TIGRIS:
TOPOLOGICALLY INTEGRATED GEOGRAPHIC INFORMATION SYSTEM

Dr. John R. Herring
INTERGRAPH Corporation
One Madison Industrial Park
Huntsville, Alabama 35807-2180

ABSTRACT

A Geographic Information System (GIS) requires both interactive edit/query, and powerful spatial analysis capabilities for large volumes of geographic data. This spatial analysis requirement has led to the investigation of the mathematical field of topology. The major problems of previous topological systems have been the construction of a logically consistent topological structure and the integration of structured spatial data with nonspatial attribute data. The TIGRIS system incorporates both types of data into a single, highly interactive data base using an object oriented system on Intergraph's stand alone workstations. This paper describes TIGRIS and its design.

INTRODUCTION

In this section, we describe the problems TIGRIS was designed to solve and outline the approaches taken.

The primary problem of any GIS is the manipulation and query of large quantities of spatial data. The accepted theoretical solution is to topologically structure the spatial data. TIGRIS has a spatial structure that is based on a generalization of two dimensional cellular topology (Switzer, 1975), implemented with sound mathematical principles, and provably correct algorithms.

Many systems built on topology require post processing to create the topological structure from coordinates input; separating data gathering, and data structuring functions. This is unsatisfactory for three reasons: it makes feedback from structuring impossibly slow; it limits or eliminates "what if" spatial queries; and it places the burden of defining spatial relations on the coordinates alone, which have inherent metric inaccuracies introduced by finite precision, finite representation of curves, and statistical limitations of data gathering technology. In TIGRIS, the extraction and structuring of spatial data occur in one process. Data is taken directly from the digitized input and placed into the topological structure, using algorithms optimized for topological extraction. TIGRIS also uses topology to automate and optimize analysis of the spatial data, relying heavily on the techniques of algebraic topology to limit computational loads. This makes TIGRIS a truly interactive spatial data base, that can update its topological structure "on the fly" while the user is digitizing or editing features.

A shortcoming of many GIS systems, that can cause performance problems, is the segregation of spatial and attribute data into separate management systems. This gives the system two different data interfaces, doubling

the complexity of I/O processing. TIGRIS combines spatial and attribute data into a single data base, supported by an integrated query language and software interface. This is made possible by basing TIGRIS on an object oriented data management system and programming environment (The Object Manager). Response times are maintained even with large data volumes through several approaches; optimized algorithms, spatial indexes (r-trees) to organize and cluster the data, and a powerful dedicated processor (InterPro/Act 32C, UNIX based on the Fairchild Clipper processor (5 million instructions per second), with multiple support processors). Further, TIGRIS supports complex, nonhomogeneous features (composed of arbitrary combinations of points, lines, and areas), multiple representations of features, and arbitrary feature to feature relations.

The result is TIGRIS, a highly interactive, powerful, spatial and geographic information system.

THE OBJECT MANAGER

The TIGRIS software environment is based on a design methodology called "object oriented programming". The Intergraph Object Manager (OM) combines data base management, I/O control, and process control into a complete object oriented design and implementation environment.

Object oriented programming begins with the "encapsulation" of data into structures based on usage. All data structures of one type are called a "class"; each occurrence of a class is an "object" and an "instance" of its class. Each class is associated to a set of procedures ("methods") for the creation, edit, query, and deletion of instances of the class. The methods define the interface to the instances of a class, ensuring a high degree of modularity through "data hiding" (only a class method can refer to the structure of a class). Each method is invoked on a particular object by a "message" send. Classes share "message protocols" through methods with identical names and parameters. Classes can be defined using other classes as part of their structure through "subclassing", and can "inherit" all methods and messages of the "parent" class. Subclassing and inheritance increase the reusability and modularity of the methods.

An object structure may include any number of named "channels" which hold pointers to other objects. Objects are related to one another through matched pointers in channels. Channel specifications determine whether the objects on a channel are ordered, and whether the relation is one to one, one to many, or many to many. Channels can be used to group objects for collective messages.

