Estimating Selectivity Factors of Spatial Operations[1]

Walid G. Aref
Hanan Samet
Computer Science Department and
Center for Automation Research and
Institute for Advanced Computer Studies
The University of Maryland
College Park, Maryland 20742

Abstract

Selectivity factors for some widely used spatial operations are defined in the context of spatial data structures that partition spatial objects into more than one piece. In relational databases, a selectivity factor helps estimate the size of the output relation (i.e., a one dimensional selectivity factor). On the other hand, spatial data sets and spatial operations require selectivity factors of higher dimensionality (e.g., two-dimensional selectivity factors). Two types of selectivity factors are defined for spatial operations: the object-selectivity factor and block- (or piece-) selectivity factor, which estimate the number of objects and object-pieces, respectively, that result from a spatial operation. Both factors are useful for cost-based spatial query optimization purposes. The spatial operations considered are window intersection and spatial join. Formulas for estimating both types of selectivity factors are derived for each of these spatial operations. The selectivity formulas are functions of simple parameters that characterize the underlying data sets as well as characterize the issued user query. These parameters are computed by preprocessing the data sets. The validity of the formulas is verified experimentally using synthesized as well as real spatial data sets.

Keywords: spatial databases, database systems, query optimization, selectivity factor, spatial join, window query

August 14, 1993.

[1]The support of the National Science Foundation under Grant IRI-9017393 is gratefully acknowledged.

1 Introduction

As a first step towards query processing and optimization of spatial queries, we need to estimate the cost as well as the cardinality and size of the output of spatial operations. In this paper, we focus on ways of estimating only the cardinality and size. Traditionally, in relational databases, a selectivity factor serves this purpose, mainly it is useful for estimating the cardinality and size of the output relation (i.e., a one dimensional selectivity factor). Here, we wish to address the problem of defining and estimating selectivity factors for some commonly used spatial operations in an integrated database environment.

Spatial data sets and spatial operations require selectivity factors of higher dimensionality (e.g., two-dimensional selectivity factors). One way to go around the dimensionality aspect of spatial data is by using data representation methods that map an n-dimensional spatial object into a sequence of simple object-pieces, each of which can be described by a one-dimensional description and stored separately inside the data structure. In dealing with data structures that decompose an object into more than one piece, we propose in this paper two complementary ways of defining what a selectivity factor is: *Object-level* selectivity factor (*OS*) and *piece-level* selectivity factor (*PS*), which estimate the number of objects and object-pieces, respectively, that result from a spatial operation. The object-level selectivity factor is useful in determining the cardinality of the objects resulting from a given operation, whereas the piece-level selectivity factor is useful in determining the size of the output of the given operation (since the number of output pieces determine the size of the output). In this paper, we present estimates of both selectivity factors for some widely used spatial operations.

An important question that we address in this paper and that arises when adopting a cost-based approach for optimizing queries in spatial databases is the following: what are appropriate statistics that can be gathered from the underlying spatial database that can be useful for predicting the cost of a given typical set of spatial operations? It is important to note that a useful set of statistics should reflect, not only a characterization of the underlying database, but also a characterization of the underlying representation of spatial d.ta and the underlying access methods. For example, regarding datatypes, the statistics gathered to optimize queries for a spatial database of points may be different for those for a spatial database of lines, or polygons. With respect to data representation, statistics useful for methods that represent spatial objects by one entity (e.g., a two-dimensional rectangle represented by a point in four-dimensional space) may be different from those that are useful for methods that represent spatial objects by more than one entity (e.g., a polygon represented by partitioning it into more than one piece, each of which is of simpler shape). Finally, with respect to access methods and data organizations, statistics that assume a grid-like data structure (e.g., the Grid File [3]) may be different from those that assume a tree-like structure (e.g. the R-tree [2]).

An important issue that is closely related to spatial database modeling for query processing and optimization purposes is that of the distribution of objects in space. Some

knowledge of the data distribution of the underlying spatial database is mandatory in order to guide in selecting sound statistics for query processing and optimization purposes as well as it affects the types of the parameters selected. In addition to handling uniformly distributed data sets, we also approximate data sets that have a non-uniform distribution by using a piece-wise uniform distribution. We study the effectiveness of our approach for modeling real as well as synthetic data sets for query processing and optimization purposes.

