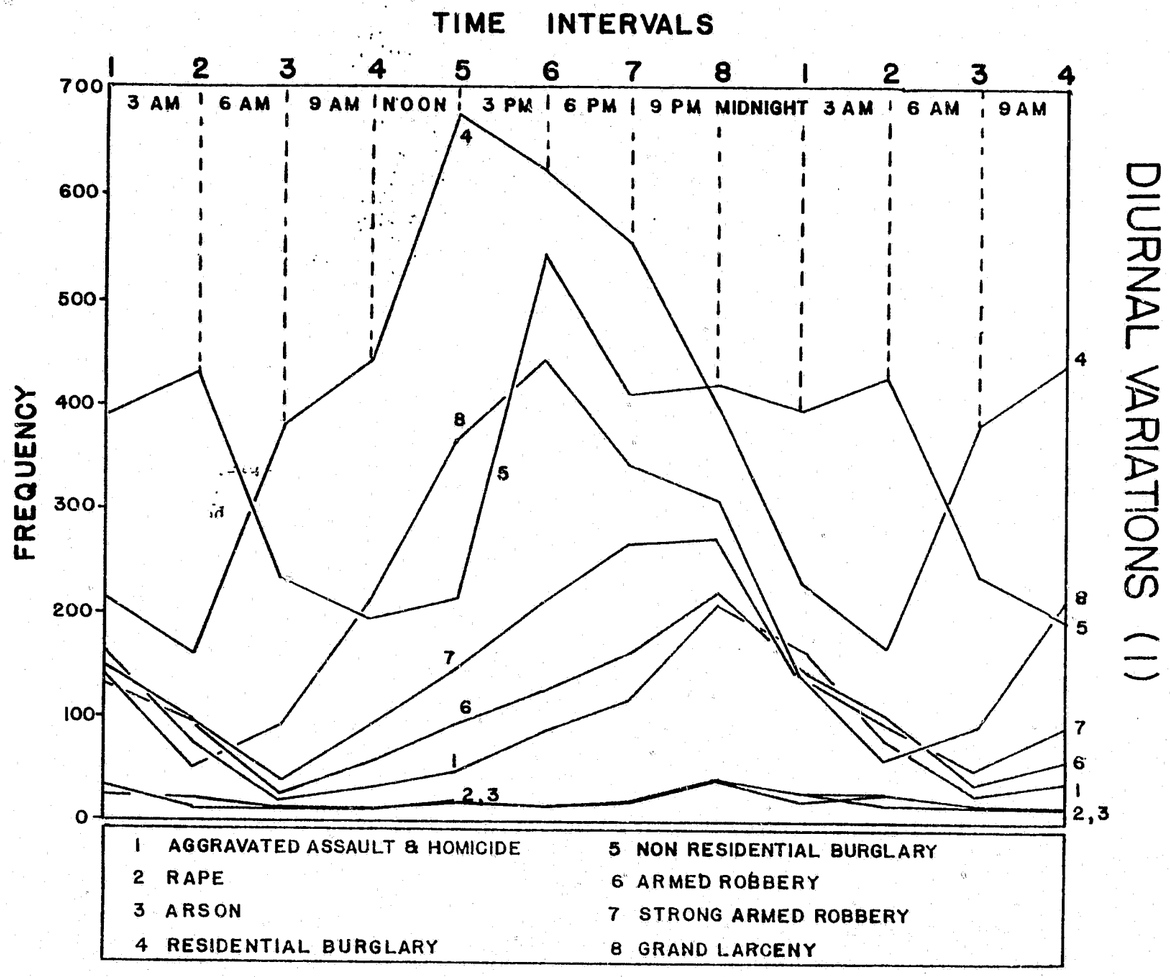
*Source: Claire, Robert W., 1974. "The Spatio-temporal Variations of Intra-Urban Crime", unpublished M.A. Thesis, S.U.N.Y. at Buffalo, New York.