The abstract collection consisting of the class *specification*, *methods*, and messages is referred to as a "package". A package is a fundamental grouping in object oriented data abstraction.

OM takes care of creating and deleting objects and channel connections. OM further controls message sends, including any needed indexing and I/O. Thus, OM is a data base management system as well as a object oriented software environment.

More complete descriptions of the object oriented software philosophy, methodology, and benefits can be found in the references (Cox, 1986, and White, 1986).

THE TOPOLOGICAL STRUCTURE

The concepts that the TIGRIS system uses to organize and manipulate spatial data are derived from cellular and algebraic theories of topology. Topology is "coordinate free geometry"; the study of those geometric concepts that can be defined independent of any coordinate system. A concept is coordinate system independent if a continuous change in the coordinate system does not alter its definition. For example, a curve can be defined as a continuous image of an interval of real numbers. Changes in the coordinate system modify the defining function from the interval into the space, while the curve, as a ordered point set, remains the same. Since "curve" can be equivalently defined in all acceptable coordinate systems, "curve" is a topological concept. Other such topological concepts include connected, adjacent, dimension, bounded, boundary, inside, outside, and orientation.

Why is topology useful in spatial data systems? Without coordinate geometry, topologists are forced to frame geometric theory in symbolic terms, to translate each geometric problem into an equivalent symbolic (algebraic) problem, to solve the symbolic problem, and to retranslate the results back into the geometric world. In this translation, the topologists gained a powerful symbolic tool for manipulating facts about geometric configurations. It is upon this tool that TIGRIS is based.

In using topological concepts to describe geographic objects, we split the spatial objects and relations away from their coordinate descriptions. These objects can then be manipulated without reference to their coordinate descriptions. Since the manipulation of coordinate descriptions is the usual bottle neck in spatial analysis, the coordinate-free, topological algorithms are often significantly more efficient than the coordinate based algorithms that they replace.

Topological Objects

The building blocks of two dimensional topological theory consist of points (nodes, 0-cells), nonintersecting curves between these points (edges, 1-cells), and the connected two dimensional areas bounded by these curves (faces). TIGRIS faces are allowed to have "holes" or "islands", making them distinct from topological 2-cells.

At each point in the life of a TIGRIS data set, the allowable coordinate values are divided so that each point

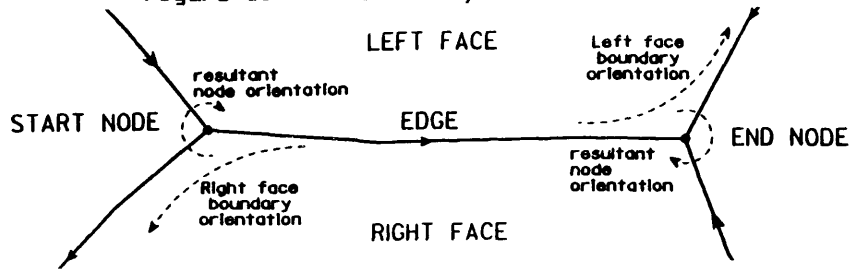
is on one and only one topological object. At the very beginning, after the set has been initialized, but before any digitization occurs, the data set consists of a single face, that covers the entire coordinate space (called face_1 in TIGRIS). During the addition of spatial data, one of several things may occur: an isolated node may be added to the interior of some face; a node may be added to the interior of some edge, splitting that edge into two edges; an edge may be placed between two nodes in such a manner as to not split a face; or an edge may be placed between two nodes splitting a face into two faces.

Topological Relationships

The topological structure of a map resembles a jigsaw puzzle. The edges and nodes are analogous to the cut lines of the puzzle, and the faces to the puzzle pieces. The difference is that in the topological structure, the "pieces" are aware of their relationships with other adjacent "pieces". This means that the topological puzzle can assemble itself, since each piece "knows" how it is related to its surroundings and, thus, how it is related to the entire puzzle.

