Proc. 2nd Symp. on the Design of Large Spatial Databases, Zurich, Switzerland, 1991, in: Lecture Notes in Computer Science, Vol. 525, Springer, 1991, pp. 5-22

The Combination of Spatial Access Methods and Computational Geometry in Geographic Database Systems+

Hans-Peter Kriegel, Thomas Brinkhoff, Ralf Schneider

Institut für Informatik, Universität München, Leopoldstr. 11, D-8000 München 40, Germany

Abstract

Geographic database systems, known as geographic information systems (GISs) particularly among non-computer scientists, are one of the most important applications of the very active research area named spatial database systems. Consequently following the database approach, a GIS has to be seamless, i.e. store the complete area of interest (e.g. the whole world) in one database map. For exhibiting acceptable performance a seamless GIS has to use spatial access methods. Due to the complexity of query and analysis operations on geographic objects, state-of-the-art computational geometry concepts have to be used in implementing these operations. In this paper, we present GIS operations based on the compuational geometry technique plane sweep. Specifically, we show how the two ingredients spatial access methods and computational geometry concepts can be combined for improving the performance of GIS operations. The fruitfulness of this combination is based on the fact that spatial access methods efficiently provide the data at the time when computational geometry algorithms need it for processing. Additionally, this combination avoids page faults and facilitates the parallelization of the algorithms.

1 Introduction

Geographic database systems, also known as geographic information systems (GISs), are one of the most important applications of spatial database systems. Basically, they consist of two parts: First, components to query and manipulate geographical data and second, components to manage and store the data. However, the main purpose of a GIS is to analyze geographical data.

GIS algorithms presented in the past assume that the maps are kept in main memory or in sequential files on secondary storage. The following two important requirements of future GISs demand for new approaches: First, the database system of a GIS must be able to manage very large volumes of data. The large amount of data (in the order of Giga- and Terabytes) is additionally increased by pursuing the goal to manage scaleless and seamless databases [Oos 90]. Second, the database system has to support spatial access to parts of the database, such as maps, and to the objects of a map. Such access is a necessary condition for efficient query and manipulation processing.

Pursuing these goals we want to take advantage of spatial access methods (SAMs). In the past few years many access methods were developed which allow to organize large sets of spatial objects on secondary storage. There are three basic techniques which extend multidimensional point access methods (PAMs) to multidimensional spatial access methods [SK

+ This work was supported by grant no. Kr 670/4-3 from the Deutsche Forschungsgemeinschaft (German Research Society) and by the Ministry of Environmental and Urban Planning of Bremen

88]: clipping, overlapping regions, and transformation. Point access methods such as the grid file [NHS 84], PLOP-hashing [KS 88], the BANG file [Fre 87] and the buddy tree [SK 90] can be extended by these techniques. Additionally, there are access methods which are designed for managing simple spatial objects directly. They use one of the above techniques inherently, e.g. the R-tree [Gut 84] and the R*-tree [BKSS 90] use overlapping regions, or the cell tree [Gün 89] uses clipping. An excellent survey of such access methods is given in [Sam 89].

The use of SAMs as an ingredient in GISs is absolutly necessary to guarantee good retrieval and manipulation performance, in particular for large maps. The use of SAMs enables us to perform operations only on relevant parts of seamless databases. GIS operations on maps modelled by a vector based representation are often very time intensive. Therefore the use of state-of-the-art computational geometry algorithms as a second step of performance improvement is straightforward [Nie 89]. In [KBS 91] we have shown in detail that the performance of the operation map overlay -an important and often used analysis operation in a GIS- can be considerably improved by applying the computational geometry technique 'plane sweep'.

The basic approach of this paper is to partition the seamless databases using SAMs according to the requirements of the GIS operations. Then state-of-the-art computational geometry algorithms are performed on these partitions and the results are combined in order to increase the overall performance of the GIS operations. Thus we combine spatial access methods and computational geometry in order to improve the efficiency of GISs. The combination of these two areas is based on the fact that both use spatial order relations.

The next section describes seamless, vector based databases in GISs. How the efficiency of a GIS is increased by using computational geometry algorithms is shown in section 3. The coupling of spatial access methods and the plane-sweep technique is presented in section 4. An approach to parallelize plane-sweep algorithms follows in section 5. The paper concludes with a summary and an outlook to future work.

2 Seamless vector-based databases in GISs

One important requirement to future GISs is the efficient management of so-called *seamless spatial databases* [Oos 90]. A database is seamless if it does not store sets of map sheets describing only particular small parts of the database, but the whole area managed by the GIS (e.g. the whole world) is stored in one database map. For analysis the user can select any area of interest by a window query. An example is shown in figure 1. This window contains the map which is of further interest to the user. Queries to and manipulations of objects of this map need access to the whole database which is in the order of Giga- and Terabytes. Therefore, the database system of the GIS must be able to support efficient access to any parts of the data on secondary storage.

A GIS is based on two types of data [Bur 86]: spatial and thematic data. *Thematic data* is alphanumeric data related to geographic objects, e.g. the degree of soil pollution. *Spatial data* has two different properties: (1) geometric properties such as spatial location, size, and shape of spatial objects, and (2) topological properties such as connectivity, adjacency, and inclusion.

Topological data can be stored explicitely or can be derived from geometric data.

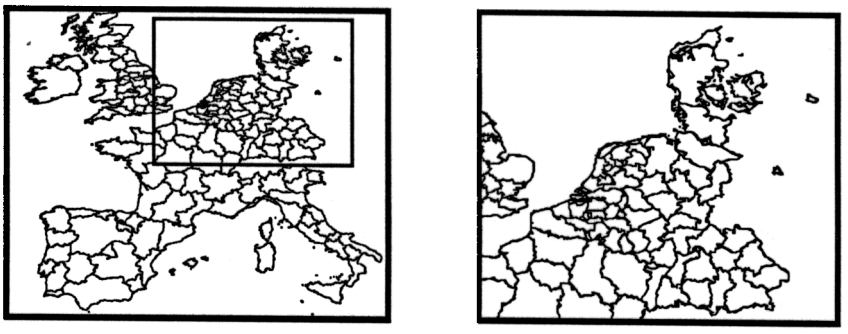
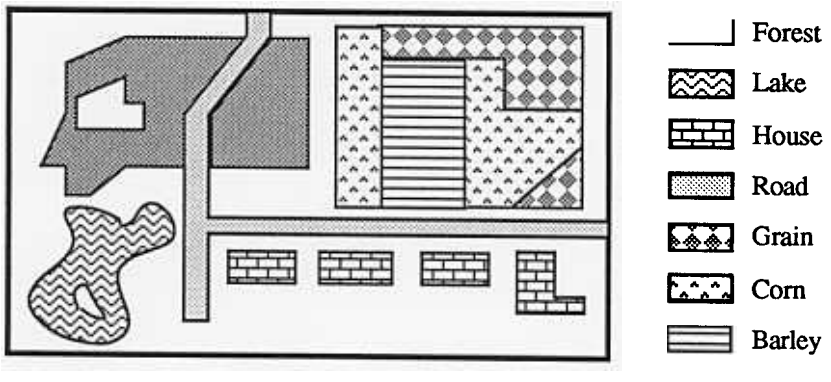


