An Efficient Data Structure Layout Design for Spatial Data Organization in Geographic Information System

Animesh Tripathy
Assistant Professor
School of Computer Engineering
KIIT UNIVERSITY, INDIA

Prashanta Kumar Patra
Professor
Department of Computer Science & Engineering
CET, BPUT, INDIA

ABSTRACT

This paper presents a design for map reading based on Corner Stitching for handling the spatial data. Corner Stitching is a technique for representing the rectangular two dimensional objects. It is a technique initially used in VLSI layout editing systems. GIS systems store spatial data using rectangular objects. The concept of corner stitching has been used to reduce the space complexity as well as time complexity pertaining to VLSI Layout Design. These rectangular objects are stitched together at their corners called corner stitches. The data organization & various operations seen, has similarity with certain operations to be performed on spatial data. The data organization is designed in order to support the parallel operations. A query processor for this new data organization has also been developed. The aim of this paper is to explore the extent to which the use of Corner Stitching can be helpful in GIS based system.

General Terms

Spatial Database system and data organization.

Keywords

Corner stitching, geographic data, point finding, area finding, data organization.

1. INTRODUCTION

GIS is best defined as "a decision support system involving the integration of spatially referenced data in a problem solving environment" (Cowen, 1988). GIS can also be defined as "an automated system for the capture, storage, retrieval, analysis & display of spatial data" (Clarke, 1997). The source of GIS data can be classified into categories: (1) the coordinates of different features (point, line, and area) along with the titles from the hand-drawn maps, and (2) logical and numerical information recording the different objects of the maps from the tables associated with the maps [3]. These geographic data can be stored using vector representation or raster representation. Geographic information systems (GIS) are systems that store, manipulate, and display geographic information in layers, sets of data that can be combined with other layers or manipulated and analyzed individually. Results can be seen instantly on a computer screen, in some cases replacing the need for paper maps, freeing the cartographer to experiment with changes in the base map or in the spatial data [5]. In addition to the information content, the map scale, symbols (points, areas, and line styles), colors, type, two and overall layout can be changed quickly, greatly speeding the process of mapping.

In recent years GIS (Geographic Information Systems) and Spatial Information Systems have gone through substantial changes with respect to users, problems, problem domains, and data. Users in decision making positions require rapid, close to instantaneous responses even to complex queries. Users today have access to an unprecedented amount of high resolution and high quality data through scanners, satellites, range finders, medical equipments and other devices. The effect of these changes is rapid and huge, increases the computational demands placed on GIS. To keep up with the computational demands without sacrificing parallel computing, appears to be the only solution. While parallelism in GIS appears to be necessary, the task of providing parallelism is highly challenging, requiring novel ideas and specialized knowledge from areas of computer science outside GIS.

Faust et. al. [8] state that "the real issues in computing in the next decade involve innovations that will allow relatively unsophisticated users to access the power of the computer hardware, without having to become experts in programming and computer operating systems. The challenge therefore is to develop tools for GIS should become easier to use … and at the same time be able to take advantage of the new advances in hardware and software technology." The geographic entities or objects in GIS are based on two different types of data: 'spatial' & 'thematic' data. In turn, spatial data have two components: 'geometric' & 'topological' data. 1) *Geometric data* is quantitative in nature and is used to represent coordinates, line equations etc. Basically, they are of two formats: 'raster' and 'vector' format which are discussed below. The 2D vector format has 3 sub types: 'point', 'line' & 'polygon'. 2) *Topological data* describes the relationship between the geometric data. There are several types of topological relationships: 'connectivity', 'adjacency' and 'inclusion'. 3) *Thematic data* is an alphanumeric data related to geographic entities; eg:- the name and capacity of a road [6].