The relations between the topological objects are based on the notion of a "boundary". The boundary of a node is empty; the boundary of an edge consists of the two end point nodes of that edge; and the boundary of a face consists of the all the nodes and edges "close to" the face (including any isolated nodes). The coboundary relation is the reflection of the boundary relation. The coboundary of a face is empty. The coboundary of an edge consists of the two faces on either side of that edge. The coboundary of a node (node star, node umbrella) consists of all of the faces and edges that surround that node. The remaining topological information relates to the order in which things appear in the boundary and coboundary of the topological objects. The general term for this order is "orientation." The orientation of the topology derives from the establishment of a direction for each edge corresponding to the order in which its coordinates are stored. The end points of the edge are called the "start node" or the "end node" depending on whether the node point is first or last in the edge coordinate list. The two faces either side of the edge are called the "left face" or the "right face" depending on the sign of the angular direction of the face as locally measured from the tangent to the edge (left is positive, corresponding to a positive angle as measured from the tangent). The orientation of the boundary of a face, derived from "left is positive", is counterclockwise for the exterior boundary components and clockwise for the interior components. Thus, the boundary of the face consists of some number of circular lists of signed edges alternating with nodes. The sign of the edges corresponds to the leftness or rightness of the face with respect to the edge. Each isolated node is included as a separate boundary component. The orientation of the coboundary of a node is induced from the orientation of the surrounding faces. At first it is a bit counter-intuitive, but this orientation is in fact clockwise (rightward or negative) and not counterclockwise (see Figure 1).

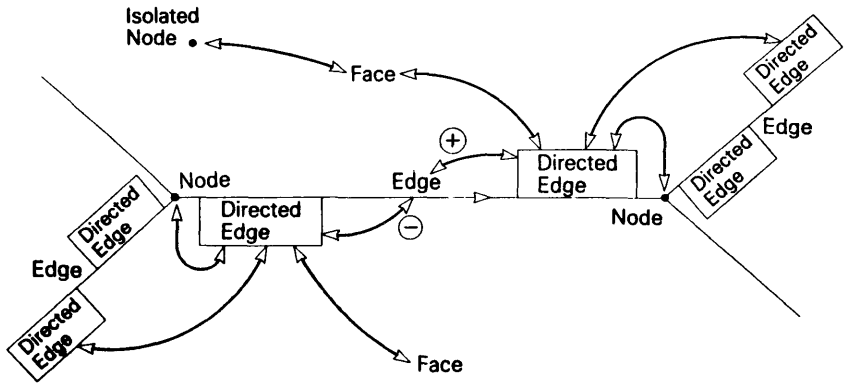
Figure 1: The Geometry of Orientation



When designing the algorithms, it becomes apparent that relationships to edges fall symmetrically into precisely two types: positive or negative orientation. To simplify code, and to eliminate orientation fields within channels, a new object type, called the "directed edge", was introduced. A directed edge is used to represent one of the two orientations of an edge and can act in place of the edge when a particular orientation is needed. There are precisely two directed edges for each edge: the positive and the negative incarnations.

It is important to distinguish the concept of directed edge from that of edge. An edge is a topological entity that has a spatial definition. A directed edge represents an oriented relationship with its associated edge and is not truly a geometric object at all. The directed edge can be considered as an alias for its underlying edge. To preserve the order of the directed edges about a face, links are provided from each directed edge to the one next counterclockwise about the face (see Figure 2).

Figure 2: The Geometry of the Topological Classes



THE FEATURE STRUCTURE

At the topological level of TIGRIS data, each object corresponds to a fundamental geometric entity (node, edge, or face). The next level collects topological objects into features components. Feature components are subclassed as point, line, or area depending on whether they are composed of nodes, directed edges, or faces, respectively. Feature components represent the simplest, physically homogeneous

features represented in the data, (road segment, river reach, forest stand). The next and subsequent levels collect feature components and other features into more complex and abstract entities (river systems, roads, administrative districts).

