A Cost Model for Query Optimization Using R-Trees

Walid G. Aref
Matsushita Information Technology Laboratory
Panasonic Technologies, Inc.
Two Research Way
Princeton, New Jersey 08540
aref@mitl.research.panasonic.com
Phone: 609-734-7349
Fax: 609-987-8827

Hanan Samet
Computer Science Department and
Center for Automation Research and
Institute for Advanced Computer Studies
The University of Maryland
College Park, Maryland 20742
hjs@cs.umd.edu
Phone: 301-405-1755
Fax: 301-314-9115

Abstract

A cost model is presented for optimising queries that involve accessing spatial objects which are indexed by an R-tree. The model is useful for estimating the selectivity factor as well as the cost of some commonly used spatial operations, e.g., the window intersection and the spatial join operations. Performance studies are conducted that verify the validity of the cost model using real as well as synthetic data sets.

1 Introduction

A spatial database is a collection of objects that span the same underlying space, where each object may have an arbitrary extent. Usually, spatial databases are very large in data volume. Consequently, a spatial database is organised using spatial access methods that provide efficient access and flexible manipulation of the data. There are many ways to represent and organise a set of objects inside a data structure [13]. In the following we review some of these representations.

One way is to decompose a spatial object into disjoint cells. This means that the spatial object is represented by more than one entity inside the data structure. Some example representations include a partition of the spatial object into a collection of convex blocks (the cell tree [8]), a collection of square blocks at predetermined positions (the quadtree [12]), or a collection of rectangles (the R^+-tree [7]). An alternative way is to represent a spatial object by only one entity inside the data structure, e.g., by a point in higher dimensions as in the case of representing an n-dimensional polygon having k boundary points by a point in nk-dimensions and then store it in a point data structure (e.g., the Grid File [16]), or by some conservative approximation of the object as in the case of representing the same polygon by its minimum enclosing rectangle (e.g., the R-tree [10]).

In this paper we focus on the representation of objects by only one entity. In particular, we use the R-tree representation as an example since it is one of the widely used data structures for indexing spatial data in extensible database systems. However, our techniques can be adapted to apply to other similar data structures (e.g., see [2] for an equivalent study for the quadtree).

Many performance studies have been conducted that test and analyse the performance of the R-tree for various types of operations (see [11, 6, 15] for some recent studies). This paper is an attempt to integrate the results of these studies into one model that can be used by an extensible query optimiser to estimate the cost of predicates that involve spatial operations for R-tree based indexes. More specifically, in this paper, we focus on two types of operations: the window intersection and spatial join operations.

In order for an extensible query optimiser to handle a newly added operation into the extensible database system, the optimiser has to be informed of the following aspects about this operation:

- an estimate of the selectivity factor of the operation
- an estimate of the cost of the operation
- the algebraic axioms that relate this operation to the other operations in the algebra

In this paper we focus on these aspects for the two spatial operations mentioned above when the underlying spatial database is indexed using an R-tree. This can serve as a framework for realising a cost model for optimising spatial queries.

The rest of the paper proceeds as follows. Section 2 provides an overview of the R-tree data structure as well as the spatial operations that we address in the paper, namely the window intersection and spatial join operations. Section 3 contains formulas for estimating selectivity factors. Section 4 contains formulas for estimating the execution cost of the operations. Section 5 describes our experimental setting as well as the experimental results while contrasting them with the developed formulas. Section 6 contains concluding remarks.

2 Overview

2.1 The R-Tree

The R-Tree [10] is a balanced tree that stores the spatial data by using the concept of a minimum bounding (or enclosing) rectangle. Objects are grouped into hierarchies of enclosing rectangles. The R-tree is similar to a B-tree with key values in the B-tree nodes replaced by bounding rectangles. However, unlike a B-tree, the bounding boxes of different nodes of the R-tree may overlap. Figure 1 gives

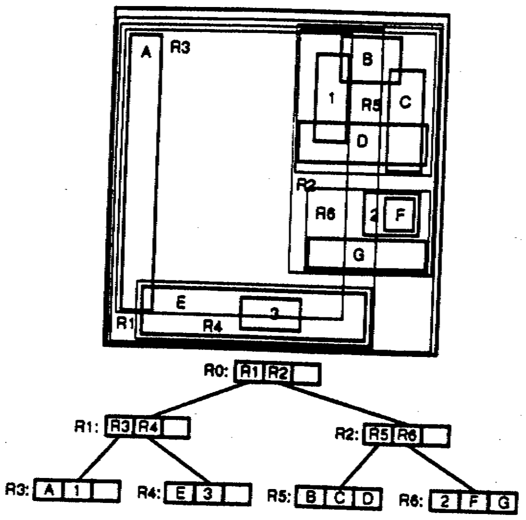


Figure 1: An example illustrating a set of rectangles and their corresponding R-tree.

an example of an R-tree. A spatial data object may be associated with only one bounding rectangle. However, the area that the object spans may be included in several bounding rectangles. This may result in descending more than one tree path when searching for an object in the database. The R^+-tree [21] was designed to address this issue by decomposing the objects into multiple pieces, so as to produce non-overlapping enclosing rectangles at the higher levels of the tree.