2. LITERATURE SURVEY

A Geographic Information System (GIS) can be defined as a computer-based system for the digital entry, storage, transformation, analysis, and display of spatial data. Although we often restrict our concept of spatial data to maps (e.g., land use, vegetation), spatial data also include images (e.g., satellite data), point observations (e.g., rainfall), or tabular data associated with geographic areas (e.g., census records) [10]. Thus, in addition to maps alone, a GIS must be capable of handling other types of data, all within a spatial frame of reference. Few would disagree with this definition, but at the same time, many would arrive at a definition on their own. Their definitions would be shaped by their own interests and needs, and how the GIS is to be applied. Thus, a GIS can mean different things to different people [12].

Spatial database system as a database system that offers *spatial data types* in its data model and query language and supports spatial data types in its implementation, providing at least *spatial indexing* and *spatial join* methods. Spatial database systems offer the underlying database technology for geographic information systems and other applications [4]. How spatial database become essential for GIS? To fully understand how ESRI envisions spatial standards and GIS interoperability evolving, it is important to review how spatial standards have evolved over the years. Until the mid-'90s, organizations purchased geographic information systems that closely tied applications to a native, proprietary spatial data model. These early non relational file structures were highly optimized for fast access to data and, being file based, were relatively easy to distribute between sites using the same GIS vendor software. However, the ability to share data among users within an organization was limited by network protocols such as network file system (NFS). Data sharing between organizations with different GIS vendor systems was limited to data converters, transfer standards, and later open file formats. Sharing spatial data with other core business applications was rarely achieved [11].

Gradually, GIS models evolved into *geo-relational structures* where related attribute data could be stored in a relational database that was linked to the file-based spatial features [12]. However, the geo-relational format had limited scalability, and the *dual data structure* (spatial features stored in proprietary file-based format with attributes stored in a relational database) meant that the GIS could not take full advantage of relational database features such as backup and recovery, replication, and fail-over. In addition, supporting large data layers required the use of complex tiling structures to maintain performance, and sharing spatial information with other core business applications was still not possible [9].

In the mid-'90s, new technology emerged that enabled spatial data to be stored in relational databases (often referred to as spatially enabling the database), opening a new era of broad scalability and the support of large, non tiled, continuous data layers. When the new spatially enabled databases were combined with client development environments that could be embedded within core business applications, the sharing of spatial features with core business applications, such as customer management systems, became possible. In addition, these spatially enabled databases allowed organizations to take the first steps toward enterprise GIS and the elimination of organizational "spatial data islands" [7].

3. PROPOSED DESIGN LAYOUT

As discussed above, the geographic data can be represented by using raster or vector technique. *Raster* is a method for the storage, processing and display of spatial data. Each area is divided into rows and columns, which form a regular grid structure. Each cell must be rectangular in shape, but not necessarily square. Each cell within this matrix contains location co-ordinates as well as an attribute value. The spatial location of each cell is implicitly contained within the ordering of the matrix.

In the *Vector* representation, geospatial data is represented in the form of co-ordinates. In vector data, the basic units of spatial information are points, lines (arcs) and polygons. Each of these units is composed simply as a series of one or more co-ordinate points. For example, line is a collection of related points, and polygon is a collection of related lines.

