THE USE OF RANGE-TREE SPATIAL INDEXING TO SPEED GIS DATA RETRIEVAL

by
Bruce Blackwell, Chief Scientist
AUTOMETRIC, INCORPORATED
5205 Leesburg Pike,
Suite 1308/Skyline 1
Falls Church, VA 22041

ABSTRACT

Rapid spatial retrieval of data elements is an essential part of an efficient GIS. Many index structures have been used in the past. This paper discusses the use of a new concept, range trees, in these applications. Range trees are well suited to indexing GIS data elements which have finite extents in the 2-D plane and which arbitrarily may be clustered. Range trees are fast and well structured for dynamic disk resident indices. Furthermore, they are readily extensible to multiple dimensions, raising the possibility of volume searches and even extension to attribute space.

INTRODUCTION

Range-trees, hereinafter referred to as R-trees, were first introduced as a spatial indexing strategy for multi-dimensional data by Guttman (1984). Their development was guided by the inadequacy of other indexing methods for handling data elements of finite extent and of arbitrary distribution in a space most common of two dimensions, but more generally of any number of dimensions. Competing index structures include binary trees, cell methods, quad-trees, k-d trees, and K-D-B trees. All of these suffer from one or more severe limitations in geodata applications. Binary trees are based on only one dimension. Even if multiple trees are built to handle more dimensions, retrieval "bands" must be intersected through sequential comparisons to find the desired data elements. All methods require specification of boundaries in advance and are hence inefficient if clustering of data elements occurs, as commonly happens with GIS data sets. Much work has been done on quad-trees and k-d trees, but the application of these structures almost requires that they be implemented in random access memory, since the node sizes are smaller than any common physical disk storage device data block, and are therefore inefficient for disk resident indices. K-D-B trees cannot index data elements of finite spatial extent. R-trees do not suffer from any of these limitations. Furthermore, R-trees are the only indices in current use that are readily extensible to more than two dimensions for special applications.

A full description of the structure, creation, and use of R-trees is contained in the literature (Guttman, 1984). Only a review will be given here, with emphasis on GIS applications. Figure 1 is an example of a portion of a GIS arc-node database. A single arc is highlighted with a bold line, together with a box known as the minimum bounding rectangle (MBR) of the arc. The MBR is a rectangle with sides parallel to the axes of the coordinate space which defines the minimum and maximum spatial extent of the coordinates delineating

the data element, in this case an arc. There would also be MBR's associated with nodes and polygons in the database. The MBR of a node is, of course, a degenerate case. MBR's have been used frequently in GIS designs as the basis of spatial research for data elements. It is the MBR's of data entities which are used in the R-tree index. An R-tree is a balanced tree structure wherein each R-Tree node contains a number of entries, and each entry consists of a pointer to a child node and the MBR of the child node. The MBR of a node is the least rectangle containing the MBR's of all its children. Two levels in the R-tree have special significance. There is a single node at the beginning of the tree called the root. At the end level of the tree, the nodes are called leaves and the child pointers are to database entries themselves rather than to lower level nodes in the tree. Recursive algorithms for initially populating, updating, and searching the R-tree have been well defined (Guttman, 1984). A small R-tree example is shown in Figure 2.

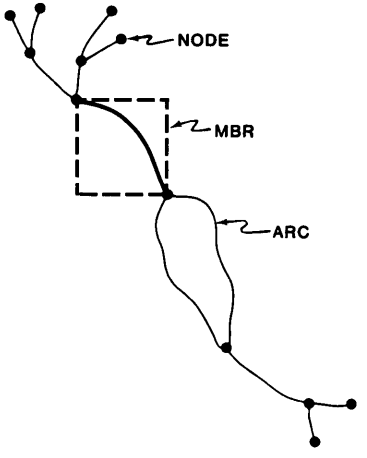

Figure 1. R-tree Example

Two aspects of R-trees are particularly important in applications: node size and node-splitting. Two parameters determine the number of child entries in each node; m, the smallest number of entries allowed, and n, the largest number allowed. No node, except the root, can have fewer than m entries. The parameter n is critical to input/output efficiency for disk-resident R-trees and should be chosen so that the physical disk block size B is given by:

$B = (S_{MBR} + S_{PTR})\ n + O$, where:

S_{MBR}, S_{PTR} are the storage required for the child MBR and pointer entries;

O is the overhead in each node, flags, etc.; and

B typically ranges from 256 to 2048 bytes for different direct access storage devices.