One important feature of the R-tree is that it is not unique. Its structure depends heavily on the order in which the individual objects were inserted into, and/or deleted from, the tree. Moreover, there is some freedom in deciding into which leaf node a newly added object gets inserted. The R^*-tree [5] is a variant of the R-tree that makes use of this latter feature and has a sophisticated node insertion algorithm that helps reduce the area of overlap among the enclosing rectangles.

2.2 The Window Intersection Operation

Window intersection is a central operation in spatial databases. A window can be in the form of a rectangle or a polygon. This operation serves as a building block for a number of queries. Usually, spatial features span a wide feature space. However, users are more interested in viewing or querying only portions of the feature space instead of the whole space. Extracting parts of the space to work with in subsequent operations is done by the window intersection operation. Given a window w, some examples of queries that involve window inclusion are: report all features inside w, intersect feature f with feature b only inside w, determine if feature f exists in w, etc. For example, Figure 2 shows a spatial database representing the road map of the city of Falls Church, where roads are the objects of the database. Figure 3 shows the roads in the city that are contained in window w (with corner values (100,100) and (300,300), respectively). More details about the window intersection operations and algorithms for implementing them can be found in [1].

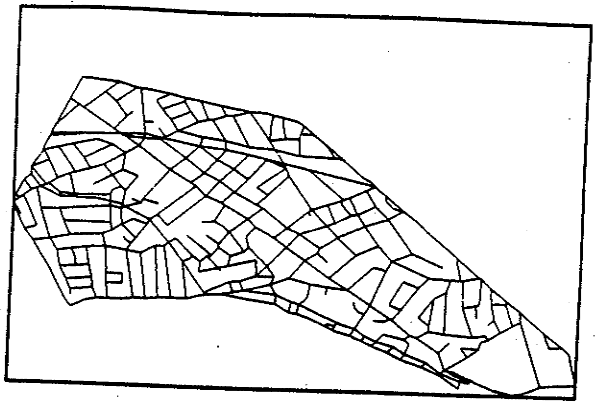


Figure 2: Part of a map sheet of the city of Falls Church, Virginia.

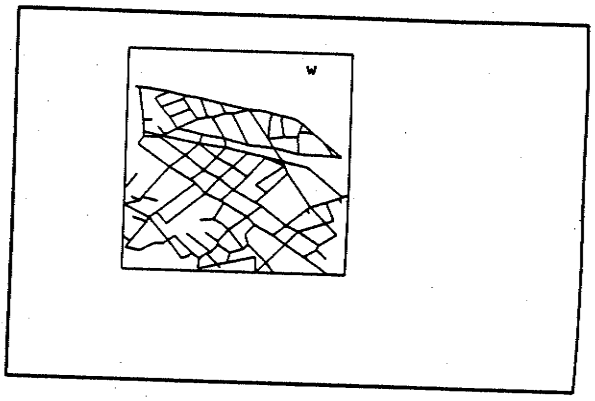
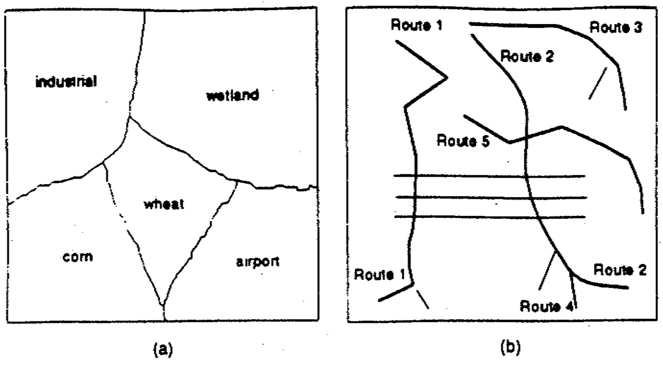


Figure 3: Roads of the city of Falls Church that are contained in window w. The origin of the space is at the lower-left corner.

2.3 The Spatial Join Operation

Spatial join is a fundamental operation for answering queries that involve spatial predicates. It is commonly used in answering queries involving spatial data sets, where it combines entities from two spatial databases into single entities whenever the combination satisfies the spatial join condition (e.g., if they overlap in space). For example, Figure 4c shows the result of joining a simple landuse database (Figure 4a) with a simple road-network database (Figure 4b). The query in this case is to find the regions of the landuse database that intersect the roads of the road-network database. For example, the join condition may be **landuse.region intersects road.coords** where **region** and **coords** store the spatial representation of the landuse regions and the road-network line segments, respectively.

