MITL

MATSUSHITA INFORMATION TECHNOLOGY LABORATORY

Query Size Estimation of Spatial Join

December 14, 1993

MITL-TR-79-93

Query Size Estimation of Spatial Join[1]

Walid G. Aref [2]
Matsushita Information Technology Laboratory
Panasonic Technologies, Inc.
Two Research Way
Princeton, New Jersey 08540
aref@mitl.research.panasonic.com

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

Abstract

Spatial join is a fundamental operation for answering queries that involve spatial predicates. It combines entities from two sets into single entities whenever the combination satisfies the spatial join condition (e.g., if they overlap in space). Many queries require performing more than one spatial join operation to answer them. In order to decide on the order in which to perform the spatial join operations, the selectivity of each operation has to be estimated. Highly selective spatial joins should be performed first. Spatial operations and spatial data sets require selectivity factors along each dimension of the space (e.g., two-dimensional selectivity factors) which are defined here in the context of spatial data structures that partition spatial objects into more than one piece. Two types of selectivity factors are estimated for the spatial join: the object-selectivity factor and the block- (or piece-) selectivity factor, which estimate the number of objects and object-pieces, respectively, that result from a spatial join. Both factors are useful for the purpose of cost-based spatial query optimization. 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. These parameters are computed by preprocessing the data sets. The formulas are validated experimentally using real and synthetic data sets.

Keywords: spatial databases, database systems, query optimization, selectivity factors, spatial join

December 14, 1993.

---

[1]The support of the National Science Foundation under Grant IRI-9017393 is gratefully acknowledged.
[2]The author conducted most of this research while at The University of Maryland, College Park.

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 organized using data structures that provide efficient access and flexible manipulation of the data. There are many ways to represent and organize a set of objects inside a data structure [14]. In the following we review some of these representations.

One 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 [18]), 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 [12]). The drawback of the former is that the transformation into the higher dimensional space does not preserve proximity. In particular, the representative points of two objects that are close to each other in the original n-dimensional space of the polygons are not necessarily close to each other in the higher dimensional space. Thus search is not always efficient. The drawback of the latter is that it does not result in a disjoint decomposition of space. This means that we may have to do extraneous work during search. Also, the search time appears to grow linearly with the number of objects stored in the database [1, 16].

An alternative 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 [10]), a collection of square blocks at predetermined positions (the quadtree [13]), or a collection of rectangles (the R^+-tree [7]). The price that must be paid for the disjointness is that in order to determine the area covered by a particular object, we have to retrieve al the cells that it occupies. A related drawback is that when we wish to determine all the objects that occur in a particular region we often retrieve many of the objects more than once. This is particularly problematic when the result of the operation serves as input to another operation via composition of functions. For example, suppose we wish to compute the perimeter of all the objects in a given region. Clearly, each object's perimeter should only be computed once. Eliminating the duplicates is a serious issue; but see [4, 5] for a discussion of how to deal with this problem for a database of line segments as well as rectangles.

From the above, we see that there is no ideal representation. In the rest of this paper we focus on the representation of objects by decomposing them into disjoint cells. In particular, we use regular decomposition representations such as the quadtree as they facilitate set operations since parts in different data collections (e.g., relations in a relational database) that span the same space are in registration. However, our methods are more general and can be applied to other representations as well.

One of the most useful tools for spatial query processing is the *spatial join*. The

spatial join is commonly used in answering queries involving spatial data sets. It combines entities from two sets into single entities whenever the combination satisfies the spatial join condition (e.g., if they overlap in space). In practice, many queries require performing more than one spatial join operation in order to answer them, or a combination of one or more relational join operations along with one or more spatial join operations. Deciding the order in which to perform the joins depends on several factors: the size of the input relations, the availability of indexes, the cost of the algorithms used for each join, the selectivity of each join, and the possibility of creating temporary relations.

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 of the output. Traditionally, in relational databases, a selectivity factor serves this purpose. It is mainly 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 the spatial join.