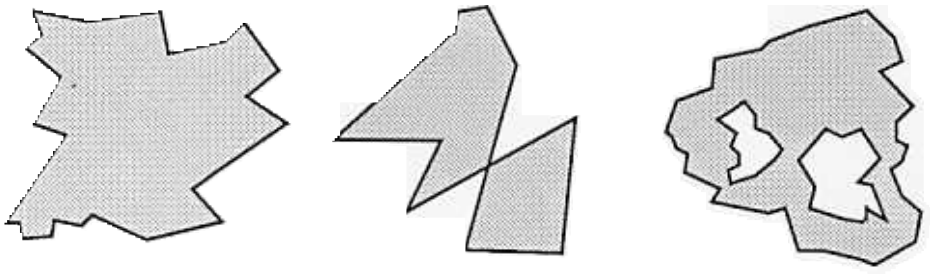
Figure 1: Window query selecting a part of the spatial database

There exist two models for spatial data: vector and raster representations. We consider in this paper only maps modelled by a *vector representation* because there are two main disadvantages of raster representations [Oos 90]: (1) Raster data depends on a specific projection. Therefore, there are problems when combining raster maps from different sources. A scaleless database cannot be realized using a raster representation. (2) Objects in raster maps generally are not handled individually. Thus, a support by access methods is more difficult. Additionally, raster data are more voluminous.

In this paper the term map is used for *thematic maps*. Those emphasize one or more selected topics, e.g. land utilization, population density etc. Thematic maps are generally represented by *choropleth maps* which separate areas of different properties by boundaries [Bur 86], e.g. forests, lakes, roads, or agriculturally used areas (see figure 2).



We assume that the connected areas with the same property are described by *simple polygons with holes*, and that the used data structures are able to handle such polygons explicitly [KHHSS 91a]. A polygon is simple if there is no pair of nonconsecutive edges sharing a point. A simple polygon with holes is a simple polygon where simple polygonal holes may be cut out (see figure 3). There may be other areas in such a hole. The areas of a map are disjoint but they do not need to cover the map completely. Each area refers to exactly one thematic attribute. In figure 2 these characteristics of a thematic map are depicted by an example which visualizes the land utilization of a part of a map.



simple polygon non-simple polygon simple polygon with holes

Figure 3: Different polygons

Below, a formal definition of a thematic map is presented where $\cap_r$ denotes the regularized intersection [Til 80] and T is the set of values of the thematic attributes of M:

$$M := \{ t = (t.P, t.A) \mid t.P \text{ is a simple polygon with holes}, t.A \in T \}$$
$$\text{where } t_1 \in M, t_2 \in M, t_1 \neq t_2 \Rightarrow t_1.P \cap_r t_2.P = \varnothing$$

Maps of different topics describing the same part of the world are called *map layers*.

3 Increasing the performance of a GIS using computational geometry

Efficient algorithms typically use general techniques such as divide-and-conquer or recursion. For algorithms solving computational geometry problems the algorithmic technique called *plane sweep* has proven to be very efficient. In this section we apply this technique to operations in GISs and examine the performance and robustness of such an approach.

3.1 The plane-sweep technique

An algorithm working in the area of GIS should define and utilize an order relation on the objects in the plane to enable a spatial partition of the input maps. Plane sweep is a technique of computational geometry which fulfills this demand [PS 88]: significant points of the objects *(event points)* are projected onto the x-axis and are processed according to the order relation on this axis. Event points are stored in a queue called *event point schedule*. If event points are computed during processing, the event point schedule must be able to insert event points after initialization. A vertical line sweeps the plane according to the event points from left to right. This line is called *sweep line*. The state of the plane at the sweep line position is recorded in vertical order in a table called *sweep line status*. The sweep line status is updated when the sweep line reaches an event point. Event points which are passed by the sweep line are deleted from the event point schedule. Figure 4 depicts an example of the event point schedule and the sweep line status.

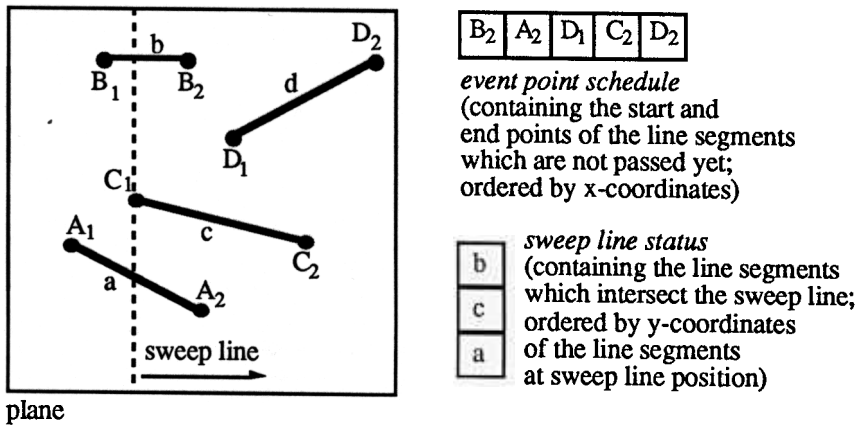


Figure 4: Example of a plane sweep

3.2 Applications of plane-sweep algorithms in a GIS

The map overlay

One of the most important operations in a GIS is the *map overlay*. It combines two or more input maps of different topics into a single new output map. The combination of the thematic attributes or of geometric or topological properties of the input areas is controlled by an *overlay function* f, where f is defined or selected by the user of the GIS. The goals are to derive new maps, to find correlations between the information encoded in maps, and to process complex queries. C.D. Tomlin's map analysis package (MAP) [Tom90] is completely based on the map overlay operation.