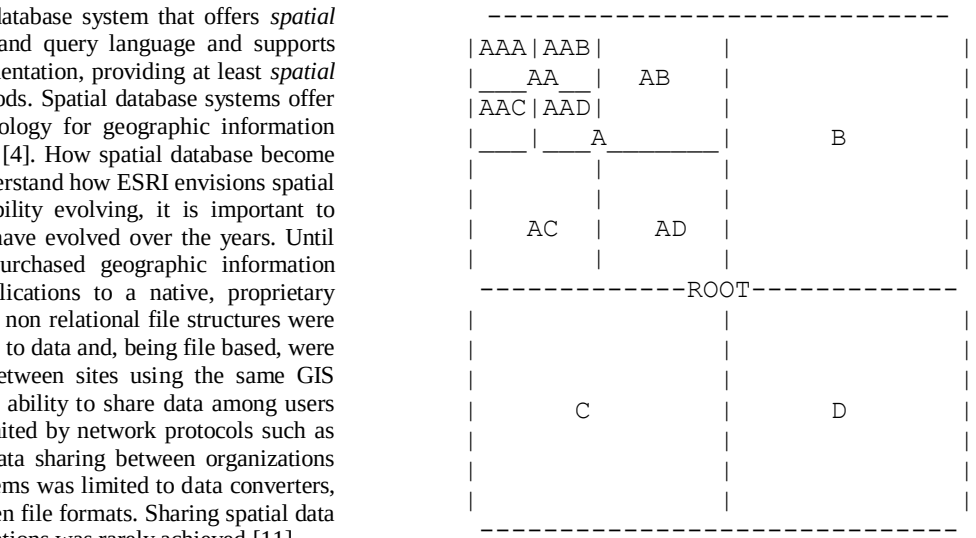

Figure 1 Basic Layout

The spatial data can also be represented by using the Quad tree's. A *quad tree* is generated recursively by successive division of the spatial data into quadrants. Subdivision continues either until every sub-sub-(…) quadrant is homogeneous in terms of the data values it contains, or until a prescribed level of subdivision (spatial resolution) has been reached. Figure 1 shows the subdivision of a three class map down to the fourth quartering (quad level 4).

The resulting quad tree can be represented as a graphical tree (Fig. 2), or as an ordered sequence of spatial addresses (Morton numbers). The Morton number of a cell in the quad tree is derived from the recursive numbering system of the quadrants (shown in Fig.1). Because the quad tree is a hierarchical data structure, operations on the data stored within it can be performed at any desired level of spatial resolution.

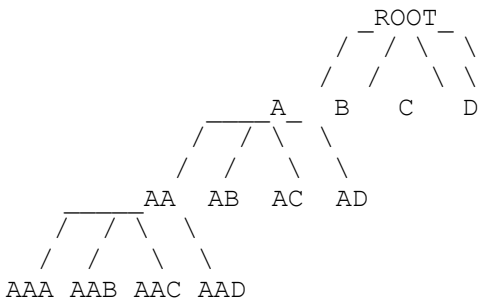

Figure 2 Hierarchical Structure

Along with these techniques there are lot more techniques for representing the spatial data. Some of them had the problem regarding higher complexity for performing the operations and some other had the problem regarding resolution or storage space required. So the development of a data structure that supports parallel operations and at the same time reducing the storage space is very much necessary. This paper discusses a data organization that serves the above said purpose.

3.1 Introduction to Corner Stitching

Corner Stitching is a technique for representing rectangular two dimensional objects. It is a technique initially used in VLSI layout editing systems. The main important feature of this data organization is that the rectangular areas (or objects) are stitched together at their corners. This data organization results in fast algorithms (linear or constant time) for searching, creating, inserting etc., to be performed on spatial data. The rectangular areas are linked by a set of pointers at their corners called *corner stitches* as shown below.

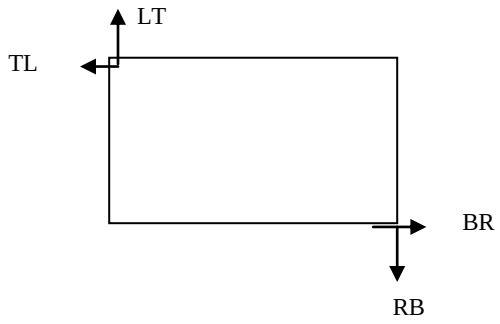

Figure 3 Corner Stitch Data Structure

Corner Stitching is a technique for representing rectangular two dimensional objects. Each rectangular area has four pointers, two at its lower-right corner and two at its upper left corner. Since there is one pointer in each of the four directions, the stitches provide a form of sorting that is equivalent to neighbor pointers. Originally eight pointers were used, two at each corner, but four turned out to be sufficient for algorithms presented here. Corner Stitching is limited to designs with Manhattan features (horizontal and vertical edges only); but within that framework it provides a variety of powerful operations, such as neighbor finding. The algorithms for the operations depend only on *local* information (the object in the immediate vicinity of the operation). Their expected run times are generally linear in the number of nearby objects; in pathological cases (which are unlikely for actual layouts) the running times may be proportional to the overall design size or to the product of nearby objects and design size. Corner stitching is especially effective when the objects are relatively uniform in size, as is the case for low-level mask features. However, it also works when there is variation in feature size. This occurs, for example, in a hierarchical layout where one cell might contain a few large sub cells and many small wires to connect them together.