Generally speaking, spatial data sets and spatial operations require selectivity factors along each dimension of the space (e.g., two-dimensional selectivity factors). One way to deal with the multi-dimensionality 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 the meaning of a selectivity factor: *object-level* selectivity factor (*OS*) and *piece-level* selectivity factor (*PS*). They 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 object pieces determines the size of the output). In this paper, we present estimates of both selectivity factors for the spatial join operation. Notice that the two factors are not directly related to each other. For example, consider two objects, say o_1 and o_2, that intersect each other so that some object-pieces of o_1 intersect some object-pieces of o_2 resulting in a number of, say ten, object-piece intersections. This will be counted as one object intersection and ten object-piece intersections when computing the object and piece selectivity factors, respectively.

An important question that we address in this paper, which 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? There are a number of requirements that these statistics have to meet. One major requirement is that these statistics should be simple to maintain and update throughout the life time of the spatial database. In addition, they should depend on the following factors: (1) the data type of the underlying spatial objects, (2) the way these objects are represented, and (3) how these objects are organized inside the database. 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 data and the underlying access methods. For example, regarding datatypes, the statistics gathered to optimize queries for a spatial database of points may be different than 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 a 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 a simpler shape). Finally, with respect to access methods and data organizations, statistics that assume a grid-like data structure (e.g., the Grid File [18]) may be different from those that assume a tree-like structure (e.g. the R-tree [12]).

The rest of this paper is organized as follows. Section 2 presents our proposed characterization of the underlying spatial data set. Section 3 contains formulas for estimating selectivity factors of the spatial join operation. Section 4 describes our experimental setting for estimating selectivity factors as well as the experimental results while contrasting them with the developed formulas. Section 5 discusses related work on the topic of spatial query optimization. Section 6 provides several directions for extending the work presented in this paper. This includes handling of non-uniform data distributions. Section 7 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 determining the characteristics of the underlying data set, e.g., by sampling and gathering statistics [15]. However, at the present 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 follow a simpler approach. In particular, we assume that we are 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 a 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 factors and the cost of spatial operations, thereby serving 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 include the number of objects in the database, the total area covered by the objects in the database, the average area of the

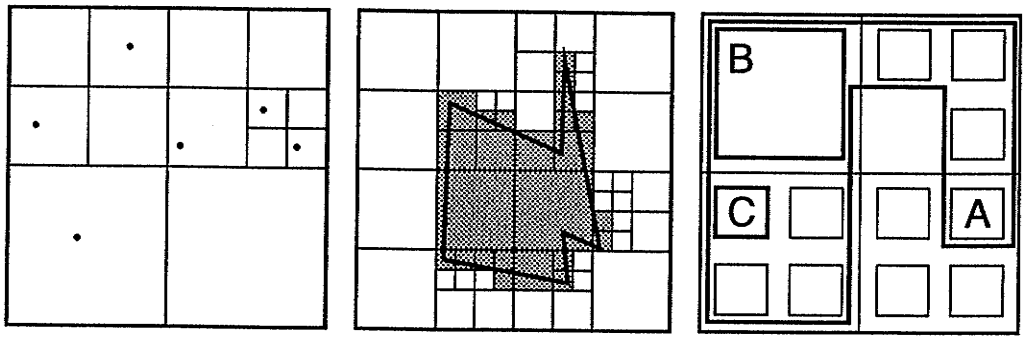


Figure 1: An example quadtree decomposition for (a) points, (b) polygonal region, and (c) a collection of overlapping regions (i.e., A,B,C).

objects, the number of blocks in the database, etc.