As pointed out in [6], both the CPU and the disk read costs of the spatial join operation are very significant. As a result, extensive research has been conducted on alternative ways of processing the spatial join efficiently (e.g., see [3, 4, 6, 9, 18, 20]). Becker [4], and Becker, Hinrichs,



land.usage	road.name
industrial	Route 1
corn	Route 1
airport	Route 2
airport	Route 4
⋮	⋮

(c)

Figure 4: (a) A simple landuse database, (b) a simple road-network database, (c) the landuse/road pairs that intersect each other.

and Finke [3] propose an algorithm for the efficient evaluation of spatial join for databases of multidimensional point objects. Günther [9] presents a hierarchical spatial join algorithm that applies efficiently for a family of tree-based data structures, termed the generalisation tree. Brinkhoff, Kriegel, and Seeger [6] apply a similar idea in the context of the R-tree [10]. In addition, they present several techniques that reduce both the disk read and CPU costs of the spatial join significantly. Rotem [20], and later, Lu and Han [14], suggest precomputing the spatial object pairs satisfying a certain spatial relationship and storing them in *spatial join indices* in order to speed up the spatial join at query run-time. Orenstein and Manola [18] present two algorithms for spatial join where the underlying representation of spatial data is the Z-Order [19].

3 Selectivity Estimation

We use the following parameters to estimate the selectivity factors and the cost of the spatial join and window intersection operation.

- NO_{db}: the number of objects in the database.
- A_{Space}: the area of the underlying space for the entire spatial database.
- C_{db}: the data coverage which is the ratio between the sum of the areas of each object in the database and the area of the underlying space for the entire spatial database (A_{Space}).
- X_{Avg}: the average width of the minimum bounding rectangles of the objects in the database.
- Y_{Avg}: the average height of the minimum bounding rectangles of the objects in the database.
- AO_{Avg}: the average area of an object in the database. This parameter can be computed as follows:
$$AO_{Avg} = \frac{C_{db}}{NO_{db}} A_{Space}.$$
- A_w: the area of the query window
- C_w: the data coverage of the query window, i.e.,
$$C_w = \frac{A_w}{A_{Space}}.$$
- N_R: the number of nodes in the R-tree.

3.1 Selectivity of Window Intersection

We start by estimating the selectivity factors of the intersection of just one rectangular object with the spatial database. Let NO_{db} be the number of objects in the underlying spatial database. Then, the selectivity for the window intersection operation, denoted OS_w, is defined as the ratio between the number of objects in the underlying spatial database that intersect a given window and NO_{db}. Assume that we are given a window, say w, with total area A_w; of width and height X_w and Y_w, respectively; and a spatial database db of NO_{db} objects, each of type rectangle, that lie in a space of area, say A_{Space}. Given one object $o \in db$, with total area A_o, and width and height X_o and Y_o, respectively, then the probability that w intersects o is computed by (refer to Figure 5):

$$\text{Probability}(w \text{ intersects } o) = \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}}$$

Notice that if w is a point, then $X_w = Y_w = A_w = 1$.

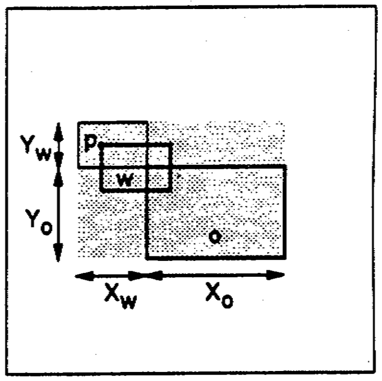


Figure 5: The rectangles w and o intersect if and only if the upper-left corner of w (point p) lies anywhere inside the shaded rectangle whose sides are $X_W + X_O$ and $Y_W + Y_O$.

The above probability formula holds for all objects in the database db. Summing for all objects in db,

$$\begin{aligned} OS_w &= \frac{1}{NO_{db}} \sum_{o \in db} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} \\ &= \frac{1}{NO_{db}} \Big(\sum_{o \in db} \frac{A_w}{A_{Space}} + \sum_{o \in db} \frac{A_o}{A_{Space}} + \\ &\quad \frac{X_w}{A_{Space}} \sum_{o \in db} Y_o + \frac{Y_w}{A_{Space}} \sum_{o \in db} X_o \Big) \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{NO_{db}}(NO_{db}C_w + C_{db} + \\
&\quad \frac{X_w NO_{db} Y_{Avg}}{A_{Space}} + \frac{Y_w NO_{db} X_{Avg}}{A_{Space}}) \\
&= C_w + \frac{C_{db}}{NO_{db}} + \frac{Y_w X_{Avg} + X_w Y_{Avg}}{A_{Space}}
\end{aligned}$$

where $C_w = \frac{A_w}{A_{Space}}$, and X_{Avg} and Y_{Avg} are the average width and height, respectively, of all the objects in the underlying spatial database.

Observe that, if $X_{Avg} = X_w$ and $Y_{Avg} = Y_w$, then

$$OS_w \quad = \quad 3C_w + \frac{C_{db}}{NO_{db}}$$

This formula is applicable if the size of the query window is of the same order as the average dimensions of the objects in the database.