Corner Stitching permits modification to the database to be made quickly, since only local information is used in making the updates. Most existing systems that provide powerful operations do not provide inexpensive updates: small changes to the database can result in large amounts of re-computation. Corner Stitching combination of powerful operations and easy updates means that many powerful tools previously available only in "batch" mode can now be embedded in interactive systems.

3.2 Corner Stitching in GIS

Using Corner Stitching, Map Reading in GIS is also possible. It supports all the operations like Map Overlay, insertion, deletion, display, selection, network analysis, & simulation to be performed on the spatial data. Basic corner stitching operations such as insertion, deletion, area search, point search, neighbor finding and area enumeration can be used to map basic GIS operations. This paper is a earnest to try out the basic GIS operations using the modified data structure of corner stitching. Since the space and time complexities are very less in this data structure when compared to other data structures that are available for GIS systems, we can use this data structure for very high performance interactive systems. Also, this data structure can be used in systems which have less processing capacity.

In order to make the Corner Stitching technique initially developed for the VLSI system to be used in GIS systems we have made some changes to the data organization. In the initial data organization the empty space is also stored which results in more amount of information to be stored. This will increase space as well as time complexity. So, to reduce the space complexity the empty spaces are not stored. Also, the GIS systems should support the storage and manipulation of Logical and Numerical Thematic data to be stored in order to perform the spatial analysis. The data organization is modified to be capable of storing the LT and NT data. These changes results in a data structure which has a very less space and time complexity.

3.3 Data Organization

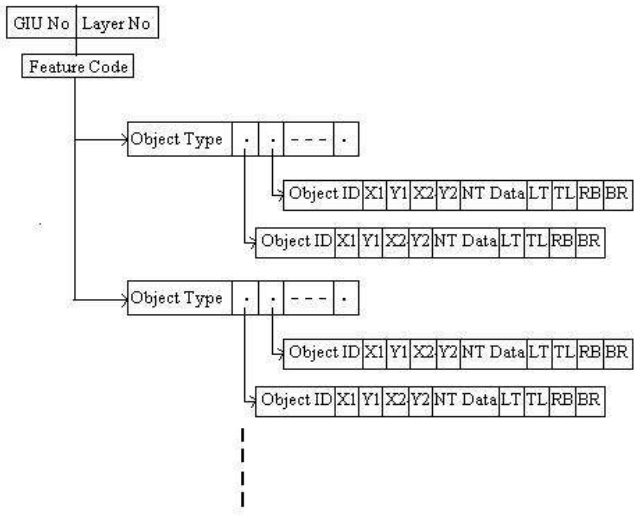

Figure 4 Basic Structure

The data organization consists of two parts. In the first part, the coordinates, the corner stitches and the numerical thematic information are stored in the form of a hierarchical storage structure. This structure is designed in such a way that it will reduce the amount of data to be stored and the data can be handled in an efficient manner. In the second part, some tables have been maintained which are to be used in data manipulation for answering different queries related with the features.

The features that are to be stored in the data organization have been described below.

Object Type: Object Type determines the type of objects that are being stored in that layer. Object Types can be road, temple, building etc.

Object ID: Object ID is used to identify each object uniquely. This is necessary when we want to query the database for performing the spatial operations.

NT Data: Numerical Thematic Data is the thematic data of an object which is a numerical value. For example, thematic value for a building may be the no of persons living in that building.

X1, X2, Y1, Y2: These are end coordinates of a rectangular object. These values are stored instead of storing each and every value. Thus we will reduce the storage space.

LT, TL, RB, BR: These are the pointer values of the rectangular object which points to the nearest object in that particular direction.

4. SPATIAL DATABASE TABLES

Along with above structure we will also maintain the following tables as discussed.