Regarding the representation and organization of spatial data, as we mentioned in Section 1, we concentrate 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 a regular decomposition of space so that a region is decomposed if it contains more than a predefined number of objects (e.g., points, or parts of different lines as in Figure 1a), or if it is a part of more than one region (e.g., if a region can be part of just one object as in Figure 1b), or if each block is totally covered by one or more objects, or is not covered by any of the objects (e.g., if objects are permitted to overlap as in Figure 1c). In these data structures, objects are represented by the set of blocks that collectively approximate and cover them. Frequently, the resulting blocks are linearized (e.g., Z-Order [22, 23], or a linear quadtree [9, 25]) in the sense that each block is represented by a unique number which enables it to be stored using a conventional index such as a B-tree. In such a case, the resulting blocks are called *Z-elements* or *Morton blocks* [17]. In this paper, we will assume objects such as those given in Figure 1b.

Assuming the use of the above data structure, we propose precomputing 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.
- A_{Space}: the area of the underlying space for the entire spatial database.
- C_{db}: the data coverage which is the ratio between the total area covered by the objects in the database and the area of the underlying space for the entire spatial database (A_{Space}).

- 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.
- AO_{Avg}: the average area of an object in the database. This parameter can be computed as follows:

$$AO_{Avg} = \frac{C_{db}}{NO_{db}} A_{Space}.$$

- AB_{Avg}: the average area of a block in the database. This parameter can be computed as follows:

$$AB_{Avg} = \frac{C_{db}}{NB_{db}} A_{Space}.$$

- R_{Avg}: the average *aspect ratio* of the minimum bounding rectangles of the objects in the database. The aspect ratio of object o is defined as $R_o = Y_o/X_o$, where X_o and Y_o are the width and height of the minimum bounding rectangle of o.
- NB_{Avg}: the average number of blocks per object in the database.

We use some of these parameters in the following section to estimate the selectivity factors for the spatial join. The remaining parameters are useful for other spatial operations (e.g., the window containment query) [2].

3 Estimates of Selectivity Factors for the Spatial Join

The selectivity factors for spatial join depend heavily on the type of spatial operator in the join predicate. This operator can be intersection, containment, etc. Examples are $db_1.a_1$ *intersects* $db_2.a_2$, and $db_1.a_1$ *contains* $db_2.a_2$, where a_1 and a_2 are spatial attributes of the databases db_1 and db_2, respectively. In this section, we examine the case of a spatial join with an intersection operator serving as the join predicate. Other operators can be dealt with analogously.

One way to compute the spatial join selectivity factors is to consider one of the two input streams of the spatial join as the underlying database and to consider the second input stream as a source for query windows where each window corresponds to the region occupied by an object from the second input stream (i.e., the inner component of the spatial join is the underlying spatial database and the outer component is a set of query windows). Following this approach enables us to first develop a selectivity formula for an intersection operation that involves just one window and then sum over all the windows w in the input stream corresponding to the outer component of the join.

As mentioned in Section 2, we assume that objects in the spatial database are represented by decomposing them into multiple, yet simple, pieces. We use the Z-Order [22, 23]

or the linear quadtree data structure [9, 25] as our final representation of the decomposition used in the spatial database. The decomposition scheme that we use is such that an object is represented by the set of blocks that conservatively cover the internal region of the object. Although such a decomposition rule is capable of being applied to a wide variety of object shapes and data types, in this paper 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.[3] It is important to mention that in developing the selectivity estimates in this paper we assume that the spatial objects are distributed uniformly in the underlying spatial database. However, in Section 4.2 we test the applicability of these formulas against data sets with non-uniform distributions (basically real data sets that consist of the road networks in the data of the U.S. Bureau of the Census Tiger/Line file [19]). In addition, in Section 6 we propose other techniques to handle the case of non-uniformly distributed objects in order to develop more accurate estimates. Notice that, even if we assume that objects are distributed uniformly in space, it is not true that the object-pieces, (e.g., the Z-elements or Morton blocks), 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 in the underlying space.