3.2 Selectivity of Spatial Join

In joining two sets of objects using an intersection predicate, the selectivity denoted OS_j, is defined as the ratio between the number of intersecting pairs of objects from the two underlying spatial databases and the size of the cross product between the two databases, i.e., $NO_{db_1} \times NO_{db_2}$. One way to compute the spatial join selectivity factor OS_j is to consider one of the two input streams of the spatial join as the underlying database and to consider the other input stream as a source for query windows (i.e., the inner component of the spatial join is the underlying spatial database and the outer component is a set of query windows). This enables us to utilize the selectivity formula for the window intersection operation described in Section 3.1, and to sum over all the windows w in the outer input stream. This results in the following (as in Section 3.1, we assume that the objects in the underlying spatial database are approximated by their enclosing rectangles resulting in a spatial database of rectangles):

$$\begin{aligned}
OS_j &= \frac{1}{NO_{db_1} NO_{db_2}} \sum_{w \in db_1} \sum_{o \in db_2} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} \\
&= \frac{1}{NO_{db_1} NO_{db_2}} (\sum_{w \in db_1} \sum_{o \in db_2} \frac{A_w}{A_{Space}} + \\
&\quad \sum_{w \in db_1} \sum_{o \in db_2} \frac{A_o}{A_{Space}} + \sum_{w \in db_1} \frac{X_w}{A_{Space}} \sum_{o \in db_2} Y_o + \\
&\quad \sum_{w \in db_1} \frac{Y_w}{A_{Space}} \sum_{o \in db_2} X_o) \\
&= \frac{1}{NO_{db_1} NO_{db_2}} (NO_{db_2} \sum_{w \in db_1} \frac{A_w}{A_{Space}} + \\
&\quad \sum_{w \in db_1} C_{db_2} + NO_{db_2} Y_{Avg2} \sum_{w \in db_1} \frac{X_w}{A_{Space}} + \\
&\quad NO_{db_2} X_{Avg2} \sum_{w \in db_1} \frac{Y_w}{A_{Space}}) \\
&= \frac{1}{NO_{db_1} NO_{db_2}} (NO_{db_2} C_{db_1} + NO_{db_1} C_{db_2} + \\
&\quad NO_{db_1} NO_{db_2} \frac{X_{Avg1} Y_{Avg2}}{A_{Space}} + \\
&\quad NO_{db_1} NO_{db_2} \frac{X_{Avg2} Y_{Avg1}}{A_{Space}}) \\
&= \frac{C_{db_1}}{NO_{db_1}} + \frac{C_{db_2}}{NO_{db_2}} + \frac{X_{Avg1} Y_{Avg2} + X_{Avg2} Y_{Avg1}}{A_{Space}},
\end{aligned}$$

or equivalently,

$$= \frac{AO_{Avg1}}{A_{Space}} + \frac{AO_{Avg2}}{A_{Space}} + \frac{X_{Avg1} Y_{Avg2} + X_{Avg2} Y_{Avg1}}{A_{Space}}$$

Notice that the roles of db_1 and db_2 are symmetric in the above formula. Also notice that the last formula indicates that the selectivity factor for the spatial join depends only on the average area (or the extent) of the objects in the underlying spatial database.

3.3 Self Join

In many circumstances we may need to join a stream with itself (which we term a *self join*). For example, finding the pairs of intersecting rectangles in a given database requires a self join operation. Since in addition to intersecting some other rectangles in the stream, each rectangle intersects itself, the formulas given in Section 3.2 for estimating the selectivity factors of the spatial join will not be accurate in the case of the self join. We need to compensate for the fact that each rectangle intersects itself. This results in some additional output pairs of the spatial join that are equal to the total number of objects in the database. Therefore, we add a corrective factor equal to the number of objects in the stream to the formulas given in Section 3.2 (the same is also true for the blocks in the stream). This results in the following formula for the selectivity of the self join, OS_{sj}. Notice that in order to include the corrective factor in the formula of the selectivity factor, we divide the corrective factor by the term $NO_{db} \times NO_{db}$ which is the size of the output of the spatial join in the worst case.

$$\begin{aligned}
OS_{sj} &= OS_j + \frac{1}{NO_{db}} \\
&= 2\frac{C_{db}}{NO_{db}} + 2\frac{X_{Avg} Y_{Avg}}{A_{Space}} + \frac{1}{NO_{db}} \\
&= \frac{2C_{db} + 1}{NO_{db}} + 2\frac{X_{Avg} Y_{Avg}}{A_{Space}}, \text{ or equivalently,} \\
&= 2\frac{AO_{Avg}}{A_{Space}} + 2\frac{X_{Avg} Y_{Avg}}{A_{Space}} + \frac{1}{NO_{db}} \\
&= 2\frac{AO_{Avg} + X_{Avg} Y_{Avg}}{A_{Space}} + \frac{1}{NO_{db}}
\end{aligned}$$

4 Execution Cost

In the following, we present an estimate of the cost of each of the two operations in the absence and presence of an R-tree index. The total cost of an operation is a combination of both the CPU and I/O costs (C_{cpu} and C_{io}, respectively).

We use a weighting factor ω between the CPU and the disk I/O costs. In other words, the total cost for an operation is estimated by:

$$C_{operation\_total} \quad = \quad C_{operation\_io} + \omega C_{operation\_cpu}$$

4.1 The Execution Cost of Window Intersection

In the absence of an index, to answer a query that involves a window intersection predicate, we need to scan the input relation and test whether each of the objects in the relation intersects the query window or not. The CPU cost of intersecting two rectangles involves four comparisons. As a result, in the absence of an index, the total cost of the window intersection operation can be estimated by:

$$\begin{aligned} C_{win\_intersect} &= C_{io} + \omega C_{cpu} \\ &= N_P + 4\omega NO_{db} \end{aligned}$$

where N_P is the number of disk pages of the entire spatial database.