We want to illustrate the overlay operation by an example. Figure 5 depicts two input maps 'land utilization' and 'soil pollution'. In the output map all areas should be reported, which are forests or agriculturally used land and where the degree of soil pollution is greater than 2.

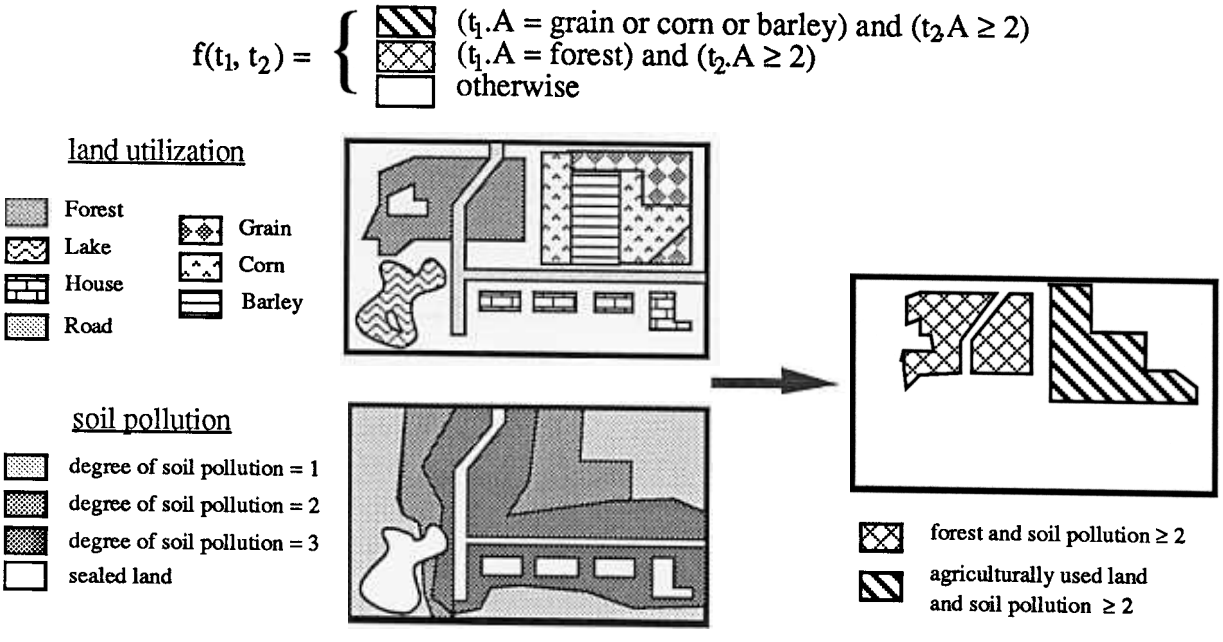


Figure 5: Example of a map overlay and an overlay function

In [KBS 91] we presented an overlay algorithm in detail which was based on the plane-sweep technique. This algorithm is called *plane-sweep overlay*.

The merge algorithm

Plane-sweep algorithms can be used for further problems in a GIS. The *merge operation* is one of them which is closely related to the map overlay [Fra 87]: Its purpose is to merge neighboring areas in one map representing the same thematic attribute (see figure 6). For example, such maps may result from a classification of the attributes or from an overlay with a non-injective overlay function. The neighboring areas with identical attributes can be merged by an plane-sweep algorithm similar to the plane-sweep overlay algorithm. The merge algorithm does not insert edges which separate areas with identical attributes into the sweep line status. Thus the resulting polygons describe the merged areas.

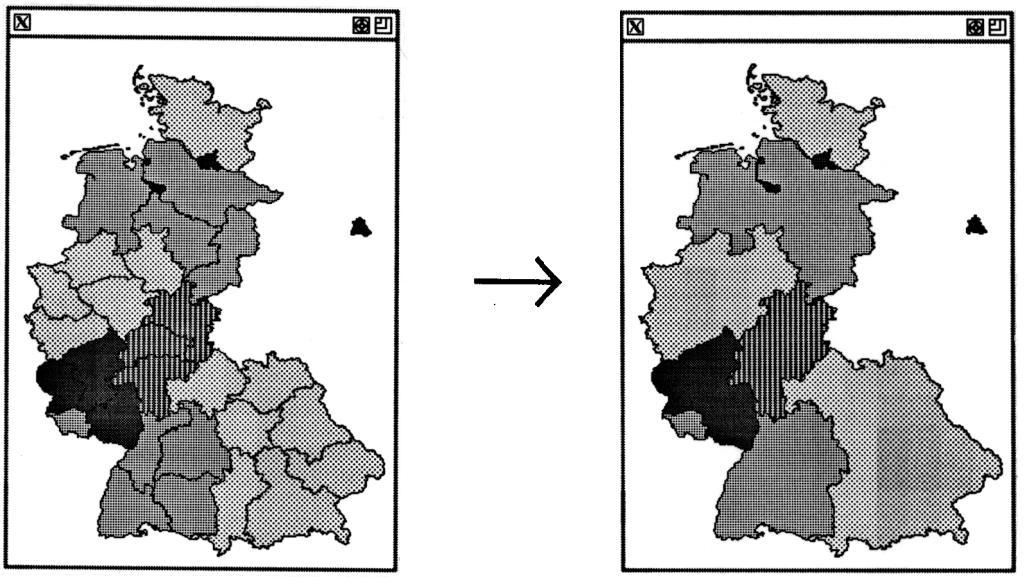


Figure 6: Merging neighbored areas with identical thematic attributes

Geometric computation of polygons from a set of line segments

Another application of plane-sweep algorithms is the following operation: Given a planar graph by a set of line segments, generate the areas limited by these line segments. This operation is needed for example to perform a geometric conversion of spatial data between different geographic information systems. Our implementation of this operation is based on the implementation of the plane-sweep overlay in [KBS 91]. Necessary modifications are an adaption of the intersection treatment and a new calculation of the thematic attributes.

3.3 Performance analysis

In this section we examine the performance of plane-sweep algorithms in an experimental framework. Because the map overlay is the most costly operation of the algorithms mentioned above, we investigated the plane-sweep overlay in the following. The principle results are also valid for the other operations if we consider that those algorithms need not to compute

intersections.

Let n be the total number of edges of all polygons and k be the total number of intersection points of all edges. In [KBS 91] we showed that the worst case performance of the plane-sweep overlay is t (n,k) = O ((n+k) * log (n)) (under the assumption that the number of edges attached to one event point is limited by a constant).