3.1 The Selectivity of One Window Intersection Operation

We start by estimating the selectivity factors of the intersection of just one window, i.e., when considering the intersection of just one rectangular object with the spatial database. In the next section we generalize the estimates that we develop to the more general spatial join operation (i.e., where more than just one object is considered in both spatial databases participating in the spatial join). The object-level selectivity for the window intersection operation, denoted OS_w, is defined as the number of objects in the underlying spatial database that intersect a given window. 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 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

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

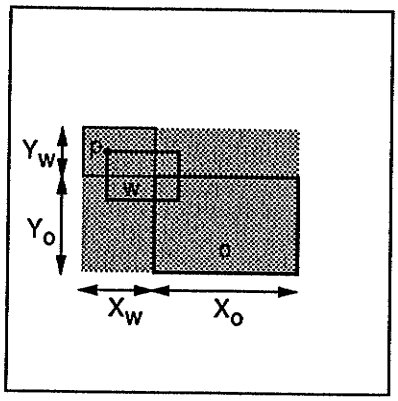


Figure 2: The rectangles w and o intersect if and only if the upper-left corner of w (point p) lies anywhere inside the shaded rectangle whose sides are $X_w + X_o$ and $Y_w + Y_o$.

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

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

where $C_w = \frac{A_w}{A_{Space}}$, and X_{Avg} and Y_{Avg} are the precomputed 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

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

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.

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 the uniformly 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. Let (x_p, y_p) be the coordinate values of the upper-left corner of w. From Figure 2, the upper-left corner of w is equally likely to be at any location inside the rectangle $[x_p, x_p + X_w + X_o] \times [y_p, y_p + Y_w + Y_o]$, i.e., with probability $\frac{1}{(X_w+X_o)(Y_w+Y_o)}$. To obtain the average area of intersection, we compute the area of intersection when w is at some arbitrary location (x, y) and multiply it by its probability of occurrence. Next, we integrate the area over all possible values of x and y inside the rectangle $[x_p, x_p + X_w + X_o] \times [y_p, y_p + Y_w + Y_o]$. The resulting average area of intersection in this case is given by (see the Appendix for a complete derivation):

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

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

$$\begin{aligned} NB_{ow} &= \frac{A_{ow}}{A_o} NB_o \\ &= \frac{A_w}{(X_w + X_o)(Y_w + Y_o)} NB_o \end{aligned}$$

where NB_o is the number of blocks comprising object 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:

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

Substituting for NB_{ow} in the above formula, we get

$$\begin{aligned} BS_w &= \frac{1}{NB_{db}} \frac{A_w}{A_{Space}} \sum_{o \in db} NB_o \\ &= \frac{A_w}{A_{Space}} \equiv C_w \end{aligned}$$

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):

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

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 object selectivity factor for the spatial join depends only on the average area (or the extent) of the objects in the underlying spatial database.

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 or equal to $\max(NB_{ow}, NB_{wo})$ (e.g., Figure 3e where it is 2)[4]. 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

[4]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 $n + m - 1$.