The rest of the paper is organized as follows. Section 2 presents our proposed characterization of the underlying spatial data set. Section 3 presents formulas for estimating selectivity factors of the spatial operations: window intersection and spatial join. Section 4 presents our experimental setting for estimating selectivity factors as well as the experimental results using both real and synthetic data sets, while contrasting the measured selectivity factors with the developed formulas. Section 5 contains concluding remarks.

2 Parametric Characterization of the Underlying Spatial Database

In order to obtain meaningful estimates of selectivity factors, i.e., ones that are close to reality, we would like to know more about the underlying spatial database. In particular, we need to know about the characteristics and parameterization of the spatial objects in the database as well as the way the underlying spatial objects are represented and organized inside the database.

There are several ways of knowing the characteristics of the underlying data set, e.g., by gathering statistics. However, until now it is not clear what statistics are useful for spatial query optimization purposes. So, it is premature to follow the statistical approach since there is no consensus yet about the useful parameters. As a result, in this paper, we assume a simpler approach. Mainly, we assume that we will be allowed to preprocess the underlying spatial database by scanning it in one pass and to gather full information about the parameters of interest to us.

In selecting parameters for characterizing spatial database, our goal is twofold. The first, and obvious, goal is that these parameters have to be useful in providing estimates of the selectivity factor and the cost of spatial operations, and hence in helping as a guide for query optimization. Second, these parameters should be easy to maintain if the underlying spatial database is updated. Sample parameters that can be gathered in one preprocessing pass of the underlying database are the number of objects in the database, the total area covered by the objects in the database, the average area of the objects, the number of blocks in the database, etc.

Regarding the representation and organization of spatial data, as we mentioned in Section 1, we focus on data structures that divide spatial objects into simple pieces (e.g., rectangles). In order to have a concrete model of the spatial database, we focus on one type of data structure that partitions a spatial object using regular decomposition of space (i.e., the Z-Order [4, 5], or the linear quadtree [1, 6]). In this data structure, a

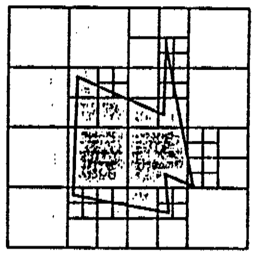


Figure 1: An example quadtree decomposition of a polygon.

spatial object is represented by a set of blocks that collectively approximate and cover the object's internal region. For example, Figure 1 gives the quadtree decomposition of a simple polygon into the quadtree blocks (termed Morton blocks in the case of the linear quadtree, or Z-elements in the case of the Z-Order) that are internal to the polygon. For the rest of this paper, we will assume that all the objects in the database are decomposed in a similar manner.

Assuming the above data structure, we precompute the following parameters during our preprocessing phase:

- NO_{db}: the number of objects in the database,
- NB_{db}: the number of object-pieces in the database,
- C_{db}: the total area covered by the objects in the database (also termed the *area coverage*),
- X_{Avg}, X_{Min} and X_{Max}: the average, minimum, and maximum width, respectively, of the minimum bounding rectangles of the objects in the database,
- Y_{Avg}, Y_{Min} and Y_{Max}: the average, minimum, and maximum height, respectively, of the minimum bounding rectangles of the objects in the database,
- NB_{Avg}: the average number of blocks per object in the database.
- A_{Space}: the area of the underlying space for the entire spatial database.

We will use several of these parameters in the following section to estimate the cost of some common spatial operations. The rest of the parameters are useful for other operations, e.g., window containment.

Notice that the above parameters can be computed differently depending on the underlying spatial data distribution. For example, if the data is uniformly distributed,

then the parameters can be computed over the entire space. On the other hand, in the case when data is non-uniformly distributed, the database is divided into a coarse uniform grid. When the database is preprocessed, the parameters are evaluated for each grid cell separately. This results in a two-dimensional array for each computed parameter, where the value stored in each entry of the array corresponds to the value of the parameter at the corresponding grid cell of the underlying database. For a given spatial operation that queries (or overlaps) a certain grid cell of the database, the parameters of this cell are the ones used by the optimizer to estimate the cost or selectivity factor of the given operation. We do not address this topic any further in the paper.