In the presence of an R-Tree index, according to Faloutsos and Kamel [11], the average cost of disk reads, $C_{indexed\_win\_io}$, is given by:

$$\begin{aligned} C_{indexed\_win\_io} &= total\_area + N_R\ query\_area + Y_w L_x + \\ &\quad X_w L_y \end{aligned}$$

where *total_area*, L_x, and L_y, are the sum of the areas, widths, and heights of all bounding boxes of the nodes of the R-Tree, respectively, *query_area* is the area of the query window, and N_R is the number of nodes in the R-tree, and hence $C_{indexed\_win\_io}$ can be computed as follows:

$$\begin{aligned} C_{indexed\_win\_io} &= C_{db} A_{Space} + N_R A_w + \\ &\quad NO_{db}(Y_w X_{Avg} + X_w Y_{Avg}) \end{aligned}$$

We also include the CPU cost $C_{indexed\_win\_cpu}$ to the above formula, where $C_{indexed\_win\_cpu}$ is derived in exactly the same way as the $C_{indexed\_win\_io}$, but with a different cost factor. Since an intersection of two rectangles takes four comparisons, therefore

$$C_{indexed\_win\_cpu} = 4\omega C_{indexed\_win\_io}$$

where ω is the weighting factor between the disk read cost and the CPU cost. Therefore, the total cost $C_{indexed\_win\_total}$ can be estimated as:

$$\begin{aligned} C_{indexed\_win\_total} &= (1+4\omega) C_{indexed\_win\_io} \\ &= (1+4\omega)(C_{db} A_{Space} + N_R A_w + \\ &\quad NO_{db}(Y_w X_{Avg} + X_w Y_{Avg})) \end{aligned}$$

4.2 The Execution Cost of Spatial Join

There are several new and efficient algorithms for performing spatial join using R-trees, e.g., see [6, 9]. In this section, we focus on the simple nested loop join in the presence of an R-tree index as the inner of the join.

One way to estimate the execution cost of spatial join C_{sp_join} is to consider one of the two input streams of the spatial join as the underlying database and to consider the other input stream as a source for query windows (i.e., the inner stream of the spatial join, db_2, is the underlying spatial database and the outer stream, db_1, is a set of query windows). This enables us to utilise the cost formula for the window intersection operation described in Section 4.1, and to sum over all the windows w in the outer input stream. This results in the following formula for C_{sp_join}:

$$\begin{aligned} C_{sp\_join\_io} &= \sum_{w \in db_1} C_{indexed\_win\_io} \\ &= \sum_{w \in db_1} (C_{db_2} A_{Space_2} + N_{R_2} A_w + \\ &\quad NO_{db_2}(Y_w X_{Avg_1} + X_w Y_{Avg_1})) \\ &= NO_{db_1} C_{db_2} A_{Space_2} + N_{R_2} C_{db_1} A_{Space_1} + \\ &\quad NO_{db_1} NO_{db_2}(X_{Avg_1} Y_{Avg_2} + X_{Avg_2} Y_{Avg_1}) \\ C_{sp\_join\_cpu} &= 4\omega C_{sp\_join\_io} \end{aligned}$$

Therefore,

$$\begin{aligned} C_{sp\_join\_total} &= (1+4\omega) C_{sp\_join\_io} \\ &= (1+4\omega)(NO_{db_1} C_{db_2} A_{Space_2} + \\ &\quad N_{R_2} C_{db_1} A_{Space_1} + \\ &\quad NO_{db_1} NO_{db_2}(X_{Avg_1} Y_{Avg_2} + X_{Avg_2} Y_{Avg_1})) \end{aligned}$$

Notice that the formula for execution cost of spatial join is not symmetric with respect to db_1 and db_2. Therefore, if there are two R-tree indexes, one for each of the two databases, then two formulas are computed in order to decide on which inner/outer assignment of the two streams will result is a cheaper execution cost of the spatial join.

5 Experimental Results

The purpose of our experiments is to verify the significance of the derived formulas for estimating the selectivity factors as well as the cost of execution under varying conditions. The formulas for the execution cost are already those of Kamel and Faloutsos ([11]), where they conducted many experiments to verify the formulas. They report that the cost formulas match the experimental results extremely well. Therefore, in this paper, we only verify the formulas for estimating selectivity factors.