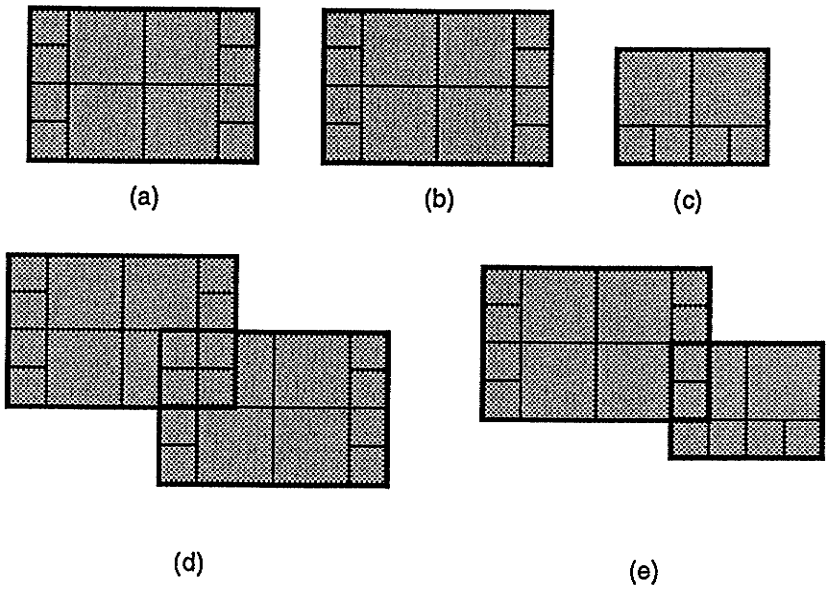


Figure 3: (a)-(c) A decomposition of three rectangles into quadtree blocks that cover the internal of the rectangles. (d) Although the number of blocks inside the intersection region is 3 blocks in each rectangle, there are 4 intersecting pairs of blocks. (e) The best case occurs when all the blocks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

of intersection between o and w. This results in the following estimates:

$$BS_{j\text{-}total} = \frac{1}{NB_{db_1} NB_{db_2}} \sum_{w \in db_1} \sum_{o \in db_2} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} (NB_{ow} + NB_{wo})$$

$$BS_{j\text{-}max} = \frac{1}{NB_{db_1} NB_{db_2}} \sum_{w \in db_1} \sum_{o \in db_2} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} \max(NB_{ow}, NB_{wo})$$

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.,

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

As we will see in Section 4, in some cases BS_j can be slightly greater than $BS_{j\text{-}total}$. This is due to the way we use average values.

By substituting the estimated values of NB_{ow} and NB_{wo} in the formula for computing $BS_{j\text{-}total}$, we get

$$BS_{j\text{-}total} = \frac{1}{NB_{db_1} NB_{db_2}} \Big(\sum_{w \in db_1} \sum_{o \in db_2} \frac{(X_w + X_o)(Y_w + Y_o)}{A_{Space}} \frac{A_w}{(X_w + X_o)(Y_w + Y_o)} NB_o +$$

$$\sum_{w\in db_1}\sum_{o\in db_2}\frac{(X_w+X_o)(Y_w+Y_o)}{A_{Space}}\frac{A_o}{(X_w+X_o)(Y_w+Y_o)}NB_w)$$

$$= \frac{1}{NB_{db_1}NB_{db_2}}(\sum_{w\in db_1}\frac{A_w}{A_{Space}}\sum_{o\in db_2}NB_o+\sum_{w\in db_1}NB_w\sum_{o\in db_2}\frac{A_o}{A_{Space}})$$

$$= \frac{1}{NB_{db_1}NB_{db_2}}(C_{db_1}NB_{db_2}+C_{db_2}NB_{db_1})$$

$$= \frac{C_{db_1}}{NB_{db_1}}+\frac{C_{db_2}}{NB_{db_2}}, \text{ or equivalently,}$$

$$= \frac{AB_{Avg1}}{A_{Space}}+\frac{AB_{Avg2}}{A_{Space}}$$

where the last formula is only dependent on the block and object average sizes in both databases.

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

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

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

$$\bar{BS}_{j\text{-}max} = \max(\bar{BS}_{j\text{-}max1}, \bar{BS}_{j\text{-}max2}), \text{ where}$$

$$\bar{BS}_{j\text{-}max1} = \frac{1}{NB_{db_1}NB_{db_2}}\sum_{w\in db_1}\sum_{o\in db_2}\frac{(X_w+X_o)(Y_w+Y_o)}{A_{Space}}NB_{ow}, \text{ and}$$

$$\bar{BS}_{j\text{-}max2} = \frac{1}{NB_{db_1}NB_{db_2}}\sum_{w\in db_1}\sum_{o\in db_2}\frac{(X_w+X_o)(Y_w+Y_o)}{A_{Space}}NB_{wo}.$$