These classes provide the data structures with which a schema can be built. In the schema, each cartographic entity is defined as a "dynamic class." A dynamic class definition consists of three pieces of information: the name of the base class (point, line, area feature component, or feature), a list of attributes, and any additional channel restrictions in terms of which dynamic classes each object channel may contain. For example a river system might have a "name" attribute and its composed of channel might contain stream segments and lakes (see Figure 3). All inter-object relations are in Figure 4.

Figure 3: A Complex, Heterogeneous Feature

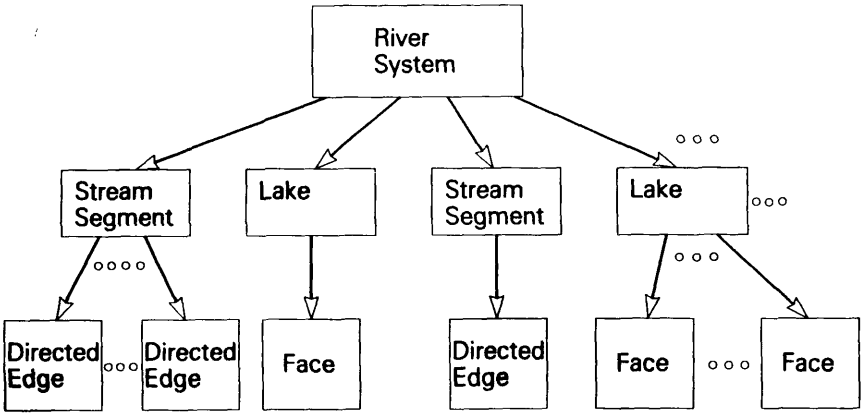
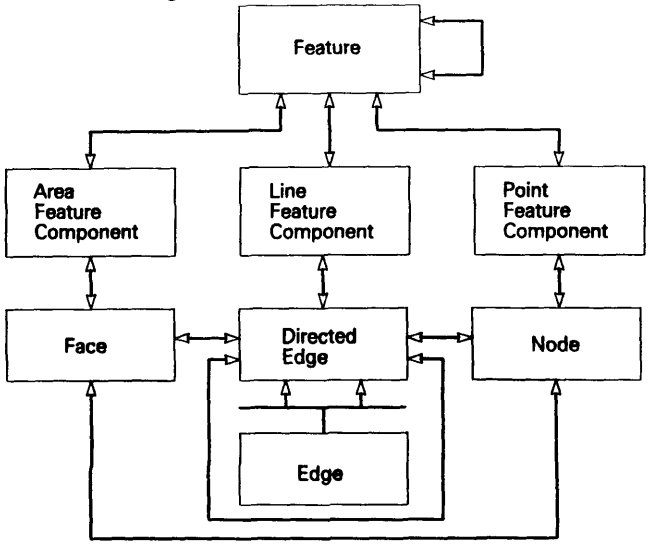


Figure 4: Class Relations



ALGORITHM DESIGN

This section describes some basic considerations in the algorithm design for the edit and query of TIGRIS data, and shows how the object oriented methodology has led to an efficient implementation of the ideas explained previously.

Spatial Localization: Intersection Search

One of the keys to the speed of the TIGRIS editor is the localization of algorithms. The topological structure expresses geometric relationships without reference to the coordinate data. Thus, if a process can be made to deal with the geometry in a localized manner, then topological linkages can be used to traverse the data in the order the process needs it. Thus, process time becomes a function of local complexity.

A example of this localization is the process of adding a new line feature to the data set. In this process, a coordinate string is specified and the topology updated to add the appropriate edges to the data set. The update of the relations for the new edge, although not simple, is straight forward, once the two associated nodes and the face to be crossed have been determined. The bottle neck is the search for the intersections of the new coordinate string with itself or with existing topology (the position of the nodes). The algorithm described here is a simplified version of the one used in the editor, but it suffices to illustrate the point.