3 Estimates of Selectivity Factors

In this section, we assume that objects in the spatial database are represented by their internal regions using the Z-order [4, 5] or the linear quadtree [1, 6] data structures. Although these data structures are capable of representing objects of a wide variety of data types, in this section we limit our discussion to spatial objects that are of type rectangle. Formulas for estimating the selectivity factors of other data types can be derived analogously[2]. We compute the object-level as well as block-level selectivity factors for some commonly used spatial operations, namely, window intersection and spatial join. Notice that, even if we assume that objects are distributed uniformly in space (e.g., in the uniform model), it is not necessarily true that the object pieces that comprise the internal region of the objects are distributed uniformly in space. For example, if a database of rectangles is distributed uniformly in space, then the Z-elements or the Morton blocks that approximate each rectangle will be clustered around that rectangle, and hence the set of blocks that correspond to the whole database is not uniformly distributed.

3.1 The Window Intersection Operation

The object-level selectivity for the window intersection operation, denoted OS_w, is defined as the number of objects that intersect a given window. Assume that we are given a window, say w, with total area, say A_w, and width and height, say X_w and Y_w, respectively, and a spatial database, say 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, say A_o, and width and height, say X_o and Y_o, respectively, then the probability that w intersects o is computed by (refer to Figure 2):

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

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

[2]In fact, for the purpose of query optimization, other objects can be approximated using their minimum bounding rectangles. Then the formulas in this section can be applied for spatial databases of other data types, e.g., the polygon data type.

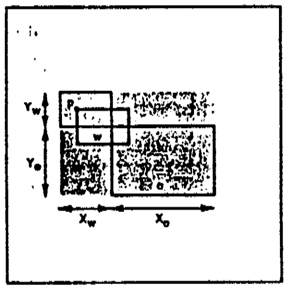


Figure 2: The rectangles w and o intersect if and only if the upper-left corner of w (point p) lies anywhere inside the shaded rectangle whose sides are $X_w + X_o$ and $Y_w + Y_o$.

holds for all objects in db. Summing for all objects in the database,

$$\begin{aligned} OS_w &= \frac{1}{NO_{db}} \sum_{o \in db} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} \\ &= \frac{C_w + C_{db}}{NO_{db}} + \frac{Y_w X_{Avg} + X_w Y_{Avg}}{A_{Space}} \end{aligned}$$

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

In order to compute the block selectivity of the window intersection operation, we follow a slightly different approach. We cannot just apply the above formula by replacing the rectangular objects by the object-blocks and sum over all the blocks in the database. The reason is that the set of all pieces composing the objects is not uniformly distributed over the entire space. The objects themselves are, but the object-blocks are clustered around th· ·niformly distributed objects. Thus we cannot generalize the above formula for object pieces as well. Instead, we proceed as follows: First, we compute the average area of intersection, say A_{ow} between the query window w and an arbitrary object, say o, in the database. Then we estimate the number of the object-blocks in A_{ow} and sum the result over all the objects in the database. From Figure 2, the upper-left corner of w is equally likely to be at any location inside the rectangle $[X_w, X_w + X_o] \times [Y_w, Y_w + Y_o]$, i.e., with probability $\frac{1}{(X_w+X_o)(Y_w+Y_o)}$. To compute the average area of intersection, we integrate over all possible values of x and y inside the rectangle $[X_w, X_w + X_o] \times [Y_w, Y_w + Y_o]$ and compute the area of intersection in each case and multiply it by its probability of occurrence, we get:

$$A_{ow} = \frac{A_w A_o}{(X_w + X_o)(Y_w + Y_o)}$$

From A_{ow}, we can estimate the number of intersecting object-blocks, say NB_{ow}, as:

$$NB_{ow} = \frac{A_{ow}}{A_o} NB_o$$

Summing over all the objects o in the database and multiplying by the probability of intersection between each o and w, we can estimate the block selectivity of the window intersection operation as follows:

$$\begin{aligned} BS_w &= \frac{1}{NB_{db}} \sum_{o \in db} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} NB_{ow} \\ &= \frac{A_w}{A_{Space}} \end{aligned}$$