We implemented the plane-sweep overlay algorithm in Modula-2. To examine the performance experimentally, we ran tests on a SUN workstation 3/60 under UNIX. We used a 8 Byte floating point representation which was supported by the hardware and system software. The implementation was developed to demonstrate the properties of the plane-sweep algorithm but it was not tuned for speed. Consequently, there is scope to speed up the overlay.

We performed four test series between two input maps. The maps consist of (1) a regular net of areas to get a constant proportion p of k / n, (2) areas covering the map completely which are generated by a tool, (3) tool-generated areas covering only 50 per cent of the map, and (4) real data. Test series 1 was performed with different proportions p. In test series 4a two maps of administrative divisions of Italy were overlaid where one was translated by an offset. In test series 4b the state frontier of Uganda and lakes near by were overlaid. Typical input maps of the series are depicted in figure 7:

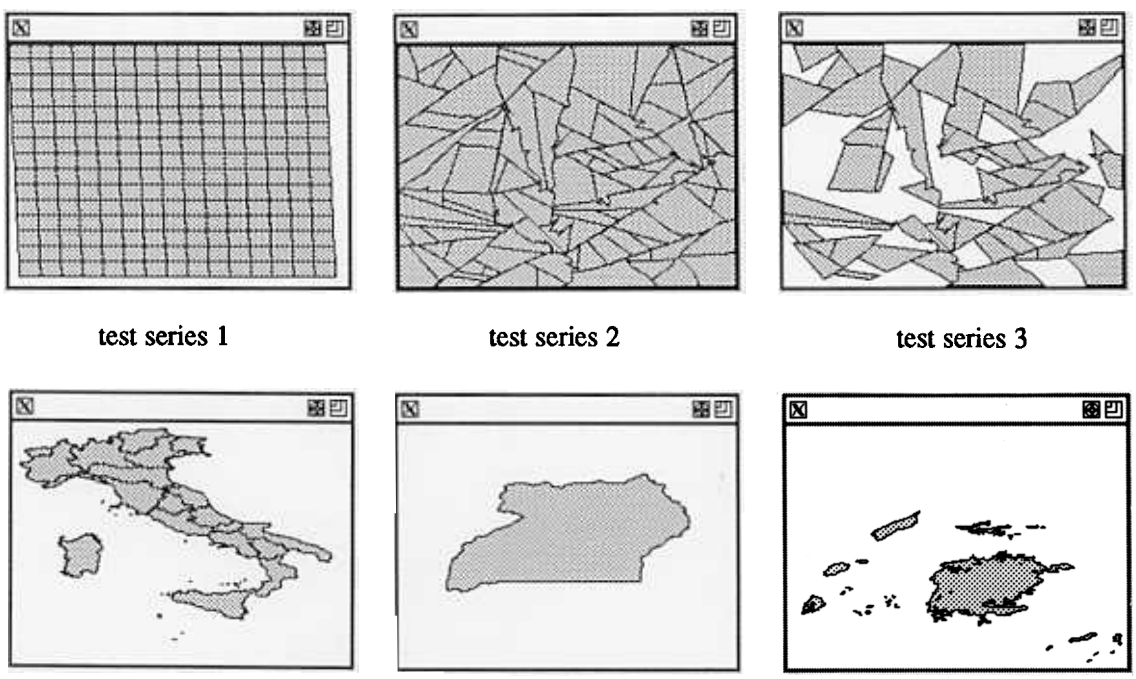


Figure 7: Input maps of the test series

The results of the test series 1 are shown in table 1. t is the used CPU time in sec which is needed to perform the overlay. Additionally, we want to determine the constant c of the map overlay algorithm hidden by the O-notation (c = t / (n * ln n)).

series 1a (p = 0.25):

n	t [sec]	c [msec]
2048	25	1.61
4608	59	1.52
8192	112	1.51
10368	142	1.48
15488	227	1.52
21632	313	1.45
25088	363	1.43

series 1b (p = 0.1):

n	t [sec]	c [msec]
2880	29	1.26
5120	53	1.22
8000	86	1.20
11520	130	1.21
15680	183	1.21
20480	246	1.21
25920	307	1.17

series 1c (p = 0.033):

n	t [sec]	c [msec]
2160	18	1.11
4860	44	1.06
8640	83	1.06
11760	113	1.03
15360	147	0.99
19440	190	0.99
24000	246	1.02

Table 1: Results of the test series 1

The test series of table 1 demonstrate how the constant depends on the number of intersection points. An analysis of these tests results in the following function:

$$t(n,k) = c' * (n + 1.75 * k) * \ln(n)$$

The value of c' in the test series 1a to 1c is approximately 1.05 msec. We would like to emphazise that this constant is very small with respect to performance criteria.

In table 2 the test series 2 and 3 are depicted where c' is calculated additionally.

series 2:

n	p	t [sec]	c [msec]	c'[msec]
13176	0.240	188	1.50	1.02
28837	0.154	372	1.26	0.97
30285	0.161	413	1.32	1.01

series 3:

n	p	t [sec]	c[msec]	c'[msec]
6251	0.262	101	1.85	1.21
14179	0.184	221	1.63	1.20
15260	0.188	245	1.66	1.22

Table 2: Results of the test series 2 and 3

Series 2 and 3 demonstrate another dependency of the running time: If a large number of edges of the polygons coincide (see figure 7), the running time decreases. The reason is that the algorithm detects such edges and combines them.

Table 3 depicts the results of test series 4 with files of real data. In series 4a (Italy) varying administrative divisions and in series 4b (Uganda) varying resolutions are considered.

series 4a (Italy):

division	n	p	t [sec]	c [msec]	c'[msec]
state	6666	0.012	67	1.14	1.12
groups of regions	9622	0.015	94	1.07	1.04
regions	11542	0.014	110	1.02	0.99
provinces	20378	0.023	194	0.96	0.92

series 4b (Uganda, p < 0.004):

n	t [sec]	c[msec]
1852	16	1.16
8973	86	1.05
17829	180	1.03

Table 3: Results of the test series 4

The results of test series 4 demonstrate the validity of the previous results for real data.

3.4 A suitable coordinate representation for plane-sweep algorithms

The instability of plane-sweep algorithms against numerical errors is an objection being raised. This reproach may be justified if a floating point representation is used to compute the intersection points. However rational coordinates are a more suitable representation because they form a vector space. For example, a rational representation of coordinates is used in an implementation of a map overlay in [FW 87].

A more detailed analysis of such a representation leads to the following statements:

1. The coordinates in maps recorded by a GIS can be represented by pairs of integers. This assumption is realistic because both, the described part of the world and the resolution are limited.