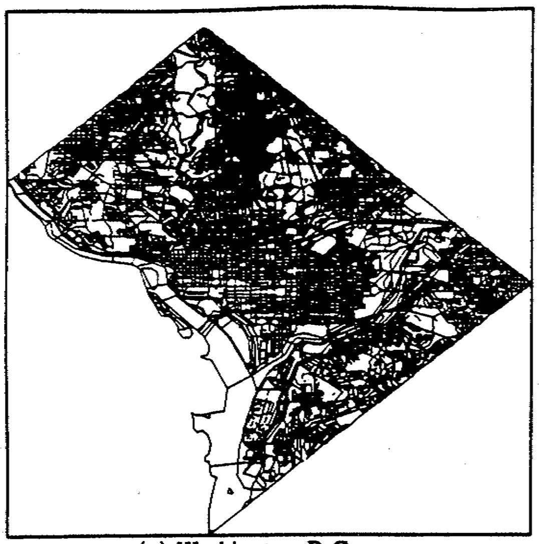
The experiments for verifying selectivity factors were conducted using real as well as synthetic data sets. The real data sets consisted of the road networks in the data of the U.S. Bureau of the Census Tiger/Line file [17] for representing the roads and other geographic features in the U.S. The synthetic data sets are collections of rectangles generated at random.

Data Coverage	Area of Rectangle	Number of Rectangles
0.01	64	40
0.10	64	409
0.20	64	819
1.00	64	4096
1.50	64	6144
2.00	64	8192
3.00	64	12288

Table 1: Parameter settings for input data sets with fixed object size and varing data coverage.

Data Coverage	Area of Rectangle	Number of Rectangles
1.00	128	2048
1.00	512	512
1.00	1024	256
1.00	2048	128
1.00	16384	16

Table 2: Parameter settings for input data sets with fixed data coverage and varying object area.


(a) Washington D.C.

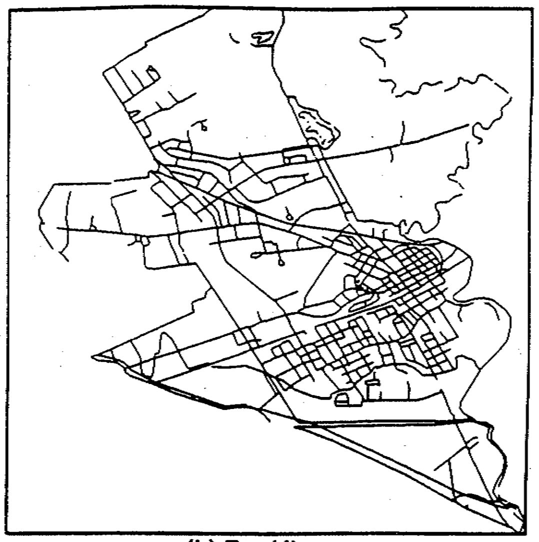

(b) Franklin

Figure 6: A Sample of the Tiger/Line spatial databases used in the experiments.

5.1 Experiments with Real Data

We used five Tiger/Line databases (two of the maps are given in Figures 6a and 6b). For each line segment in the Tiger/Line database, we constructed the line's minimum bounding rectangle. These data sets help verify our estimates for arbitrary distributions of objects in space.

Table 3 gives a characterisation of the real data sets (the Tiger/Line) files using some of the parameters suggested and discussed in Section 3. These include the data coverage of all the objects, the average area of all the bounding boxes that include each line segment in the database, and the number of objects in the entire database. The size of the underlying space for all the Tiger/Line data sets is normalised to 512 × 512.

Map Name	Data Coverage	Avg. Area of Bound. Box	Number of Line-segs.
Falls Church (FC)	0.43	192.10	587
Bedford (BF)	0.34	54.17	1644
Williamsburg (WB)	0.19	23.62	2112
Franklin (FR)	0.35	54.46	1685
Washington D.C. (DC)	0.56	7.90	18517

Table 3: Parameter setting for the Tiger data files. The size of all the maps is normalized to 512 × 512.

In order to conduct our experiments, we spatially join each pair of the Tiger/Line files together. Notice that although the Tiger/Line data files that we use refer to non-overlapping geographical regions, we normalise the coordinate values of each file so that its upper-left corner has the coordinate values (0,0). This results in overlapping data sets and allows us to perform spatial join using data sets derived from real sources.

Table 4 gives the relative error in the estimated value of the selectivity factor of the spatial join over the measured value. In the table, a negative percentage indicates that the measured value was smaller than the estimated value while a positive percentage indicates that the estimated value was smaller. The total number of data set pairs that we spatially joined together was 15. Figure 7 summarises the above tables by giving the percentage of the number of spatial joins whose estimated selectivities lie within a certain relative error from the actual value.

In the case of estimating selectivity factors, for the 15 spatial join operations that we conducted, 14 of them (i.e., over 93% of the joins performed) have their estimated value of the selectivity factor lie within 30% of the actual measured value, and 13 of them (i.e., over 86% of the joins) have their estimated value lie within 25% of the actual measured value.

These estimates are certainly good enough to be useful for query optimisation purposes. The experiments also show that our formulas for estimating the selectivity factor for the spatial join operation perform quite well for real data sets, given the non-uniformity in the distribution of the objects in the underlying space of these data sets.

	FC	BF	WB	FR	DC
FC	0.24	-0.03	0.15	0.13	0.25
BF		0.13	-0.10	-0.04	0.10
WB			0.29	0.06	-0.22
FR				0.22	0.14
DC					0.55

Table 4: The relative error in the value of the estimated selectivity factor of the spatial join over the measured value.