3.2 The Spatial Join Operation

In joining two sets of objects using an intersection predicate, the object-level selectivity denoted OS_j, is defined as the number of intersecting pairs of objects from the two sets. 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):

$$OS_j = \frac{1}{NO_{db_1} NO_{db_2}} \sum_{w \in db_1} \sum_{o \in db_2} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}}$$

By summing and simplifying, we get:

$$OS_j = \frac{C_{db_1}}{NO_{db_1}} + \frac{C_{db_2}}{NO_{db_2}} + \frac{X_{Avg1} Y_{Avg2} + X_{Avg2} Y_{Avg1}}{A_{Space}}$$

Notice that the roles of db_1 and db_2 are symmetric in the above formula.

We can compute the block selectivity of spatial join BS_j as follows. Assume that objects w and o belong to db_1 and db_2, respectively. We estimate the average area of intersection (A_{ow}) between these two objects in the same way as in Section 3.1. Now, we compute the number of blocks of each object that overlap with A_{ow}. Let these be denoted by NB_{ow} and NB_{wo}. The maximum number of pairs of intersecting blocks is less than $NB_{ow} + NB_{wo}$ (e.g., Figure 3d where it is 4), and in the best case, it is greater than

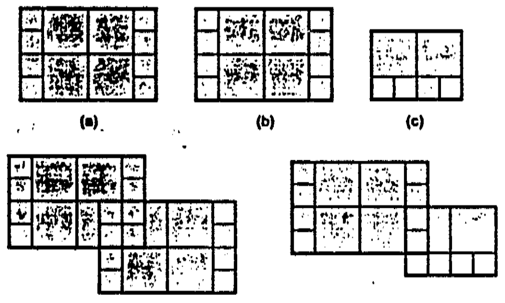


Figure 3: (a)-(c) A decomposition of three rectangles into quadtree blocks that cover the internal of the rectangles. (d) Although the number of blocks inside the intersection region is 3 blocks in each rectangle, there are 4 intersecting pairs of blocks. (e) The best case occurs when all the blocks in one rectangle intersect just one block in the other rectangle.

or equal to $\max(NB_{ow}, NB_{wo})$ (e.g., Figure 3e where it is 2)[3]. We define $BS_{j\text{-}max}$ and $BS_{j\text{-}total}$ as the block selectivity for spatial join in the best and worst cases, respectively. They can be computed by summing over all the objects o and w in the two input streams and in each case multiplying the number of estimated intersecting pairs by the probability of intersection between o and w. This results in the following estimates:

$$BS_{j\text{-}total} = \frac{1}{NB_{ds_1} NB_{ds_2}} \sum_{w \in ds_1} \sum_{o \in ds_2} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} (NB_{ow} + NB_{wo})$$

$$BS_{j\text{-}max} = \frac{1}{NB_{ds_1} NB_{ds_2}} \sum_{w \in ds_1} \sum_{o \in ds_2} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} \max(NB_{ow}, NB_{wo}) .$$

where NB_{ow} and NB_{wo} are defined and computed as in Section 3.1 (The formula for NB_{wo} is the same as the one for NB_{ow} once the roles of o and w have been interchanged). Notice that the actual value of the block selectivity of spatial join, BS_j lies somewhere between $BS_{j\text{-}max}$ and $BS_{j\text{-}total}$, i.e.,

$$BS_{j\text{-}max} \leq BS_j \leq BS_{j\text{-}total}$$

By substituting the estimated values of NB_{ow} and NB_{wo} in the formula for computing

[3]Notice that the worst case of overlaying n and m blocks is nm. However, because of the restrictiveness of the quadtree recursive decomposition of space, blocks cannot intersect each other arbitrarily; they can either coincide in space or one block can contain the other. As a result, the worst case is reduced to $n + m - 1$.