2. To compute intersection points, integer coordinates are insufficient [Fra 84]. But the computation of the intersection of line segments described by integer coordinates, needs only a limited number of digits to represent the intersection points by rational numbers. Let n the number of digits of the integers of the input map then the number of digits of the nominator of the intersection points is smaller than 2*n+4 and the number of digits of the denominator is smaller than 3*n+4 (see [Bri 90]).

3. If the input maps of an overlay or of another operation producing intersections, result from an analysis operation (thus containing rational coordinates), the same number of digits as in statement 2 is sufficient for the representation of the intersection points. This is due to the fact that no line segments connecting intersection points are introduced.

Under realistic assumptions rational coordinates of finite precision are an easy, relative efficient, and numerical exact coordinate representation for geographic information systems. Plane-sweep algorithms are absolutly robust by this approach. For an efficient use of rational coordinates an adequate support by hardware and system software is desirable but lacking today.

4 Coupling spatial access methods and plane-sweep algorithms

The database system of a GIS must support efficient query processing as well as efficient manipulation and combination of maps. To fulfill these requirements we assume that the database system uses suitable *spatial access methods (SAMs)* for the management of the database. In particular, this allows to extract the relevant parts from the seamless database (maps). In the following we assume that each map layer is organized and supported by its own SAM because in GISs an efficient access to a map of one topic is desirable, e.g. land utilization or soil pollution.

An often used technique to store areas with SAMs is to approximate them by *minimal bounding boxes (MBBs)*. MBBs preserve the most essential geometric properties of geometric objects, i.e. the location of the object and the extension of the object in each axis. The query processing is carried out in two (or more) steps. MBBs are used as a first *filter* to reduce the set of candidates. The second step *(refinement)* examines those candidate polygons by

decomposing them into simple spatial objects such as convex polygons, triangles, or trapezoids ([KHHSS 91a], [KS 91]). To test the polygons for intersection with a sweep line or query rectangle, MBBs are a sufficient first filter.

In the following we assume that the SAM organizes the access to the objects of a database using a tree-like *directory*. Such access methods are adequate in handling non-uniform spatial data [KSSS 89]. The inner nodes are called *directory pages*, the leaves of the tree are *data pages*. The data and directory pages of a SAM define a partition of the data space. In our case a record in a data page consists at least (a) of a MBB, (b) of the value of the thematic attribute, and (c) of a polygon description or of a pointer to such a description depending on the size of the polygon, see [KHHSS 91b].

As mentioned in the introduction, the database and the maps in a GIS may be very large. Therefore it is not useful to keep all maps in main memory, especially not in multi user systems. In systems with a virtual storage manager the efficiency could decline by a large number of page faults.

Instead of processing the maps completely, it is more efficient to partition the maps and to carry out the plane sweep algorithms on these partitions. One approach is to partition the map using a uniform grid like in [Fra 89]. Obviously, this is not the best way because a non-uniform data distribution is not adequately handled by this approach. We will partition the map by using SAMs and the plane-sweep technique.

Another important reason to partition the maps is the running time of plane-sweep algorithms which is often more than linear. By partitioning we reduce the number of polygons and edges which have to reside in main memory performing the plane sweep. This speeds up the running time for the complete plane sweep.

4.1 Sweep-line partition

For a plane-sweep algorithm only those polygons are relevant which intersect the sweep line. Thus we have a criterion for partitioning by the algorithm itself: Only polygons intersecting the sweep line or close to the sweep line, are kept in main memory. In terms of SAMs this means to read data pages from secondary storage as soon as the sweep line intersects them. We call this approach *sweep-line partition.*

Sweep-line partition and transformation

For example, the sweep-line partition can be realized by the *transformation technique* [SK 88]. This technique transforms the coordinates of a 2-dimensional MBB into a 4-dimensional point. There are two representations of such points:

(1) The *center representation* consists of the center of the rectangle (c_x,c_y) and the distance of the center to the sides of the rectangle (e_x,e_y).

(2) The *corner representation* stores the lower left (x_1,y_1) and the upper right corner (x_2,y_2) of the box.

The 4-dimensional points are stored by a suitable *multidimensional point access method*, e.g. the grid file [NHS 84], PLOP-hashing [KS 88], the BANG file [Fre 87], or the buddy tree [SK 90] .

In the following we use the transformation technique with corner representation. The SAM uses its own sweep line. These sweep line is driven by the partition of the SAM. Performing a plane-sweep algorithm, we must synchronize the sweep line of the algorithm and the sweep line of the SAM. When the sweep line of the algorithm overtakes the sweep line of the SAM, new data pages must be read from secondary storage by the SAM. An example is depicted in fig. 8.

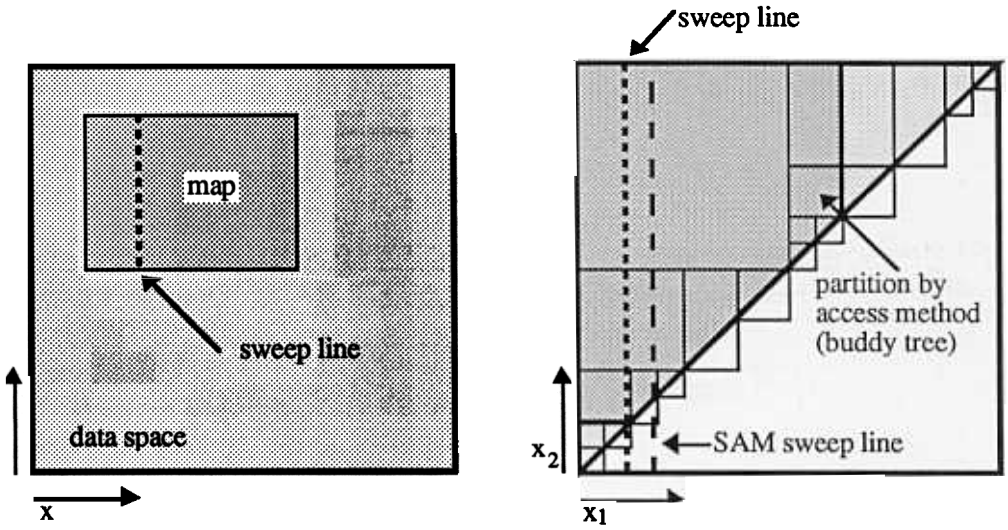


Figure 8: Sweep-line partition and realization by transformation (x-dimensions are shown only)

The sweep-line partition is also applicable to the other techniques (i.e. *clipping* and *overlapping regions* [SK 88]) and to access methods inherently using these techniques, e.g. the R-tree [Gut 84] and the R*-tree [BKSS 90] (overlapping regions), or the cell tree [Gün 89] (clipping).