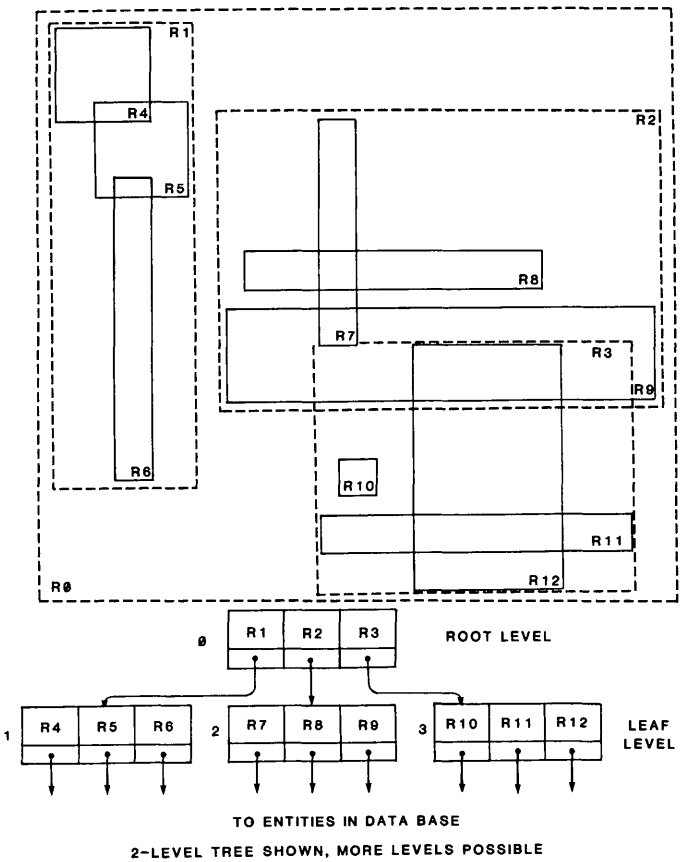

Figure 2. R-tree Spatial Search Index

When a new entry is to be made in the tree, the algorithm (Guttman, 1984) picks the leaf node for insertion which would have to be enlarged least to accommodate the new entry. If this leaf already contains n entries, it must be split. Node-splitting is critical to future search performance of the tree. Exhaustive, quadratic cost, and linear cost algorithms were explored by Guttman (1984). In my laboratory, I have employed a modified form of Guttman's linear cost algorithm with excellent results. Recently, it has been suggested (Roussopoulos and Leifker, 1985) that for relatively static spatial databases, a special packing algorithm be employed for initial population of the R-tree, since Guttman's recursive insertion can lead to inefficiencies in terms of total coverage area and coverage overlaps. While true, it has been my experience that in a GIS production environment the R-tree is built from the bottom for each geounit during data extraction, meets interactive search response time requirements, and the commonality of software between initial population and dynamic updating is a great advantage. There is nothing to prohibit later restructuring of the R-tree with optimized packing algorithms if desired, once a condition of data status is achieved.

EXPERIENCE WITH R-TREES IN A GIS LABORATORY

Application

Autometric, Inc. is working with a GIS in its laboratory which is a fully topologically integrated arc-node data structure. Cartographic information, that is, features, are maintained in a separate set of records which link component topological node, arcs, and polygons. Stored with the features, or in a companion relational database, are feature attributes. The topological arcs and nodes carry the spatial component of the information. For each geounit, then, there are two separate file aggregates which are networked together, the topological/spatial and the feature files. Each entity in each file has an MBR associated and stored with it. The MBR of a feature is derived by combining the MBR's of its component topological parts.