Table 3

A Cross-tabulation of Offense Types

By Time Intervals

frequency ..... row percent ..... column percent	Time Intervals 1	2	3	4	5	6	7	8	
1 Aggravated Assault & Homicide	162 21.6 12.9	75 10.0 7.8	20 2.7 2.5	30 4.0 2.9	48 6.4 3.0	85 11.3 4.1	119 15.9 6.4	210 28.0 10.9	749 6.5
2 Rape	29 21.2 2.3	13 9.5 1.4	9 6.6 1.1	4 2.9 0.4	12 8.8 0.7	10 7.3 0.5	18 13.1 1.0	42 30.7 2.2	137 1.2
3 Arson	27 18.6 2.2	23 15.9 2.4	5 3.4 0.6	7 4.8 0.7	13 9.0 0.8	9 6.2 0.4	18 12.4 1.0	43 29.7 2.2	145 1.3
4 Residential Burglary	218 6.3 17.4	161 4.6 16.8	379 10.9 47.4	444 12.8 43.0	678 19.5 42.4	626 18.0 30.5	555 16.0 29.7	410 11.8 21.3	3471 30.2
5 Non-residential Burglary	394 14.0 31.4	428 15.2 44.8	231 8.2 28.9	190 6.7 18.4	216 7.7 13.5	540 19.1 26.3	409 14.5 21.9	415 14.7 21.5	2823 24.6
6 Armed Robbery	149 16.1 11.9	95 10.3 9.9	25 2.7 3.1	55 6.0 5.3	92 10.0 5.7	125 13.5 6.1	159 17.2 8.5	224 24.2 11.6	924 8.0
7 Strong-armed Robbery	132 10.5 10.5	95 7.6 9.9	37 2.9 4.6	90 7.2 8.7	147 11.7 9.2	217 17.2 10.6	265 21.1 14.2	275 21.9 14.3	1258 10.9
8 Grand Larceny	142 7.2 11.3	66 3.3 6.9	93 4.7 11.6	213 10.7 20.6	394 19.9 24.6	441 22.2 21.5	328 16.5 17.5	307 15.5 15.9	1984 17.3



DIURNAL VARIATIONS (1)

— FIGURE 4 —

Table 5

Factor Loading Matrix

	Offense Type	Time	1	2	3	4	5	6	7	8	h^2
1	Aggravated	1			-.55						.73
2	Assault	2							.34		.58
3	and	3	.50		-.62						.51
4	Homicide	4			-.75						.85
5		5			-.58						.84
6		6			-.72						.88
7		7			-.67						.88
8		8	.52		-.78						.90
9	Rape	1					.67				.64
10		2						-.57			.53
11		3							.60		.72
12		4	.50						.71		.62
13		5			-.77						.67
14		6				-.76					.66
15		7				-.37					.73
16		8	.52		-.59						.54
17	Arson	1	.56								.56
18		2								-.74	.67
19		3								-.70	.74
20		4	-.46								.63
21		5					-.63				.55
22		6								-.52	.62
23		7	.52								.51
24		8	.59								.63
25	Residential	1	.52		-.54						.77
26	Burglary	2	.81								.89
27		3	.71								.82
28		4	.72								.91
29		5	.75								.88
30		6	.79								.86
31		7	.68		-.60						.87
32		8			-.52						.76
33	Non	1					-.35				.66
34	Residential	2		-.50							.60
35	Burglary	3		-.60							.60
36		4		-.36							.74
37		5		-.66							.77
38		6			-.36						.70
39		7		-.42							.59
40		8		-.55							.80
41	Armed Robbery	1			-.59						.70
42		2					-.53				.73
43		3			-.36						.52
44		4			-.57						.79
45		5			-.56						.87
46		6			-.57						.84
47		7			-.70						.87
48		8			-.47						.79
49	Strong-Armed	1			-.64						.63
50	Robbery	2				-.75					.63
51		3			-.41						.81
52		4			-.65						.82
53		5				-.50					.86
54		6			-.47						.78
55		7			-.53						.78
56		8			-.62						.82
57	Grand Larceny	1		-.34							.55
58		2		-.55							.69
59		3		-.66							.78
60		4		-.70							.86
61		5		-.79							.86
62		6		-.62							.69
63		7		-.35							.74
64		8		-.43							.50