Performance

Using the sweep line partition reduces the number of page faults considerably because only those parts of the maps intersecting the sweep line reside in main memory. Minimizing the number of page faults during the algorithm improves the overall performance.

This gain of efficiency is only slightly reduced by the following effect: Without partition every required page is accessed exactly once. The pass through the tree of the SAM according to the sweep line may cause several accesses to the same directory page. However, the number of accessed directory pages is, compared to the total number of read pages, generally very small. In table 4, the space requirements of real maps are listed (assumming an R*-tree with pages of 2 KB):

map	data	directory	directory share
Africa (countries)	4679348 byte	3924 byte	0.084 %
Africa (topography)	5528816 byte	46332 byte	0.831 %
Latin America (countries)	5178440 byte	10332 byte	0.199 %
Latin America (topography)	3785440 byte	51480 byte	1.342 %
EC (regions)	1126360 byte	29916 byte	2.587 %

Table 4: Space requirements of data and directory

4.2 Strip Partition

Contrary to a partition driven by the sweep line, an orthogonal partition is possible. To support the plane sweep, it is sensible to divide the map into strips S_i which extend over the whole map *(strip plane sweep)*. In the following we assume proceeding from the top strip S_1 to the bottom strip S_M (see figure 9). The strip partition shortens the length of the sweep line which decreases running time. The height of the strips may vary to adapt the partitions to non-uniform data distributions.

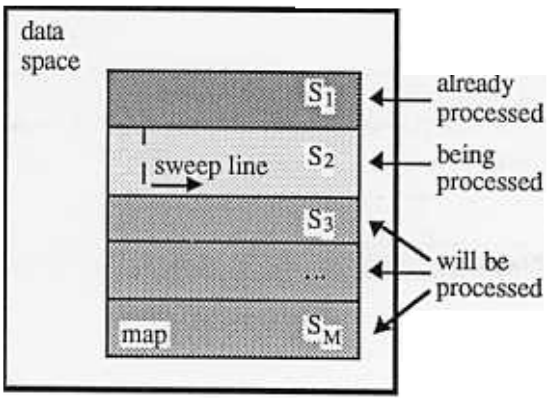


Figure 9: Strip partition

Some areas of a map may intersect more than one strip. One solution is to read all necessary areas for each strip. The consequence is that many data pages are accessed several times. Therefore, this procedure is too costly. Another way is to store such areas temporarily in a buffer. Those areas are an additional input to the next strip plane sweep. Thus, every area of an map can be assigned to exactly one strip S_i and needs to be read from secondary storage only once.

As in section 4.1 we assume that each accessed data page is completely read. Areas not intersecting the actual strip are buffered. The access to the areas of one strip corresponds to standard region queries which supply all polygons intersecting the strip. There is only one exception: Data pages accessed by previous strips are not read again. We call this kind of query *modified region query*. Such queries are performed very efficiently by the R*-tree [BKSS 90], a variant of the well-known R-tree [Gut 84]. This is caused by the minimization of area, margin, and overlap of directory rectangles in the R*-tree.

Generating an optimal strip partition

In the following, we want to describe how an optimal strip partition of the map is generated. An optimal strip partition adapts the strips to the distribution of the areas of the map, exploits the size of main memory, and avoids page faults. The strip partition is best supported by using an efficient SAM, such as the R*-tree.

As mentioned, the areas of each map which are simple polygons with holes, are approximated by minimal bounding boxes, which preserve the location and the extension of the areas. The number of bytes representing an area is assigned to the MBB. This is necessary

because we cannot expect in GIS applications that each area is described by the same number of bytes. Each data page represents a set of areas. Thus, for each data page the number of bytes can be calculated which is necessary to store the data of this page in main memory. This information is stored in lowest level of the directory.

In a preprocessing step of a plane-sweep algorithm, we determine the data pages which intersect the map. Each data page of the SAM corresponds to a region of the data space, e.g. the regions of the R*-tree are rectangles. These regions are sorted in descending order according to the highest y-coordinate of the region. Initially, the buffer is empty which stores areas which are not performed completely by a strip sweep line. According to the order mentioned above the first k regions are determined where the sum of bytes which are represented by these k regions is smaller than the size of main memory minus the size of the buffer. Thus, the first strip S_1 is limited by the highest y-coordinate of the (k+1)st data page. The areas which are not handled completely in the first strip sweep line will be stored in the buffer. The above procedure is iterated.

To illustrate this approach we present the following example where the size of main memory is restricted to 8 mega bytes (see figure 10).

