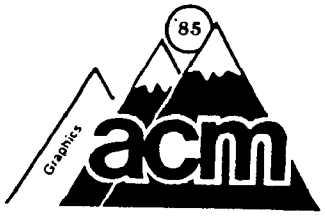


Line Intersection in a Geographic Information System

(Extended Abstract)

Anthony V. Ercolano
Interactive Systems Corporation
Littleton, Colorado USA

This talk will discuss the line intersection function, essential to the overlay operations available with some Geographic Information Systems (GIS's). We will discuss the intersection function's use in the context of the overlay operation, its use to correct "bad" input data inherent to the world of GIS's, and its sensitivity to placing cartographic data in fixed sized partitions of space. Also addressed are line shift and performance characteristics of a line intersection function over an entire geographic database.

Introduction

A Geographic Information System will generally utilize a large graphic database containing line segments used to define cartographic features: the boundaries of polygonal features (property ownership, soil type, contours) and linear features (roads, underground cables, delivery routes). A GIS will also associate with cartographic features additional non-graphic information. Such information, not obvious from the "draw from here to there" specifications that manifest on a graphics device, are generally stored separately from the line segment data. This associated information has of late been stored in relational data management systems or other data management systems having an ad-hoc query capability.

An important aspect of a GIS is the ability to perform operations on the cartographic data, extracting information not available from simply looking at a display of the data. To illustrate this idea, suppose that in a geographic database there are lines bounding all land for which ownership information is available. In addition, there are lines bounding all beaver populations, along with lines defining all the counties, cities, townships, tax districts, soil types, leases on government lands, roads, and so on. Finding all instances of a particular combination of features using a simple graphical presentation can be arduous and fraught with error.

A planner for a government agency might wish to ask the GIS to highlight all beaver populations on United States Forest Service land. This kind of query is known as an overlay operation. Queries of this type can be strung together using operators such as AND , OR , NOT , INTERIOR , and EXTERIOR. The difficult thing about answering the query is not the finding of line segments that belong to the USFS or the finding of lines that bound beaver populations (which is generally a straightforward operation). The difficulty is in taking each of those line segments that make up each cartographic feature satisfying an operand of the query and finding where segments from the different operands interact. Segments that make up the cartographic features in general have no "knowledge" of the existence of other cartographic features in the database and in at least one GIS the segments that make up a feature usually have no "knowledge" that they are part of a feature. In essence, the line intersection function is used to make line segments "aware" of what's going on around them. It should also be noted that interactions can take place such that no segments cross each other. A good proportion, however, do involve segment crossings.

Inherent Problems with Digital Cartographic Data for a GIS

When a geographic database is created for analytic purposes, it is not the case that they slap a map on a digitizing table and start digitizing line information. This procedure might be done for a very small amount of information, but for vast data sets most of the line information is obtained from previously hand digitized line data resident in diversely formatted files. The problem is that most of these files deal with only one kind of information. One file might contain only state boundaries, while another file contains all of the township information. It is very likely that this line information came from two different systems, each with their own biases. It is nearly always the case that one set of information will be skewed with respect to another set. Thus, lines meant to be coincident may actually be separated by a small distance in the digitized data. As a brief example, the state boundaries for a study area might come from a federal tape while county boundaries come from a state source. One could have a case where the county borders lie outside the state borders. Part of Colorado would creep into Kansas, an act which both populations would consider heinous. There also are cases where lines that are supposed to be straight jump back on themselves due to human error during digitization. To put it simply, the data that is normally placed into a GIS is very "dirty" and one must assume that it has to be "cleaned" before useful analyses can be done. One must introduce an operation which either cleans up these problems or draws them to the attention of the analyst. The intersection function is one technique for dealing with these problems.

1985 ACM 0-89791-170-9/85/1000-0085 $00.75

Placing Data in a Fixed Size Partition of the Plane

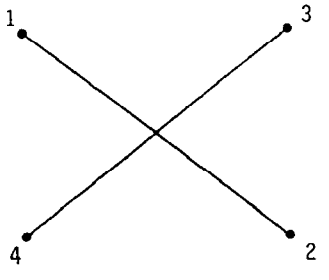
When line data is placed into a system that uses a partition, where each "cell" of the partition has a unique "address", and this address refers to an area rather than a point, there will be a shift in the placement of the lines. This occurs because the end-point specifications for the line will rarely fall in the exact center of a cell. To overcome the effect of the shift, one can store the actual end points at the cell, but this means using extra space in the file to store the original points and additional code to handle this information. As the original data is usually not exact, this rarely produces a good return on the code investment. It could be conjectured that populating all the cells along the line segment path retains more information than simple end-point storage, but this is extremely expensive and the jagged line segments lead to additional data-handling problems. These inaccuracies inherent in a partitioned system, however, rarely lead to significant error.