APPENDIX C: OVERVIEW OF THE DIME SYSTEM

OVERVIEW OF THE DIME SYSTEM

The DIME System was initially developed by the Bureau of the Census to assist in the compilation of census data. In 1970, the census was implemented in part by a mail questionnaire. Some system was required to convert the address-coded returns to census areal units. The DIME System is based upon graph theoretical principles, and because of its strong logical underpinnings, has been adopted in various forms to other systems.

The DIME system views a map of the city as a network, where points of intersection, termination, or noticeable direction changes are defined as nodes. The linear features that run between nodes are defined as segments. A segment may represent a street, river, administrative boundary, a railroad, and the like. The bounded areas delineated by segments are defined as blocks.

The DIME file is comprised of segment records. Each segment record retains the following information:

- segment name describing the linear feature
- 'from' and 'to' node id's
- 'left' and 'right' block id's
- If the linear feature represents a street segment, address ranges for both sides of the segment are included.

Additional codes may include municipality id's, zip codes, ward or other election district id's, and codes for nonstreet features.

A sample listing of a DIME file is given in figure 22.

A powerful aspect of the DIME System is its ability to support automated topological edits. Because of the extensiveness of data required to encode a map of the city in a DIME form, the opportunities for errors are many. The graph theoretical construct of the DIME System allows for chaining operations to logically check the internal consistency of the encoded data file and flag errors. Described below is the block chaining edit procedure as described in the Technical Description of the DIME System [2].

Block Chaining Edit

The block chaining edit operates on the three mandatory coded elements for each segment record: Segment name or description, node numbers, and block codes. It also serves as a check on the accuracy of the census tract code as the records are sorted by tract prior to the computer processing. As a byproduct of the topological edits, erroneous tract codes are detected. An elementary illustration of the method used in the block chaining edit is shown below. Block 105, the block to be edited, is shown in figure 23. The basic elements needed for the edit of block 105 are shown in figure 24.

The basic steps followed by the computer are:

1. All segments coded to block 105 (either block-left or block-right) for the census tract being edited are selected from the file.

2. As each segment record for block 105 is selected, the computer checks the position of the block number of the block being edited.

   a. If the block number is in the block-left position, it is transferred to the block-right position and the other block number is transferred to the block-left position. The node numbers are also exchanged; the 'from' node replaces the 'to' node and vice versa.
   b. If the block number is in the block-right position, no changes are made.

3. When all the block numbers for the block being edited are in the block-right position, the computer attempts to link or chain the nodes from one record to another, rearranging the sequence of segments as necessary. Notice that it was necessary to move the last segment record in figure 25 to a position between the first and second records. Figure 26 illustrates the final arrangement of the segments and the dotted lines indicate how the computer chains the segment records.

If the nodes chain and the first 'from' node is the same as the last 'to' node the block is considered topologically correct. Note the parallel of the computer operation in the hypothetical chaining of block 105 in figure 27.

If any segments remain, or if the block cannot be chained, the block records are rejected as a potential error. For instance, if any of the records in the above example were missing (i.e. not coded) the block would not chain and would therefore be rejected.

If the node numbers or block numbers were reversed, the block would not chain properly and would be rejected. As an example, if the left and right block numbers for 1st Street in figure 24, were coded 105 to the left rather than to the right and 102 to the right rather than the left, the block would contain a 'reversal' and would be rejected. Figure 28 illustrates this point.

Segment records for the blocks rejected are printed out on a reject listing for review. When reviewed, and corrected or recoded, the segment records are keypunched, inserted in the computer file, and reedited.

A similar operation may be implemented for nodes.

Address ranges may be checked by chaining sequentially the segments associated with a street name. Deviations (eg overlaps, gaps) in the expected ascending order of the address ranges can be flagged as possible errors. Also, odd/even assignments for left/right sides of the street can be checked.