(I). Coordinates Table

In the above table we will store the end point coordinates of each rectangular object along with the corresponding object ID. By storing only the end point coordinates of the rectangular object we can reduce the storage space. At the same time we can regenerate the map whenever required using these stored coordinate values. This table helps us in performing different operations like regenerating the map etc.

GISCOORDINATES (OBJECTID, X1, Y1, WIDTH, HEIGHT)

(II). Corner Stitches table

In this table, we will store the pointer values of each rectangular object. These pointer values are nothing but the nearest rectangular object that is present in the corresponding four directions for that object. This corner stitches helps us to perform certain operations in a very small time. This table helps us in performing different operations like finding the nearest neighbor etc.

GISPOINTER (RID, OBJECTID, LT, TL, RB, BR)

(III). NT Data Table

In this table, we will store the NT value along with other thematic information about each and every rectangular object. This information is necessary because this data is very much necessary when performing certain spatial analysis operations like finding the population in a given area.

GISNTDATA(SLNO,OBJECTID,OBJECT_TYPE, THEMATIC_DATA)

(IV) GIU table

In this table, we will store the layer no., GIU no. and the feature code. This helps in identifying all GIU's when a feature code is present. Also, we can find all the layers that contain the same GIU no.

GIUTAB (FEATURE_CODE, GIUNO,LAYERNO)

4.1 Advantage of Data Organization

The advantages of the data organization presented above are discussed below.

Applicability of Data Organization

The Algorithms based on Corner Stitching to be used for VLSI layout has been discussed in [1]. How these algorithms work has also been described in [1]. So, here we would not describe how these algorithms work. Here we discuss about how some of those algorithms can be useful in GIS & what advantages can be achieved.

A. Point Finding

This algorithm is used to search which rectangular object contains the given (x,y) location. This algorithm is very useful in GIS to find the NT data present at a point of interest. For example, we want to find the amount of rainfall at a particular location. Using the corner stitching technique if we have 'N' no. of rectangular objects then we can perform this operation in O (n) time, which is very less when compared to the others.

B. Neighbor Finding

Neighbor finding is one of the most commonly used operation in GIS. Several papers have already been published which describes about the efficient way of processing the nearest neighbor queries. Even some papers describe the use of parallelism for processing the nearest neighbor queries as in [to be entered]. Using corner stitching finding the nearest neighbor becomes very easy using the corner stitches. Using the corner stitching technique neighbor finding can be done approximately in O (n) time when there are 'N' no. of objects.

C. Area Search

This is also one of the most commonly used operations in GIS. In area search, we will search whether there are any spatial objects present in a given area. For example, we want to check whether there is an empty space in certain location for constructing a building. Using the corner stitching technique area search can be done approximately in O (n) time when there is 'N' no. of objects.

D. Directed Area Enumeration

This algorithm determines which objects are present in an area. This algorithm is useful in GIS in order to find which objects are to be traversed to go to a given object from present object. For example, we want to go from New York to Washington for this purpose which cities should be visited can be found using this area enumeration. Using corner stitching, area enumeration takes linear time.

Suitability in Parallel Computing Environment

As discussed in [2], due to the very nature of the data organization, parallelism can be applied in different levels of computation for replying to any complicated and time-consuming query. For queries which deal with large volumes of data and involve a good amount of computation, application of parallel logic in different levels yields better result.

Interlayer Parallelism: For any multilayer query, the required computations for each layer can be accomplished in parallel as each layer of information is independent of the other layers. For example, to find the intersection points of the roadways and railways under a district, the selection of the road features and the rail features for the said district from the entire road and the rail map can be computed independently and the selected features could be superimposed to get the final cross-over points.

Intralayer Parallelism: Beside the inherent parallelism in the interlayer query, parallelism also exists in the intralayer

computation. The queries where parallelism can be explored have already been presented. Since, our data organization resembles the existing data organization. The queries discussed there will also work for our data organization with minor modifications.