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

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

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

$$\bar{BS}_{j\text{-}max1} = \frac{1}{NB_{db_1}NB_{db_2}}\sum_{w\in db_1}\frac{A_w}{A_{Space}}\sum_{o\in db_2}NB_o$$

$$= \frac{C_{db_1}}{NB_{db_1}} = \frac{AB_{Avg1}}{A_{Space}}$$

$$\bar{BS}_{j\text{-}max2} = \frac{1}{NB_{db_1}NB_{db_2}}\sum_{w\in db_1}NB_w\sum_{o\in db_2}\frac{A_o}{A_{Space}}$$

$$= \frac{C_{db_2}}{NB_{db_2}} = \frac{AB_{Avg2}}{A_{Space}}.$$

Therefore,

$$\begin{aligned} \bar{BS}_{j\text{-}max} &= \max(\frac{C_{db_1}}{NB_{db_1}}, \frac{C_{db_2}}{NB_{db_2}}), \text{ or equivalently,} \\ &= \max(\frac{AB_{Avg1}}{A_{Space}}, \frac{AB_{Avg2}}{A_{Space}}). \end{aligned}$$

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 $NB_{db_1} \times NB_{db_2}$ (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).

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 formulas for the object and block selectivity factors of the self join, OS_{sj} and BS_{sj}, respectively. 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.

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

Similarly,

$$BS_{sj\text{-}total} = BS_{j\text{-}total} + \frac{1}{NB_{db}}$$

Data Coverage	Area of Rectangle	Number of Rectangles	Average Number of Morton Blocks
0.01	64	40	950
0.10	64	409	9385
0.20	64	819	18937
1.00	64	4096	94239
1.50	64	6144	142086
2.00	64	8192	189496
3.00	64	12288	283062

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

$$
\begin{aligned}
 & = 2\frac{C_{db}}{NB_{db}} + \frac{1}{NB_{db}} \\
 & = \frac{2C_{db}+1}{NB_{db}}, \text{ or equivalently,} \\
 & = 2\frac{AB_{Avg}}{A_{Space}} + \frac{1}{NB_{db}} \\
\bar{BS}_{sj\text{-}max} & = \bar{BS}_{j\text{-}max} + \frac{1}{NB_{db}} \\
 & = \frac{C_{db}}{NB_{db}} + \frac{1}{NB_{db}} \\
 & = \frac{1+C_{db}}{NB_{db}}, \text{ or equivalently,} \\
 & = \frac{AB_{Avg}}{A_{Space}} + \frac{1}{NB_{db}}
\end{aligned}
$$

4 Experimental Results

The purpose of our experiments is to verify the significance of the derived formulas for estimating selectivity factors under varying conditions. Experiments 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 [19] for representing the roads and other geographic features in the U.S. The synthetic data sets are collections of rectangles generated at random. Each rectangle was subsequently decomposed into its corresponding constituent Morton blocks. Notice that since it is possible for the generated rectangles to overlap, the same is true for the Morton blocks inside the database.

Data Coverage	Area of Rectangle	Number of Rectangles	Average Number of Morton Blocks
1.00	128	2048	69813
1.00	512	512	40010
1.00	1024	256	29096
1.00	2048	128	21754
1.00	16384	16	8404

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

4.1 Experiments with Synthetic Data

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

We conducted the following experiment: for each entry in Tables 1 and 2, we generate 10 sets of random rectangles 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_{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 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 4 compares the estimated value of the object 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 object selectivity factor of the spatial join is constant (appoximately, 95×10^{-5}). This is because the estimates for the object 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 object selectivity factor that resulted from the experiments. It is found that they are relatively constant and within 20% of the estimated value.

Figure 5 performs the same comparison while varying the average areas of the objects