Once the DIME file is considered topologically clean coordinates for the nodes can be inserted.

Segment name or description	Code	From node	To node	Block No.		Left Addresses		Right Addresses		Header No.
				Left	Right	Low	High	Low	High	
ANDERSON RD.		75	76	111	120	900	998	901	999	30151
ANDERSON RD.		76	77	112	119	1000	1098	1001	1099	30151
ANDERSON RD.		77	78	113	118	1100	1198	1101	1199	30151
ANDERSON RD.		78	79	114	117	1200	1248	1201	1249	30151
ARGONNE ST.		34	36	271	279	400	488	401	449	30151
ARGONNE ST.		36	35	270	283	450	498	451	499	30151
ARGONNE ST.		35	39	270	282	500	598	501	599	30151
BADGER RIVER	2	107	108	137	137					30151
BADGER RIVER	2	108	112	137	137					30151
BADGER RIVER	2	112	113	138	137					30151

Figure 22. Segment items (codes for each segment)

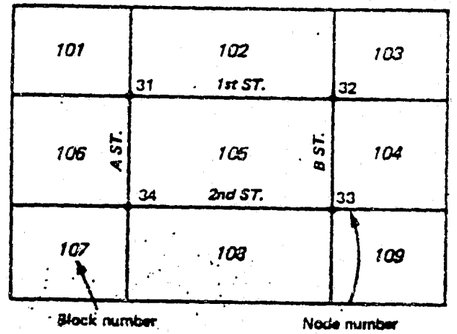


Figure 23.

Segment name	From node	To node	Block left	Block right
1st St.	31	32	*102*	*105*
2nd St.	34	33	*108*	*108*
A St.	34	31	*106*	*105*
B St.	33	32	*105*	*104*

Figure 24.

Segment name	From node	To node	Block left	Block right
1st St.	31	32	*102*	*105*
2nd St.	33	34	*108*	*105*
A St.	34	31	*106*	*105*
B St.	32	33	*104*	*105*

Figure 25.

Segment name	From node	To node	Block left	Block right
1st St.	31	32	*102*	*105*
B St.	32	33	*104*	*105*
2nd St.	33	34	*108*	*105*
A St.	34	31	*106*	*105*

Figure 26.

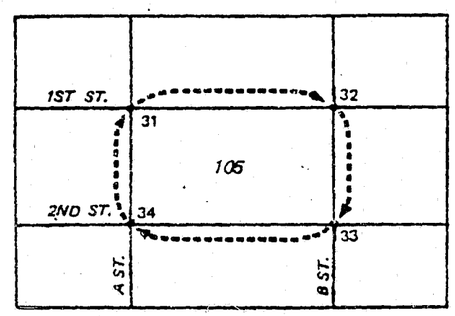


Figure 27.

Segment name	From node	To node	Block left	Block right
B St.	32	33	*104*	*105*
2nd St.	33	34	*108*	*105*
A St.	34	31	*106*	*105*
1st St.	32	31	*102*	*105*

Figure 28.

APPENDIX D: FILE STRUCTURE FOR GIRAS

GIRAS FILE STRUCTURE

The GIRAS file structure was designed to encode land use and land cover data. It is a polygon-based system which employs a topologically-structured, vector representation for delineating polygons. A map header provides descriptor information, parameters used in processing, counts, and control data for the entire map file. Because of the data volumes involved, a map may be subdivided into sections, each with 'digestible' amounts of data. Section headers, as with the map header, provide global data for the individual sections. Topological, measurement, and attribute data for spatial elements are retained in separate polygon and arc files (for each section). No node subfile exists, but node id's are referenced to ensure connecting arcs. Image data is stored by arc in a coordinate subfile.

As with GIRAS, many of the overlays in the geographic data base design for resource allocation represent a closed and exhaustive polygon set. To that extent, the resources associated with GIRAS, particularly in respect to data capture and quality control, may be applicable.