$BS_{j\text{-}total}$, summing, and simplifying, we get:

$$BS_{j\text{-}total} = \frac{C_{ds_1}}{NB_{ds_1}} + \frac{C_{ds_2}}{NB_{ds_2}}$$

It is difficult to get a closed form expression for $BS_{j\text{-}max}$ since it depends on the sum of the maximum of pairs of blocks. However, since $BS_{j\text{-}max}$ serves as a lower bound for BS_j, we can replace the formula for $BS_{j\text{-}max}$ by a further lower value $\bar{BS}_{j\text{-}max}$ such that

$$\bar{BS}_{j\text{-}max} \leq BS_{j\text{-}max} \leq BS_j \leq BS_{j\text{-}total}$$

$\bar{BS}_{j\text{-}max}$ is defined as follows:

$$\bar{BS}_{j\text{-}max} = \max(\bar{BS}_{j\text{-}max1}, \bar{BS}_{j\text{-}max1}), \text{ where}$$

$$\bar{BS}_{j\text{-}max1} = \frac{1}{NB_{ds_1} NB_{ds_2}} \sum_{w \in ds_1} \sum_{o \in ds_2} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} NB_{ow}, \text{ and}$$

$$\bar{BS}_{j\text{-}max2} = \frac{1}{NB_{ds_1} NB_{ds_2}} \sum_{w \in ds_1} \sum_{o \in ds_2} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} NB_{wo}.$$

Notice that $\bar{BS}_{j\text{-}max} \leq BS_{j\text{-max}}$ since

$$\max(\sum_{a \in A} \sum_{b \in B} a, \sum_{a \in A} \sum_{b \in B} b) \leq \sum_{a \in A} \sum_{b \in B} \max(a, b)$$

By substituting the formulas for NB_{ow} and NB_{wo}, we get:

$$\bar{BS}_{j\text{-}max1} = \frac{C_{ds_1}}{NB_{ds_1}}$$

$$\bar{BS}_{j\text{-}max2} = \frac{C_{ds_2}}{NB_{ds_2}}.$$

Therefore,

$$\bar{BS}_{j\text{-}max} = \max(\frac{C_{ds_1}}{NB_{ds_1}}, \frac{C_{ds_2}}{NB_{ds_2}})$$

The number of output pairs of blocks can be estimated by multiplying $\bar{BS}_{j\text{-}max}$ or $BS_{j\text{-}total}$ by the Cartesian product $NB_{ds_1} \times NB_{ds_2}$ (according to the definition of the join selectivity factor, where the Cartesian product of the two input data sets is considered to be the maximum size of the input).

4 Experimental Results

The purpose of our experiments is to verify the significance of the derived formulas for estimating selectivity factors under varying conditions. Due of the limited space, we only present the results of testing the spatial join operation. Experiments are conducted using real as well as synthetic data sets. The real data sets were built from Tiger/Line files which describe road networks of cities, counties, and states of the United States of America. The synthetic data sets are collections of rectangles that are randomly generated. Each rectangle is then decomposed into its corresponding constituent pieces (in our case, the pieces are the Morton blocks inside each rectangle). Notice that since the generated rectangles can possibly overlap, so will the Morton blocks. Recall that in the context of rectangular data sets and the spatial join operation, the object selectivity is an indicator of the number of pairs of intersecting rectangles while the block selectivity is an indicator of the number of pairs of intersecting blocks.

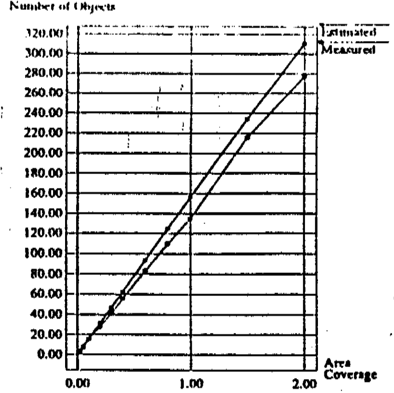
4.1 Experiments with Synthetic Data

The syntheticd data sets are characterized using several parameters. These include the number of objects in the database, the average sizes of the objects, the area of the underlying space, and the area coverage of all the objects. Mainly, these parameters, among others, are the ones suggested and discussed in Section 2. In this section, we only study the variation in the area coverage.