Real Intersections in a Node-to-Node Structure

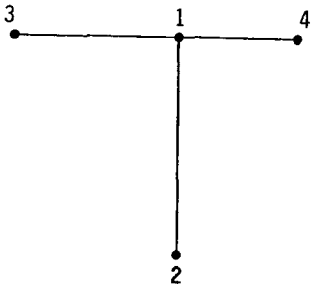
One can define a "real" intersection for two arbitrary line segments to be that place which the two line segments have in common. This will generally be a single point, although in a partition this point refers to an area of space. There are also line segments that are parallel and overlapping (coincident).

We have found it convenient to classify intersections into several categories. The five following figures give examples of our classifications. End points are referred to by integer labels. In these examples, point 1 and point 2 always know that they are connected. This information is placed in the records for point 1 and point 2. Likewise, point 3 is always connected to point 4. Points 1 and 2 have no knowledge, however, that points 3 and 4 exist.

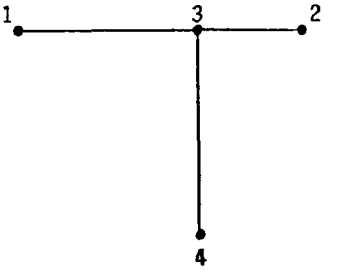
INT_CROSS



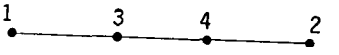
INT_T1



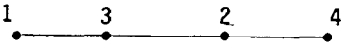
INT_T3



INT_P1342



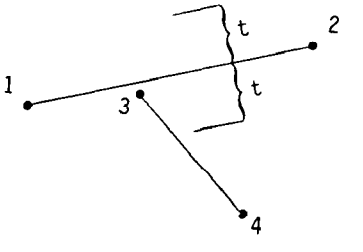
INT_P1324



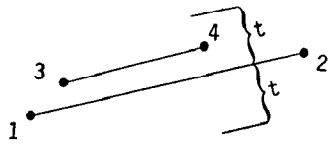
The INT_P####-type intersections are parallel overlapping intersections. As an implementational matter, these intersection types can be stored as permanent symbols and if suitably encoded can be quite useful in the generalization of code for updating the records involved in an operation. There are 13 types of intersections encountered in this way of handling line data: INT_CROSS, INT_T1, INT_T2, INT_T3, INT_T4, INT_P1342, INT_P1432, INT_P1324, INT_P1423, INT_P2314, INT_P2413, INT_P3124, INT_P3214. All intersections fall into one of these categories or the mirror reflection of one of these categories.

Pseudo-Intersections in a Node-to-Node Structure

As mentioned previously, data sets used by the GIS will usually be skewed in relation to each other. When a county boundary and a city boundary are supposed to be on the same line, the segments might actually be separated by a very small distance. It is desirable (and more space efficient) to store the separate segments as single segments. This problem can be approached by introducing a tolerance (t) to the intersection computation algorithm. Generally, the intersection types defined above can be used to represent these pseudo-intersections, and the same code can be used to update the records. In the next figure the intersection code would return an INT_T3 type of intersection.



In the next figure the intersection type of INT_P1342 would be returned.



It is worth mentioning that in all of the intersections discussed, no new records need to be added to the database except in INT_CROSS type intersections. In all of the other types simply reassigning which point is connected to which point is sufficient.

Unfortunately, some end point to end point proximity problems cannot be easily dealt with by intersection operations. These problems will be discussed in the talk.

Ramifications of Intersection Computation Over an Entire File

Calculation of intersections can be a very intensive procedure due to the size of current production databases, which can approach several HUNDRED thousand line segments. Demonstration databases have been created with between 1 and 2 million segments.

The previous identification and update methods are quite suitable for finding the intersection of two line segments in an interactive environment. It is another matter to implement a function to find all lines intersecting a given segment in an automated manner. As above, line shift comes into play. Rather than adding new records, we are simply reassigning which point is connected to which point. In an over-the-file operation, this can cause a line to shift significantly. This shift can become even more pronounced when a tolerance is used.

One very important consideration in the design and implementation of algorithms to handle this problem is processing speed. As has been noted, GIS databases can be very large. It is too expensive to simply choose an arbitrary segment and check all other segments in the database to see if they intersect. A clustering approach needs to be used in which only those segments close to the segment to be checked should be retrieved from the database. Another matter to be considered in this kind of operation is the question of updating the database. It can be reasoned that it is better to first find all the segments that intersect a given segment and then perform the updates on the records involved to prevent excessive database requests and to minimize the shift involved. It might also be desirable to let the analyst specify only certain kinds of lines in the database to check for intersections: it might be the case that religious preference would never be involved in queries involving soil type polygons. An additional consideration is that for large, infrequently changed databases that are queried frequently it might be better to do a single over-the-file intersection operation on all of the data and store the results. This would consume a large amount of time but could be more efficient than repeated intersection testing in response to ad-hoc queries.

In summary, inherent data inaccuracy, line shift, and processing speed can be major problems in overlay operations on Geographic Information Systems. Well-defined intersection operations can be used to alleviate these difficulties.