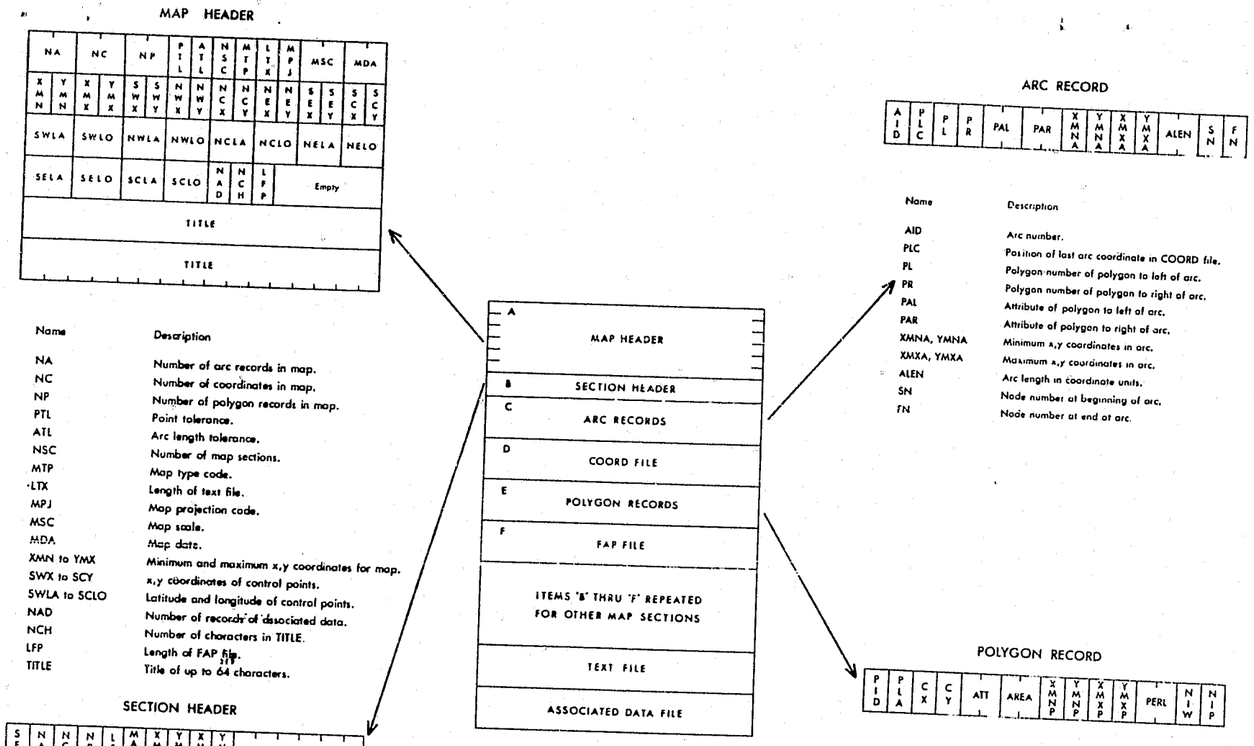


FIGURE 3 — GIRAS file structure

APPENDIX E: OVERVIEW OF PATROL CAR ALLOCATION PROGRAM

OVERVIEW OF PATROL CAR ALLOCATION MODEL (PCAM)

The software capabilities for patrol unit allocation are supported by PCAM. PCAM makes unit allocations on a precinct level, where a precinct is presumed to be an administrative area: (a) over which a commander has the authority as to how many units will be fielded at various times, and (b) which dispatchers treat as an independent entity by assigning its unit to its incidents in all but unusual situations.

PCAM has a number of features that distinguish it from other allocation programs:

- Calls for service can be classified according to three priority levels. If all units in a precinct are busy when a cfs is made, the cfs is placed in a queue and assigned by priority level to the next available unit.

- A patrol unit's time is divided into cfs time, non-cfs time (downtime), and preventative patrol time. By relating cfs time to non-cfs time, statistics are computed for 'effective' (vs actual) units.

- PCAM allows for one overlay tour that overlaps regular tours of duty.