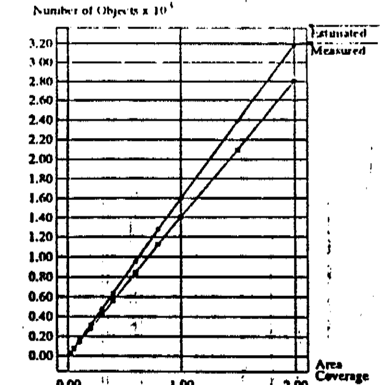
We verify the estimates for the spatial join operation that were derived in the previous section as follows. We generate 10 sets of random rectangles with the same parameter setting PS (in our case, the area coverage). We call the 10 sets a master data set, and denote it by MDS_{PS}. To measure the selectivity 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_{PS1} is joined with each set of rectangles in MDS_{PS2} using spatial join. The number of output pairs (object-pairs as well as block-pairs) is measured, summed and averaged (by dividing it by 100, which is the total number of times the spatial join was executed fo ' iis 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 area coverage of the input streams.

Figures 4a, 4b, and 4c compare the estimated value of the object selectivity factor for spatial join against the measured value. The area coverage of the first input stream is fixed at 0.01, 0.10, 1.0, respectively, while the area coverage of the second input stream varies from 0.01–2.0.

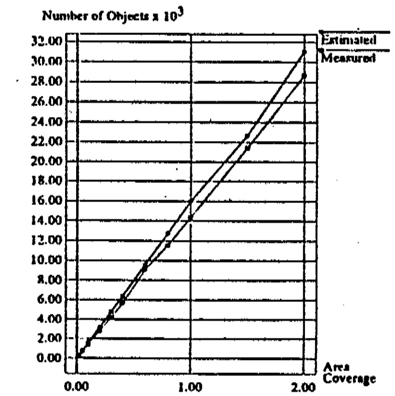
Figures 4d, 4e, and 4f compare the estimated value of the block selectivity factor for spatial join against the measured value. The area coverage of the first input stream is fixed at 0.01, 0.10, 1.0, respectively, while the area coverage of the second input stream varies from 0.01–2.0. The two estimates for block selectivity presented in Section 3.2 are plotted. As expected, the measured value is greater than Est-Max and less than



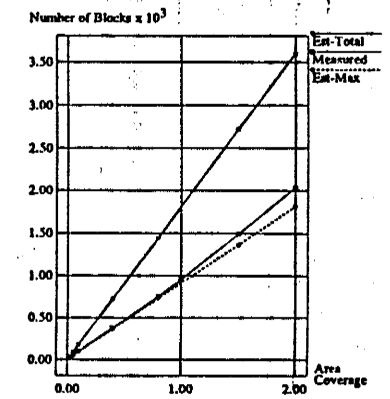
(a) Coverage = 1% (1st stream).



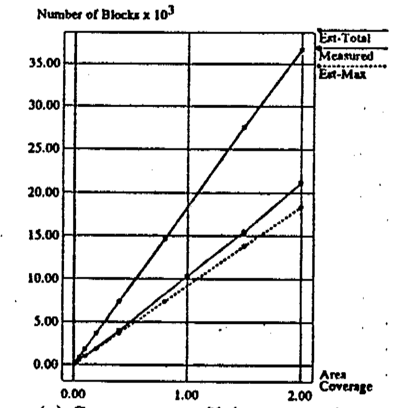
(b) Coverage = 10% (1st stream).



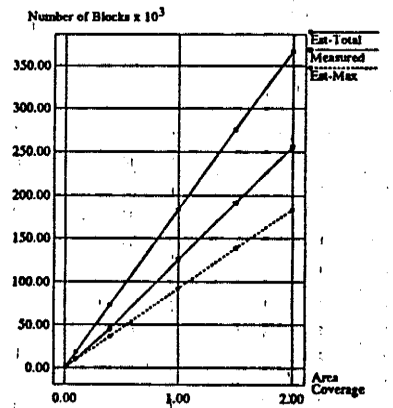
(c) Coverage = 100% (1st stream).



(d) Coverage = 1% (1st stream).



(e) Coverage = 10% (1st stream).



(f) Coverage = 100% (1st stream).

Figure 4: A comparison of the measured vs. the estimated object selectivity (a, b, and c), and block selectivity (d, e, and f) for the spatial join operation. The data coverage of the first input stream of the join is (a) 1%, (b) 10%, (c) 100%, (d) 1%, (e) 10%, and (f) 100%. The x-axis corresponds to the data coverage of the second input stream.