5.2 Experiments with Synthetic Data

Tables 1 and 2 describe the synthetic data sets that we used in our experiments. The tables use some of the parameters given in Section 3. These include the data coverage of all the objects, the average area, in pixels, of all the objects in the database, and the number of objects in the entire database. The size of the underlying space for all the synthetic data sets is normalised to 512 × 512.

We conducted the following experiment: for each entry in Tables 1 and 2, we generate 10 sets of random rectangles

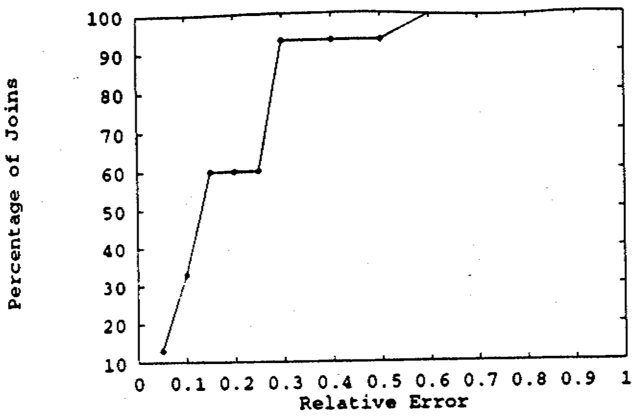


Figure 7: The number of spatial joins whose estimates were within the relative error given in the x-axis.

satisfying the same parameter setting PS. We call the 10 sets a master data set, and denote it by MDS_{PS}. To measure the selectivity factor for spatial join for a pair of parameter settings PS_1 and PS_2, we ran the following experiment: each set of rectangles in MDS_{PS_1} is joined with each set of rectangles in MDS_{PS_2} using spatial join. The number of output pairs is measured, summed and averaged (by dividing it by 100, which is the total number of times the spatial join was executed for this experiment). For our experiments, we fixed the area of the underlying space to be always 512×512. We verified our estimates with the measured selectivity factors while varying the data coverage of the input streams as well as the average size of the rectangles in the underlying spatial database.

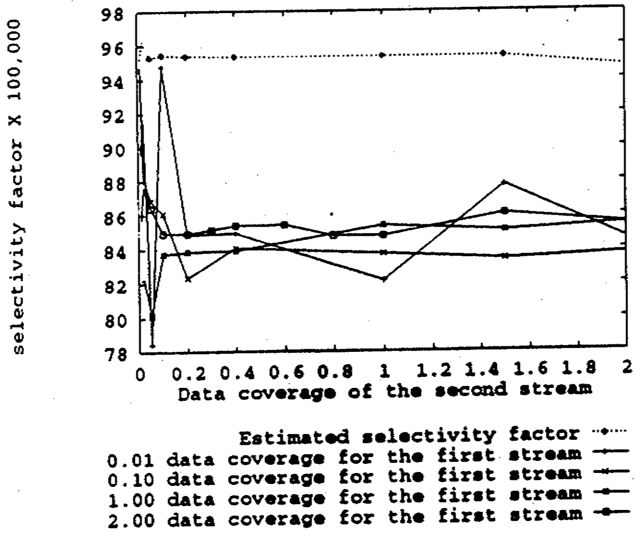


Figure 8: A comparison of the measured vs. the estimated selectivity for the spatial join operation as a function of the data coverage of the second stream.

Figure 8 compares the estimated value of the selectivity factor for the spatial join against the measured value for different data coverages (we use data sets from Table 1 for both input streams). The data coverage of the first input stream was fixed at 0.01, 0.1, 1.0, and 2.0, respectively, while the data coverage of the second input stream varied from 0.01–2.0. In the figure, the estimated value for the selectivity factor of the spatial join is constant (approximately, 95×10^{-5}). This is because the estimates for the selectivity factor that we developed in Section 3.2 do not depend on the the data coverage. They depend only on the average sizes of the objects in each stream. The figure also shows the measured values of the selectivity factor that resulted from the experiments. It is found that they are relatively constant and are within 20% of the estimated value.

Figure 9 performs the same comparison while varying the average areas of the objects in the second input stream (we use data sets from Tables 2 for both input streams). The average area of each object in the first input stream in Figure 9 is 1024. Since the total area of the space is 512×512, 1024 accounts for about 0.4% of the whole space area, respectively. The relative error of the estimated values for the selectivity factor of the spatial join is within 10% of the measured values.

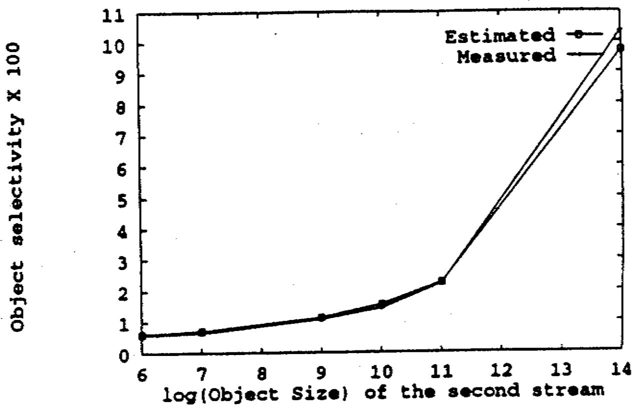


Figure 9: A comparison of the measured vs. the estimated selectivity for the spatial join operation as a function of different average object areas. The data coverage of both input streams of the join is 100%. The average area of the objects in the first stream is 1024. The x-axis corresponds to the average area of the objects of the second stream.