4.2. Spatial Data Types

Systems of spatial data types, or spatial algebras, can capture the fundamental abstractions for point, line and region with relationships between them and operations for composition. Three common spatial data types are point, line and region. Among the operations offered by spatial algebras, spatial relationships are the most important ones. For example, they make it possible to ask for all objects in a given relationship with a query object, e.g. all objects within a window. One can distinguish several classes.

• *Topological relationships*, such as adjacent, inside, disjoint, are invariant under topological transformations like translation, scaling, and rotation.

• *Direction relationships*, for example, above, below, or north_of, southwest_of, etc.

• *Metric relationships*, e.g. "distance < 100" .

4.3 Querying

In the following three subsections, the fundamental operations needed at the level of manipulating sets of database objects, graphical input and output, and techniques and requirements for extending query languages are described.

4.3.1. Fundamental Operations (Algebra)

We now consider from an algebraic point of view operations for manipulating sets of database objects with spatial attributes. They can be classified as spatial selection, spatial join, spatial function application, and other set operations.

Spatial Selection is used in the literature to describe a selection based on a spatial predicate. For example:

"Find all rivers no more than 100 kms from Bhubaneswar" (Bhubaneswar being a POINT value).

rivers select [dist (center, Bhubaneswar) < 100]

Spatial Join, similarly to a spatial selection, a spatial join is a join which compares any two objects through a predicate on their spatial attribute values. For examples:

"Combine cities with their states."

cities states join [center inside area]

"For each river, find all cities within less than 50 kms."

cities rivers join [dist(center, route) < 50]

Spatial function application is similar to use of aggregate functions in relational database. Some predefined functions are used like operator in query. For example:

"For each river going through Bhubaneswar, return the name, the part of its geometry lying inside Bhubaneswar, and the length of that part."

rivers select[route intersects Bhubaneswar] extend[intersection(route, Bhubaneswar) {part}] extend[length(part) {plength}] project[rname, part, plength]

Here *extend* is used as spatial function.

Other Set Operations manipulate whole sets of spatial objects in a special way; some suggested operations are the following:

Overlay: computes the elementary regions resulting from overlaying two partitions.

Fusion: this is a special kind of grouping. Objects are grouped by some arbitrary attribute values. For example, given a set of region objects with a "land-use" attribute, one can group by land-use to obtain one object for land-use "wheat" with the associated union region.

4.4 Query Optimization

Query optimization is one of the most important tasks of a relational DBMS. One of the strengths of relational query languages is the wide variety of ways in which a user can express and thus the system can evaluate a query. A given query can be evaluated in many ways and the difference in cost between the best and worst plans is our main theme of discussion.

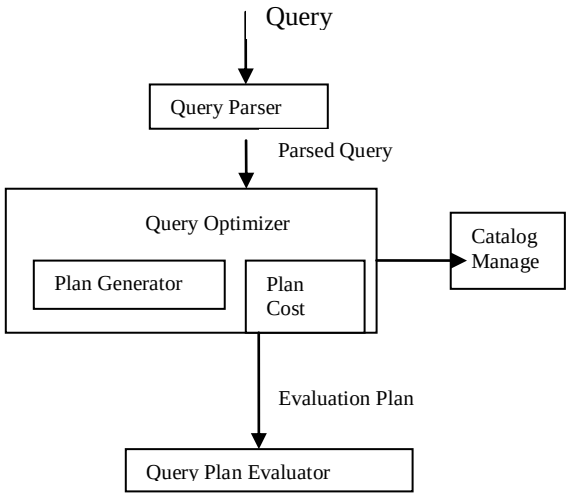

Figure 5 Query Optimization

Realistically, we cannot expect to always find the best plan, but we expect to consistently find a plan that is quite good. Queries are parsed and then presented to a query optimizer, which is responsible for identifying an efficient execution plan.
The optimizer generates alternative plans and chooses the plan with the least estimated cost. The space of plans considered by a typical relational query optimizer can be understood by recognizing that a query is essentially treated as a **σ – Π -** ∞ algebra expression, with the remaining operations (if any) carried out in the result of the **σ – Π -** ∞ expression. Optimizing such a relational algebra expression involves two basic steps:

- Enumerating alternative plans for evaluating the expressions. Typically, an optimizer considers a subset

of all possible plans because the number of possible plans is very large.