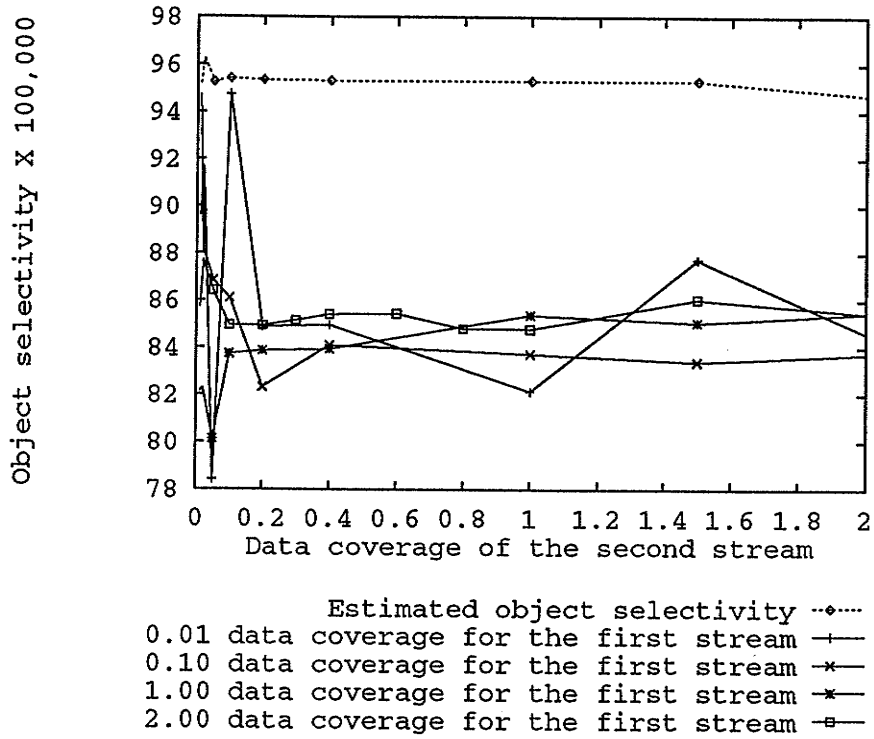


Figure 4: A comparison of the measured vs. the estimated object selectivity for the spatial join operation as a function of the data coverage of the second stream.

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 5 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 object selectivity factor of the spatial join is within 10% of the measured values.

Figure 6 compares the estimated value of the block selectivity factor for the spatial join against the measured value for different data coverages. The two estimates for block selectivity presented in Section 3.2 are plotted (namely, $\bar{BS}_{j\text{-}max}$ and $BS_{j\text{-}total}$ where they correspond to the plots for Est-Max and Est-Total in the figure, respectively). The data coverage of the first input stream was fixed at 0.01, 0.1, 1.0, 2.0, and 3.0, respectively, while the data coverage of the second input stream varied from 0.01–2.0. As expected, the measured value is greater than Est-Max and less than Est-Total. Notice that for spatial databases with small data coverages, Est-Max is a better estimate than Est-Total. On the other hand, as the data coverage gets larger (and hence more object overlaps occur), the measured values for the block selectivity factors approach the estimate Est-Total. This can be used by the query optimizer as a strategy for choosing between the two estimates.

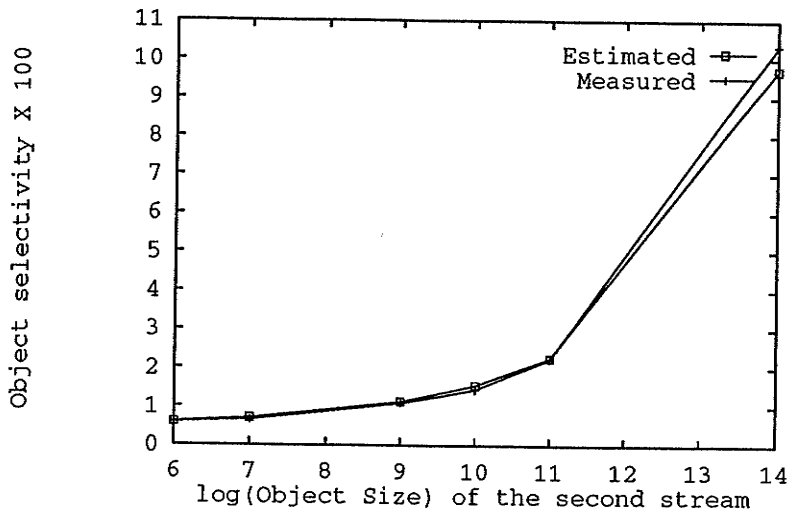


Figure 5: A comparison of the measured vs. the estimated object selectivity for the spatial join operation as a function of different average object areas. The data coverage of both input streams of the join is 100%. The average area of the objects in the first stream is 1024. The x-axis corresponds to the average area of the objects of the second stream.