R-trees are built for both the topological and feature files as each element is entered in the database interactively at the workstation. The R-tree serve several purposes:

Display Windowing. The user is able to select, via cursor or keyboard entry, a new display window. The geographic coordinates of the new window are used as input to an R-tree search to determine the list of entities to be displayed. If a symbolized display is desired, the search proceeds through the feature R-tree, and graphics lists are derived subsequently by following links to the topology. If a topological network display is desired, the search proceeds through the topological R-trees.

Cursor Location in the Topology. Formation of topology is done "on-the-fly" while digitizing or editing from photo or cartographic source. The operator can place the cursor anywhere in the geounit area, create a node, snap to an existing node or arc, and digitize arcs. The topological R-tree is used to rapidly obtain such information as the containing polygon, the nearest node, or the nearest arc. Upon completion of digitizing an arc, the arc MBR is built up from its component spatial coordinate list and is passed through the R-tree to retrieve any other arcs in the neighborhood. These arcsare tested for intersections with the candidate arc. If the arc is acceptable, it is entered in the database and in the R-tree. If the arc divides a polygon, two new polygons are formed and the old is deleted, both in the database and in the R-tree. The R-tree, therefore, serves as an adjunct to database navigation and edit.

Entity "Pick" Functions. Cartographic features are assigned to component topological entities by picking nodes, arcs, or polygons by placing the cursor on or near them. The R-tree is used to find rapidly one or more candidates in response to the "pick" request. The candidate(s) is then checked on the basis of a spatial tolerance before being assigned to the feature.

Implementation

In the Autometric GIS laboratory, R-trees have been implemented, along with other GIS functionalities, on a VAX 11/750 computer. The workstation consists of an APPS-IV analytical plotter with graphic superpositioning, an ALTEK digitizing table, a LEXIDATA color graphics terminal, and an A/N CRT and keyboard. The R-trees are disk-resident with a physical record size of 256 bytes so that each

R-tree node has a maximum of 12 entries. A typical data set size for a GIS geounit is 30,000 to 50,000 topological entities and 8,000 to 12,000 cartographic entities.

Performance

With the data set size and R-tree organization described in the above paragraph, virtual windowing and entity pick functions can be performed with response times adequate to support interactive operations. The R-tree search time is well described by the following relation:

$$T_R = A\ m \log n + B$$

where

T_R = retrieval time, wall clock

m = the number of retrieved entities

n = the total number of entities in the R-tree index

A = a constant

B = b constant

In our application, A is about 2.5×10^{-4} seconds and B is about 4 seconds. Note that the search time is linear in number of retrieved entities but logarithmic in the size of the database. The slow search time growth with tree size is a main advantage of tree structures.

Future Possibilities

Most of the nodes in an R-tree are at the leaf level, but most of the node traversals during search, intersection, and deletion take place at levels in the tree above leaf level. An obvious R-tree usage speed improvement could be obtained by holding all R-tree nodes above leaf level in random access memory. The number of node entities above leaf level is approximately equal to n/m (m-1) where n is the total number of database entries and m is the average node fill rate. For n = 50,000 and m = 10, the number of nodes above leaf level in 555, and these can be held in 140 k bytes of memory. Preliminary work reveals that a performance improvement of between 6 and 10 to 1 is possible with this mechanism.

R-trees are in no way restricted to two-dimensional spatial indexing. They generalize readily to a space of an arbitrary number of dimensions. An obvious extension is to include the elevation of features in a 3-D minimum bounding rectangular solid (MBRS) and use the MBRS's to build an R-tree index. Such an index might be useful for the use of a GIS for the production of special products such as air navigation hazard guides. Less obvious is the possibility of treating attributes, properly hierarchically coded, as a "dimension" of an abstract space, and including the attribute dimension in the R-tree index. Such an implementation would permit rapid simultaneous spatial and cartographic layer retrievals from the database. For example, a query to obtain and display all hydrologic features in a horizontal zone could be quickly processed.

REFERENCES

Guttman, A. 1984, R-Trees: A Dynamic Index Structure for Spatial Searching: Proc. of ACM SIGMOD Conference on Management of Data, Boston, June 1984.

Roussopoulos, N. and D. Leifker 1985, Direct Spatial Search on Pictorial Databases Using Packed R-trees: ACM Transactions on Database Systems, Vol.5, 1985, pp. 17-31.