The first step is to locate the position of the starting point of the new line feature with respect to the topology. A global r-tree index search (a subject for another paper) of the data locates the face, edge, or node upon which the first point lies. This is an index send of the query message "is on" to all topological objects, nodes, edges, and faces. OM terminates the send mechanism when a successful answer is obtained. If necessary, a node is added at this point.

With the determination (or creation) of the starting node and the specification of the second point, the add line process localizes. If the first node is isolated, the first intersection of the new line with the existing topology will occur on the boundary of the face in which this node lies. When the first node is not isolated, the exit angle of the first coordinate determines which face (or edge) the line moves through. This is a send of the query message "compare exit angle" from the node to each associated directed edge. In all cases, the search for the next intersection is a local problem.

Once the line has moved away from existing edges and nodes, for each new point, the line segment from the last point is compared with the edges and nodes bounding the face being crossed. Since the face is now known to the line method, a "find cross" message is sent to it with the new line segment. The face passes the message on to its isolated nodes and directed edges. The directed edges pass the message on to edges and nonisolated nodes. If no

intersection is found for this face, then it is known that no intersection exists and the next data point is taken. Once an intersection is found, the topology is modified and the feature is integrated into the data set up to the point of intersection (now a node).

At this node, the exit angle of the line is compared with existing edges to determine which face or edge the line will follow when exiting the node. A new face is established and the process repeats until the last point in the line is placed. The entire add line process is accomplished with minimal unproductive intersection search.

Spatial Factoring: Green's Formula and Point-In-Polygon

Another key to programming in TIGRIS, is the spatial factoring of algorithms. This approach to algorithm design breaks the process into subprocesses determined by the spatial subcomponents.

A prime example of spatial factoring is the line integral used to calculate the area and orientation of polygons such as face boundaries. Green's formula (a special case of Stokes' theorem) equates twice the area enclosed within a positively oriented closed curve and the integral of x dy - y dx along the curve itself. Integrals are the prime example of spatially factorable problems. The reason for this is that if three points, P1, P2, and P3, lie on a curve, and G(.,.) is an integral expressed as an operator on the limits of integration then

$$G(P1,P2) = G(P1,P3) + G(P3,P2).$$

and

$$G(P1,P2) = - G(P2,P1)$$

Thus, Green's integral around the boundary of a face is equal to the sum of the integrals along the associated directed edges. Further, the integral along a directed edge equals the integral along the edge multiplied by the orientation. Thus, a face, queried for its area, zeros a variable and sends it as a parameter in a Green's integral message to all of its directed edges. The directed edges pass on the message to the edges, adding their orientation to the parameter list. The edges, which store the geometry, calculate the integral, multiply by orientation, and sum in their contribution to the passed parameter. When completed, the parameter is the area of the face.

The point-in-polygon test for faces is similar. The problem is, given a coordinate point and a face, to determine if the point is interior to the face. The classic solution is to intersect a ray from the point to infinity (such as the one parallel to the x-axis) with the boundary of the face and count the intersections (ignoring noncrossing tangents). If there are an even number of crossings (or none), then the point is not in the face. If there are an odd number of crossings, then the point is interior to the face. In TIGRIS, this has been modified to count crossings with orientation, 1 for an upward going crossing and -1 for a downward going crossing. The answer returned by a face is either 0 or 1 (-1 is possible for

face 1). The crossing count function acts much as an integral, except that geometric observations can simplify the calculations. If the edge's minimum bounding rectangle does not intersect the ray, then the edge's crossing count is zero (no cross can occur). If the edge's minimum bounding rectangle does not contain the point, then the count is the same as that of the line joining the start node and end node of the edge (other crossings cancel in pairs).