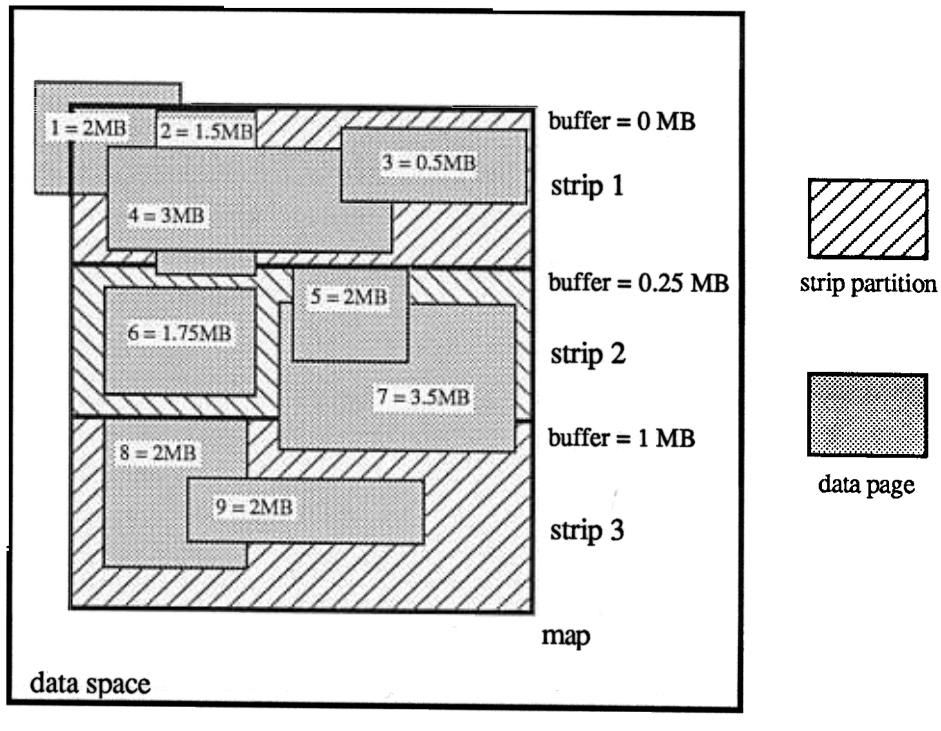


Figure 10: Example of generating an optimal strip partition

The numbers 1 to 9 of the data pages indicate the order mentioned above. The size of the data pages 1 to 4 amounts to 7 MB. With the next data page the size of main memory would be exceeded (7MB+2 MB ≥ 8MB) and page faults could occur. Therefore, the first strip ends at the highest y-coordinate of the data page 5. Let us assume that after the first strip plane sweep 0.25 MB are stored in the buffer. Then the second strip can be extended until 7.75 MB are not exceeded. Thus, the data pages 5 to 7 are associated to the second strip. Finally, the data pages

8 and 9 and the regions stored in the buffer are accomodated in the third strip.

Generating the optimal strip partition is not time intensive because only directory pages are read to get the necessary information, such as the size of the data pages and bytes represented by the data pages. Data pages are only read from secondary storage when the plane-sweep algorithm is performed actually. The ratio of read directory pages to read data pages is very small when performing a plane-sweep algorithm (compare section 4.1).

5 Parallel processing of plane sweeps

In the last years there are many efforts to design, to manufacture, and to utilize computer architectures of multiple, parallel central processing units (CPUs). Computers using such architectures are called *multiprocessor systems* or *parallel computers*. Their main objective is to increase the performance compared to one-processor systems. Future database systems and particularly spatial database systems of GISs have to pursue using such architectures. This is important especially for time-consuming operations such as the map overlay or related GIS operations.

The use of parallel architectures necessitates the development of spatial access methods which support parallel access to the database and (if possible) utilize the parallel architecture. The second goal is the design of parallel algorithms which exploit the parallelism offered by the architectures and the parallelism hidden in the problem in a best possible way. In this section we demonstrate such exploitation of parallelism for plane-sweep algorithms.

There exist different types of multiprocessor systems. We assume that each CPU has its own main memory *(local memory)*. Parts of the memory may be shared. One important characteristic of multiprocessor systems is the communication between the processors. There exist many *interconnection networks* in such systems ([GM 89], [SH 87]), e.g. static networks as rings, trees, hypercubes, or grids. Modern architectures allow dynamic routing between arbitrary processors. In the following, we assume a linear arrangement where each processor can communicate with its direct neighbor. Such a structure can be realized by most interconnection networks.

The strip partition seems to be the best candidate for a parallel execution of plane sweeps. A natural approach is to process the strips simultaneously and independently. But there are some problems: As mentioned in section 4.2, areas exist which intersect more than one strip. If we perform the plane sweeps independently, many data pages must be read from secondary storage several times. This effect decreases the performance of the approach considerably. Another problem is that we may need the results of strip S_{i-1} for processing strip S_i which is e.g. necessary for the plane-sweep overlay of thematic maps without complete cover by the areas.

Therefore, we have to synchronize the strip processing. We introduce a second sweep line for each strip indicating the part of the map which is already processed completely. The first sweep line of strip S_{i+1} is not allowed to overtake the second sweep line of S_i. The process of S_{i+1} is suspended if necessary. An example of parallel strip processing is shown in figure 11:

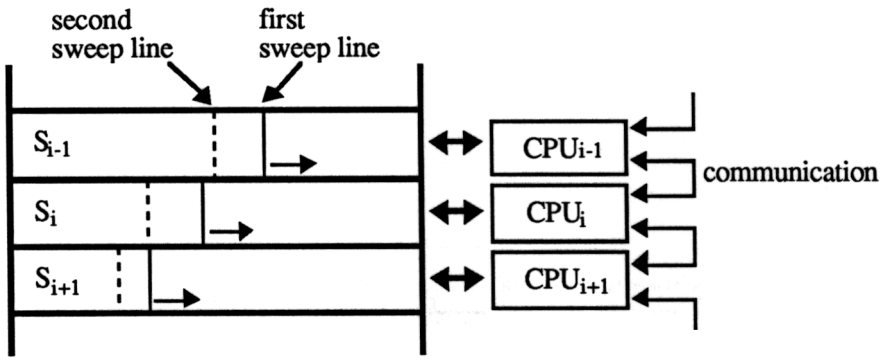


Figure 11: Parallel strip processing

Parallel plane-sweep overlay

This approach can be realized for the plane-sweep overlay [KBS 91] with little extensions to the original algorithm: For maintaining the second sweep line, we need a new data structure L. The actual x-position P of the first sweep line and an identification number of the region are inserted into L when a new region is starting. L is ordered by P and can be implemented using a balanced tree. The position P and the region ID are also stored additionally to the edges in the sweep line status. If the algorithm detects that two regions with different region IDs are identical, the entry with P starting further to the right is deleted from L. When a region is closed, the associated entry is deleted from P. If this entry was the minimum entry, the second sweep line is allocated at the position of the new minimum entry of L. This processing is illustrated in figure 12. Other plane-sweep algorithms can be modified in a similar way.

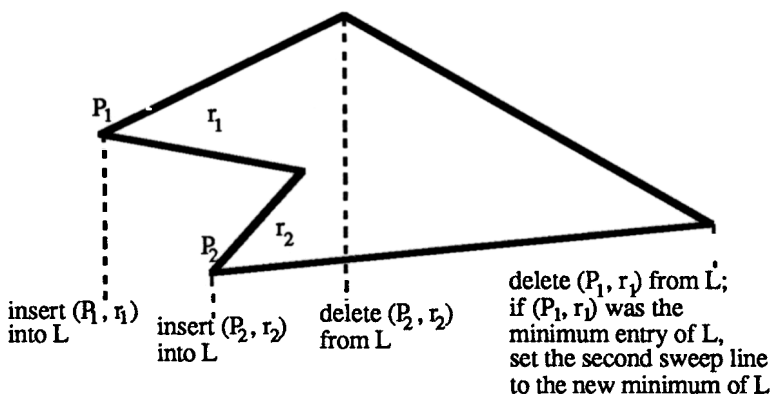


Figure 12: Update of the data structure L

6 Conclusions

In this paper, we demonstrated the fruitfulness of combining spatial access methods and computational geometry concepts, in particular for the plane-sweep paradigm, in order to increase the efficiency of geographic database systems. The marriage of these two areas was enabled by the property that the spatial access method supports the plane-sweep paradigm.