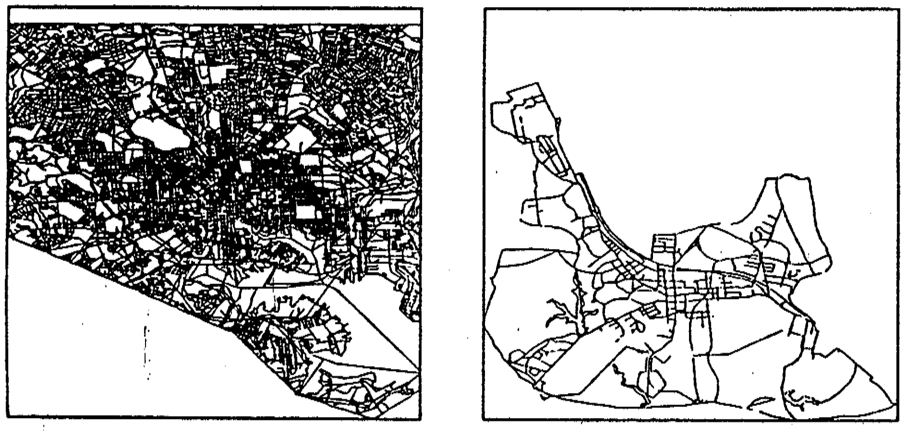

(a) Baltimore (b) Williamsburg

Figure 5: A sample of the Tiger/Line spatial databases used in the experiments.

Est-Total (those correspond to $BS_{j_{max}}$ and $BS_{j_{total}}$ defined in Section 3.2, respectively).

From the figures we can see that our estimates form an upper bound of the measured values. The maximum error between the two values is less than 30%, although in most cases it is much less.

4.2 Experiments with Real Data

We used the Tiger/Line databases (a sample is given in Figures 5a and 5b). We spatially join each pair of the Tiger/Line files together. The size of the underlying space for all the Tiger/Line data sets is normalized to 512 × 512. For each line segment in the Tiger/Line database, we constructed the line's minimum bounding rectangle. Each rectangle is then decomposed into its corresponding constituent pieces (in our case, Morton blocks). These data sets help verify our estimates for non-uniform distributions of objects in space.

In the case of estimating object selectivities, over 80% of the spatial joins performed had their estimated value of the object selectivity factor lie within 30% of the actual measured value, and 71% of the joins had their estimated value lie within 25% of the actual measured value. For block selectivity estimates using $BS_{j\text{-}total}$, over 80% of the join operations performed had their estimated value of the block selectivity factor lie within 30% of the actual measured value and 71% had their estimated value lie within 25% of the actual measured value. When using $\bar{BS}_{j\text{-}max}$ only 43% of the join operations

performed had their estimated value of the block selectivity factor lie within 30% of the actual measured value and 67% had their estimated value lie within 40% of the actual measured value. Therefore, the experiments show that $BS_{j\text{-}total}$ is a better estimate for the block selectivity factor than $\bar{BS}_{j\text{-}max}$. However, all three estimates are certainly good enough to be useful for query optimiation purposes. The experiments also show that our formulas for estimating the object and block 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.

5 Concluding Remarks

In estimating selectivity factors for spatial operations, several factors must be considered: the nature of the operation and the distribution of the objects in the underlying data set. Techniques for estimating selectivity factors in spatial databases differ from those used for relational databases. One major difference is the dimensionality of the data. As a result, two types of selectivity factors were introduced and were estimated in this paper: object-level, and block-level selectivity factors. The object-level selectivity formulas that were given in the paper apply regardless of any underlying data structure used. On the other hand, the block-level selectivity formulas are geared towards data structures that partition an object into more than one piece. Although demonstrated for this class of data structures, a similar framework can be modeled for other classes of data structures.

References

[1] I. Gargantini. An effective way to represent quadtrees. *Communications of the ACM*, 25(12):905–910, December 1982.

[2] 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.

[3] 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.

[4] J. A. Orenstein. Algorithms and data structures for the implementation of a relational database system. Technical Report SOCS-82-17, School Comput. Sci., McGill Univ., Montreal, Quebec, Canada, 1983.

[5] 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.

[6] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.