- User has flexibility in specifying which precincts, days, and tours are to be input into the allocation.

- PCAM can operate in a descriptive or prescriptive mode.

PCAM computes the following performance measures:

- Average utilization of an actual unit.
- Average utilization of an effective unit.
- Average number of units available.
- Average travel time.
- Average total delay.
- Average patrol frequency.
- Patrol hours per suppressible crime.
- Average patrol frequency times suppressible crimes per hour.

In addition to computing the performance measures for the existing or some user-specified allocation, PCAM can generate an allocation to meet certain constraints or to optimize some objective function. In either case, PCAM begins with an initial allocation that may represent the

existing allocation, the output of a previous step, or the minimum allocation. (The minimum allocation refers to the number of units necessary in each time block for the utilization of an effective unit to be less than 1.)

The user can instruct PCAM to generate allocations to meet one, some, or all of the following constraints:

- Average utilization of effective unit.
- Average travel time.
- Average number of units available.
- Patrol hours per suppressible crime.
- Patrol frequency.
- Minimum number of units allocated.
- Percent calls delayed.
- Average delay of calls of priority 'i'.
- Total delay (queue + travel).

Each constraint is checked for each time block to see if it is satisfied; if not, the number of units is increased by 1 and checked again. All the performance measures display the property that an increase in the number of units will lead to an improvement in any measure. The user has the added flexibility of specifying certain tours, shifts, or precincts where constraints are to be met.

PCAM can generate allocations that minimize the following objective functions:

- Average percentage of calls delayed in queue.
- Average length of time a call stays in a queue.
- Average total response time (queue + travel).

The user specifies the total number of car-hous to be allocated. PCAM begins with the initial allocation, and computes the value for the objective function. PCAM then temporarily adds a unit in turn to each tour in each precinct and recomputes the values of the objective function. The unit is ultimately assigned to that shift that yields the greatest improvement to the objective function. This process is continued for the balance of the car-hours to be allocated. As with constraints, allocations can be made over certain specified tours and/or precincts.

The input data requirements for PCAM are discussed in section 5.3 .

APPENDIX F: OVERVIEW OF HYPERCUBE QUEUING MODEL

CONTINUED

1 OF 2

OVERVIEW FOR THE HYPERCUBE QUEUING MODEL

The capability for evaluating alternative beat configurations is supported by the Hypercube model. Hypercube does not generate an 'optimal' configuration; rather it computes performance measures for a specified configuration which allow for comparisons of different plans.

Hypercube requires that an area (eg precinct) be subdivided into geographical atoms (200 maximum), no more than a few blocks in size. The user aggregates the atoms into beats, which may or may not overlap. Although units are assigned a set of atoms for preventative patrol, the assumption is made than any unit can travel to any atom to service a call. Given this structure, separate Hypercube runs are made for individual precincts.

The model consists of an exact and approximate version. The total number of units is limited to 15 for the exact version, and something reasonable (eg 50) for the approximate version. Both versions require the same input data and produce the same output.

The Hypercube model works with the following assumptions:

- Calls for service are generated independently for each atom.
- Patrol units can travel to any atom.
- One unit responds to a call. If no units are available, the call is placed in a queue, and serviced on a first-come basis.
- Service time (travel time + on-scene time) has a known average value, and the standard deviation is approximately equal to the mean.
- Variation in service times due to travel are assumed to be minor in comparison to variations of on-scene time.

The following performance measures are computed by Hypercube:

- Region-wide mean travel time.
- Region-wide workload and workload imbalance.
- Region-wide percent dispatches that remove a unit from its beat.
- Workload of each response unit (percent of time unit is busy servicing calls).
- Mean travel time to each geographical atom.
- Mean travel time to each district.
- Mean travel time of each response unit.
- Percent of responses in each unit's district that are handled by other units.
- Percent of responses of each unit that dispatch the unit outside its district.
- Percent of responses within each atom that are handled by each unit.
- Frequency of perventative patrol passings in each of the atoms.

The input data requirements for Hypercube are discussed in section 5.2 .

END