Since plane-sweep processing generates results in sorted order, the spatial access method must be robust with respect to sorted insertions for storing these results. As an example of providing good performance we presented the plane-sweep map overlay which is a very important analysis operation in a geographic information system. Good analysis and retrieval performance are important factors for good user acceptance of a GIS. Thus, in our future work, we will design efficient algorithms based on spatial access methods and computational geometry for all retrieval operations. Performance improvements which exceed those realized in this paper by coupling spatial access methods and computational geometry, are feasible by using processors suitable for rational numbers and by implementing parallel GIS algorithms on parallel multiprocessor systems. These issues are important goals in future work.

Acknowledgement

We thankfully acknowledge receiving real data representing national administrative divisions of the European countries by the Statistical Office of the European Communities. Further real data are taken from the World Data Bank II. Additionally, we would like to thank Holger Horn for making his map generator available to us.

References

[BKSS 90] Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. Proc. ACM SIGMOD Int. Conf. on Management of Data, 322-331, 1990

[Bri 90] Brinkhoff, T.: Map Overlay of Thematic Maps Supported by Spatial Access Methods. Master thesis (in German), University of Bremen, 1990

[Bur 86] Burrough, P.A.: Principles of Geographical Information Systems for Land Resources Assessment. Oxford University Press, 1986

[Fra 84] Franklin, W.R.: Cartographic Errors Symtomatic of Underlying Algebra Problems. Proc. Int. Symp. on Spatial Data Handling, Vol. I, 190-208, 1984

[Fra 87] Frank, A.U.: Overlay Processing in Spatial Information Systems. Proc. 8th Int. Symp. on Computer-Assisted Cartography (Auto-Carto 8), 16-31, 1987

[Fra 89] Franklin, W.R. et al.: Uniform Grids: A Technique for Intersection Detection on Serial and Parallel Machines. Proc. 9th Int. Symp. on Computer-Assisted Cartography (Auto-Carto 9), 100-109, 1989

[Fre 87] Freeston, M.: The BANG file: a new kind of grid file. Proc. ACM SIGMOD Int. Conf. on Management of Data, 260-269, 1987

[FW 87] Franklin, W.R., Wu, P.Y.F.: A Polygon Overlay System in Prolog. Proc. 8th Int. Symp. on Computer-Assisted Cartography (Auto-Carto 8), 97-106, 1987

[GM 89] Gonauser, M., Mrva, M. (eds.): Multiprozessor-Systeme: Architektur und Leistungsbewertung. Springer, 1989

[Gün 89] Günther, O.: The Design of the Cell Tree: An Object-Oriented Index Structure for Geometric Databases. Proc. IEEE 5th Int. Conf. on Data Engineering, 598-605, 1989

[Gut 84] Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. Proc. ACM SIGMOD Int. Conf. on Management of Data, 47-57, 1984

[KBS 91] Kriegel, H.-P., Brinkhoff, T., Schneider, R.: An Efficient Map Overlay Algorithm based on Spatial Access Methods and Computational Geometry. Proc. Int. Workshop on DBMS's for geographical applications, Capri, May 16-17, 1991

[KHHSS 91a] Kriegel, H.-P., Heep, P., Heep, S., Schiwietz, M., Schneider, R.: An Access Method Based Query Processor for Spatial Database Systems. Proc. Int. Workshop on DBMS's for geographical applications, Capri, May 16-17, 1991

[KHHSS 91b] Kriegel, H.-P., Heep, P., Heep, S., Schiwietz, M., Schneider, R.: A Flexible and Extensible Index Manager for Spatial Database Systems. Proc. 2nd Int. Conf. on Database and Expert Systems Applications (DEXA '91), Berlin, August 21-23, 1991

[KS 88] Kriegel, H.-P., Seeger, B.: PLOP-Hashing: A Grid File without Directory. Proc. 4th Int. Conf. on Data Engineering, 369-376, 1988

[KS 91] Kriegel, H.-P., Schneider, R.: The TR*-tree: A New Representation of Polygonal Objects Supporting Spatial Queries and Operations. Submitted for publication, 1991

[KSSS 89] Kriegel, H.P., Schiwietz, M., Schneider, R., Seeger, B.: Performance Comparison of Point and Spatial Access Methods. Proc. 1st Symp. on the Design of Large Spatial Databases, 1989. In: Lecture Notes in Computer Science 409, Springer, 89-114, 1990

[NHS 84] Nievergelt, J., Hinterberger, H., Sevcik, K.C.: The Grid File: An Adaptable, Symmetric Multikey File Structure. ACM Trans. on Database Systems, Vol. 9, No. 1, 38-71, 1984

[Nie 89] Nievergelt, J.: 7±2 Criteria for Assessing and Comparing Spatial Data Structures. Proc. 1st Symp. on the Design of Large Spatial Databases, 1989. In: Lecture Notes in Computer Science 409, Springer, 3-28, 1990

[Oos 90] Oosterom, P.J.M.: Reactive Data Structures for Geographic Information Systems. PhD-thesis, Department of Computer Science at Leiden University, 1990

[PS 88] Preparata, F.P., Shamos, M.I.: Computational Geometry. Springer, 1988

[Sam 89] Samet, H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley, 1989

[SH 87] Siegel, H.J., Hsu, W.T.: Interconnection Networks. In: Milutinovic (ed.): Computer Architecture: Concepts and Systems. North-Holland, 225-264, 1987

[SK 88] Seeger, B., Kriegel, H.-P.: Techniques for Design and Implementation of Efficient Spatial Access Methods. Proc. 14th Int. Conf. on Very Large Data Bases, 360-371, 1988

[SK 90] Seeger, B., Kriegel, H.-P.: The Buddy-Tree: An Efficient and Robust Access Method for Spatial Database Systems. Proc. 16th Int. Conf. on Very Large Data Bases, 590-601, 1990

[Til 80] Tilove, R.B.: Set Membership Classification: A Unified Approach To Geometric Intersection Problems. IEEE Trans. on Computers, Vol. C-29, No. 10, 874-883, 1980

[Tom 90] Tomlin, C.D.: Geographic Information Systems and Cartographic Modeling. Prentice-Hall, 1990