From the figures we can see that our estimates form an upper bound on the measured values. The maximum error between the two values is less than 25%, although in most cases it is much less.

6 Concluding Remarks

The formulas in Sections 3 and 4 make use of several parameters that characterise the underlying spatial database and their R-tree index. These parameters are simple to compute and easy to maintain as the spatial database evolves. However, it would be interesting to study ways of estimating the selectivity and cost of the spatial operations discussed in this paper by statistical methods based on sampling. We also plan to consider other important spatial operations, e.g., nearest-neighbor and window inclusion, spatial databases that involve non-rectangular spatial objects, e.g., polygons, and other spatial access methods.

7 Acknowledgements

The first author conducted most of this research while at The University of Maryland, College Park. The support of the National Science Foundation under Grant IRI 9216970 is gratefully acknowledged.

References

[1] W. G. Aref and H. Samet. Efficient processing of window queries in the pyramid data structure. In *Proceedings of the 9th. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 265–272, Nashville, TN, April 1990.

[2] W. G. Aref and H. Samet. Query size estimation of spatial join. Technical Report 79, Matsushita Information Technology Laboratory, Princeton, NY, 08540, December 1993.

[3] L. Becker, K. Hinrichs, and U. Finke. A new algorithm for computing joins with grid files. In *Proceedings of the 9th International Conference on Data Engineering*, pages 190–197, Vienna, Austria, April 1993.

[4] L. A. Becker. *A New Algorithm and a Cost Model for Join Processing with Grid Files*. PhD thesis, University of Siegen, July 1992.

[5] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, volume 19, pages 322–331, Atlantic City, NJ, June 1990.

[6] T. Brinkhoff, H. P. Kriegel, and B. Seeger. Efficient processing of spatial joins using R-trees. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 237–246, Washington, DC, May 1993.

[7] C. Faloutsos, T. Sellis, and N. Roussopoulos. Analysis of object oriented spatial access methods. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pages 426–439, San Francisco, May 1987.

[8] O. Günther. The design of the cell tree: an object-oriented index structure for geometric databases. In *Proceedings of the Fifth IEEE International Conference on Data Engineering*, pages 598–605, Los Angeles, February 1989.

[9] O. Günther. Efficient computation of spatial joins. In *Proceedings of the 9th International Conference on Data Engineering*, pages 50–59, Vienna, Austria, April 1993.

[10] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, June 1984.

[11] Ibrahim Kamel and Christos Faloutsos. On packing R-trees. In *Second Int. Conf. on Information and Knowledge Management (CIKM)*, pages 490–499, Washington, DC, November 1993.

[12] A Klinger. Patterns and search statistics. In J. S. Rustagi, editor, *Optimizing Methods in Statistics*, pages 303–337. Academic Press, New York, 1971.

[13] H. P. Kriegel, P. Heep, S. Heep, M. Schiwietz, and R. Schneider. An access method based query processor for spatial database systems. In G. Gambosi, M. Scholl, and H.-W. Six, editors, *Geographic Database Management Systems. Workshop Proceedings, Capri, Italy, May 1991*, pages 194–211, Berlin, 1992. Springer-Verlag.

[14] W. Lu and J. Han. A new algorithm for computing joins with grid files. In *Proceedings of the 8th International Conference on Data Engineering*, pages 284–292, Tempe, Arizona, February 1992.

[15] Abha Moitra. Spatio-temporal data management using R-Trees. In Niki Pissinou, editor, *Proceedings of the ACM Workshop on Advances in Geographic Infomation Systems*, pages 28–33, Arlington, Virginia, November 1993.

[16] H. Nievergelt, H. Hinterberger, and K. C. Sevcik. The grid file: an adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, 9(1):38–71, March 1984.

[17] Bureau of the Census: 1990 technical documentation. Tiger/line precensus files. Technical report, US Bureau of Census, Washington, DC, 1989.

[18] J. A. Orenstein and F. A. Manola. PROBE spatial data modeling and query processing in an image database application. *IEEE Transactions on Software Engineering*, 14(5):611–629, May 1988.

[19] J. A. Orenstein and T.H. Merrett. A class of data structures for associative searching. In *Proceedings of the 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS)*, pages 181–190, Waterloo, Canada, April 1984.

[20] Doron Rotem. Spatial join indices. In *Proceedings of the 5th International Conference on Data Engineering*, pages 500–509, Kobe, Japan, April 1991.

[21] T. Sellis, N. Roussopoulos, and C. Faloutsos. The r^+-tree: a dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Databases (VLDB)*, pages 507–518, Brighton, England, September 1987.