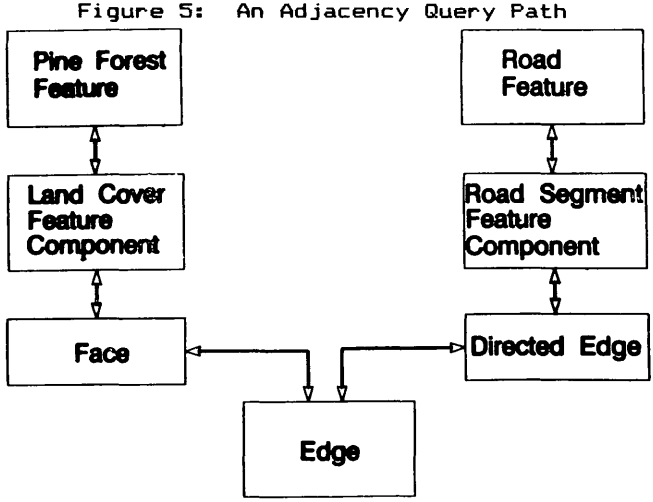
In the point-in-polygon, the factoring did not lead directly to an increase in efficiency, but allowed optimization based on geometric reasoning. In the topological version, each edge, except those whose minimum bounding rectangle includes the point in question, is treated as a whole, the count made using only the end points. In real data, the number of coordinate points in an edge can average 50 to 100 points. Thus, an increase in algorithmic efficiency by a factor of 50 to 100 can be reasonably expected.

Spatial Without Coordinates: Adjacency

In TIGRIS, all features derive their spatial extent through relationships to topological objects and all topological relationships are explicitly stated or easily derived from explicitly stated relationships. Thus, all queries based on topological relationships can be answered without reference to coordinates.

Consider the query "select all pine forests adjacent to roads." The forest half of the operation entails finding the pine forest, following a sequence of channels to locate all faces that make up the pine forests, and then following the face channels to all edges that lie adjacent to the pine forest (see Figure 5). The road half of the operation follows line to directed edge to edge channels to find all edges that are roads. By comparing these sets of edges, the query can be answered easily.

Figure 5: An Adjacency Query Path



An easy implementation of spatial query would use a query mask stored within the objects. In the example above, a message is sent by an index on forest type to the pine forest, and is passed on to the pine forest faces. The faces pass the message to the directed edges which pass the message to the edges to clear the query mask. A message is sent through roads, to directed edges, to edges to set the mask. A final message is sent through the forest (to faces, to directed edges) to the edges to check the query mask. The edges with the query mask set are the ones where roads lie adjacent to pine forest. The pine forest object sending the message is the one adjacent to the road. The sequence of messages have identified the pine forest adjacent to the road as well as the location of the road.

The time required to perform this spatial query is the sum of two message propagations through the pine forest and one propagation through the roads. This is a significant difference from the problems involved in a pure spatial intersection. Further, the algorithm takes great advantage of the parallelism of the message-send subroutine stack and the object relation graph, since when a edge finds a set query mask, it returns a success code directly to a method on the member of pine forest that is its owner.

CONCLUSIONS

The synergism between the topological representation of geographic data and the object oriented design philosophy has enabled TIGRIS to meet its major design goals of providing the geographic user with an interactive edit, query, and spatial analysis tool based upon integrated topological, spatial and attribute data. This in turn, should allow TIGRIS to redefine the capabilities expected of a GIS.

References

Artin, E., and Braun, H., 1969, Introduction to Algebraic Topology, Charles E. Merrill Publishing Company, Columbus, Ohio

Cox, B. J., 1986, Object Oriented Programming, An Evolutionary Approach, Addison-Wesley Publishing Company, Reading, Massachusetts

Lefschetz, S., 1975, Applications of Algebraic Topology, Springer-Verlag, New York

Switzer, R. M., 1975, Algebraic Topology - Homotopy and Homology, Springer-Verlag, Berlin

Spanier, E. H., 1966, Algebraic Topology, McGraw-Hill Book Company, New York

White, E., and Malloy, R., ed., 1986, Object Oriented Programming, BYTE, August 1986, pp 137-233