- Estimating the cost of each enumerated plan and choosing the plan with the lowest estimated cost.

A query evaluation plan (or simply plan) consists of an extended relational algebra tree, with additional annotations at each node indicating the access methods to use for each table and the implementation method to use for each relational operator.

Consider the following SQL query:
SELECT S.sname FROM Reserves R, Sailors S
WHERE R.sid=S.sid AND R.bid=100 AND S.rating >5

This query can be expressed in relational algebra as follows:

$$\Pi_{sname}(\sigma_{bid=100 \wedge rating>5} (Reserves \infty_{sid=sid} Sailors))$$

This expression is shown in the form of a tree as:

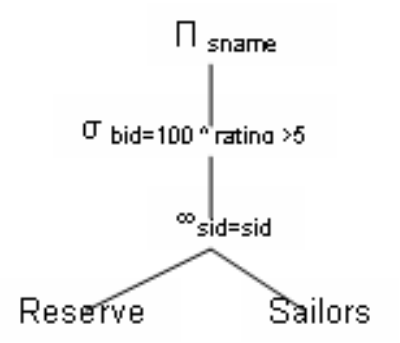

Figure 6 Operator tree

The algebra expression partially specifies how to evaluate the query – we first compute the natural join of Reserves and Sailors, then perform the selections, and finally project the sname field. To obtain a fully specified evaluation plan, we must decide on an implementation for each of the algebra operations involved.

5. CONCLUSION AND FUTURE WORK

We have developed a data organization based on Corner Stitching for use in GIS. We want to explore the extent to which the use of our data organization can be helpful in GIS. In future, we want to enhance this data organization to increase its efficiency. We want to modify this data organization further such that more and more operations can be performed in GIS with less complexity. We want this data organization to be modified such that the map reading becomes more and more simple and at the same time the resolution is not lost. We will develop new algorithms for performing the certain spatial operations which are having less complexity and which will work on our data organization.

In this paper we have discussed briefly about the existing methodologies in Geographic Information Systems and also their advantages as well as disadvantages. Also, we presented a new data organization which is based on corner stitching. We assumed that the spatial objects to be stored are rectangular in shape. As we have discussed some of the algorithms that are used for VLSI layout system based on corner stitching technique can also be used for the GIS based systems. Also, we have discussed the use and advantages of these algorithms. We have also discussed how our data organization supports parallelism for performing the spatial operations.

REFERENCES

[1] John K. Ousterhout, Jan 1984 Corner Stitching: A Data Structuring technique for VLSI layout tools.

[2] R. Dasgupta and S. Bhattacharya, 1996 Parallel Algorithmic Design of an Integrated Geographic Information System.

[3] The National Spatial Data Infrastructure, Jerzy Albin,, 10th EC GI & GIS Workshop, ESDI State of the Art, Warsaw, Poland, 23-25 June 2004

[4] A. Belussi, E. Bertino And B. Catania, 2002 Using Spatial data access structures for filtering nearest neighbor queries.

[5] Shuxin Yuan, 2000 Development of a Distributed Geoprocessing Service Model.

[8] N.L. Faust, W.H. Anderson, and J.L. Star, 1991 Geographic Information Systems and Remote Sensing Future Computing Environment, Photogrammetic Engineering & Remote Sensing 57(6), pp. 655-668.

[9] D. Hutchinson, M. Lanthier, A. Maheswari, D. Nussbaum, D. Roytenberg, and J.R. Sack, 1996 Parallel Neighbourhood Modeling,.

[10] D.M. Mark, N. Chrisman, A.U. Frank, P.H. McHaffie and J. Pickles, 1996 The GIS History Project

[11] Petr Kuba: Data structures for spatial data mining, Masaryk University Brno, Czech Republic, September 2001

[12] Spatial SQL: A Query and Presentation Language; Max J. Egenhofer, IEEE Transactions on Knowledge and Data Engineering 6(1):85-95,1994