Figure 7 provides a comparison of the estimated and measured block selectivity for the spatial join while varying the average areas of the objects in one of the input streams. The average area of an object in the other stream is 1024. From the graph, we notice that the measured selectivity value is greater than Est-Max and less than Est-Total which correspond to $\bar{BS}_{j\text{-}max}$ and $BS_{j\text{-}total}$ defined in Section 3.2, respectively. This is true for all cases. It is worth mentioning that although our estimates produce satisfactory results, they ignore the fact that smaller blocks are closer to the boundary of the objects. In particular, the estimate of the number of intersecting object-blocks NB_{ow}, i.e.,

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

assumes that size distribution of the blocks inside an object is uniform, which is not the case for the quadtree decomposition of space. In this case, a better estimate can be developed that takes into consideration the fact that the blocks inside the object are not equally distributed but are clustered more towards the borders.

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.

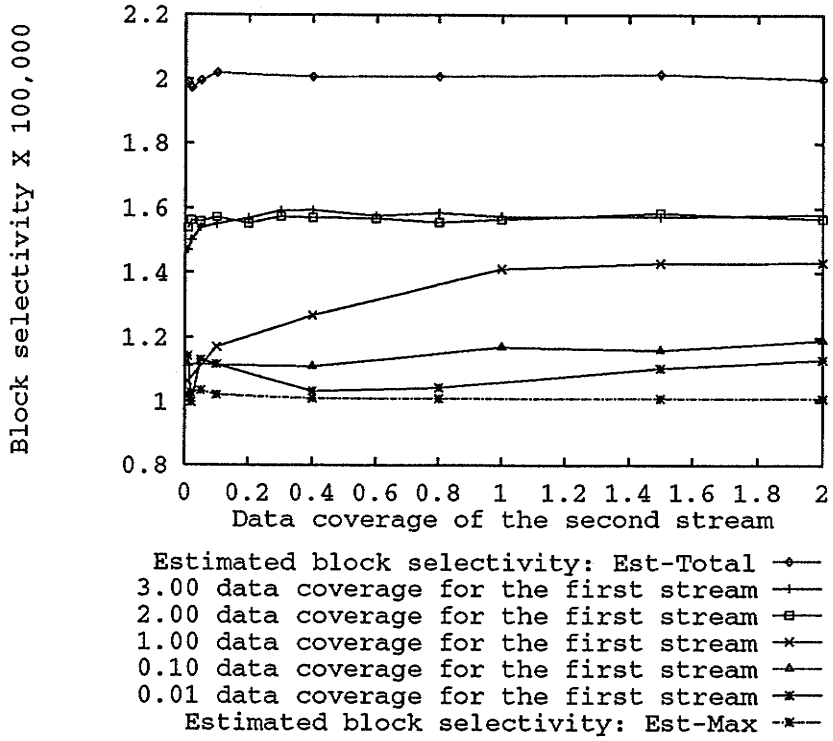


Figure 6: A comparison of the measured vs. the estimated block selectivities for the spatial join operation as a function of the data coverage of the second stream.

4.2 Experiments with Real Data

We used five Tiger/Line databases (given in Figures 8a–8f). For each line segment in the Tiger/Line database, we constructed the line's minimum bounding rectangle. Each rectangle was then decomposed into its corresponding constituent Morton blocks. These data sets help verify our estimates for arbitrary distributions of objects in space.

Table 3 gives a characterization of the real data sets (the Tiger/Line) files using some of the parameters suggested and discussed in Section 2. These include the area coverage of all the objects, the average area of all the bounding boxes that include each line segment in the database, the number of objects, and the number of object-pieces in the entire database. The size of the underlying space for all the Tiger/Line data sets 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 normalize 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.

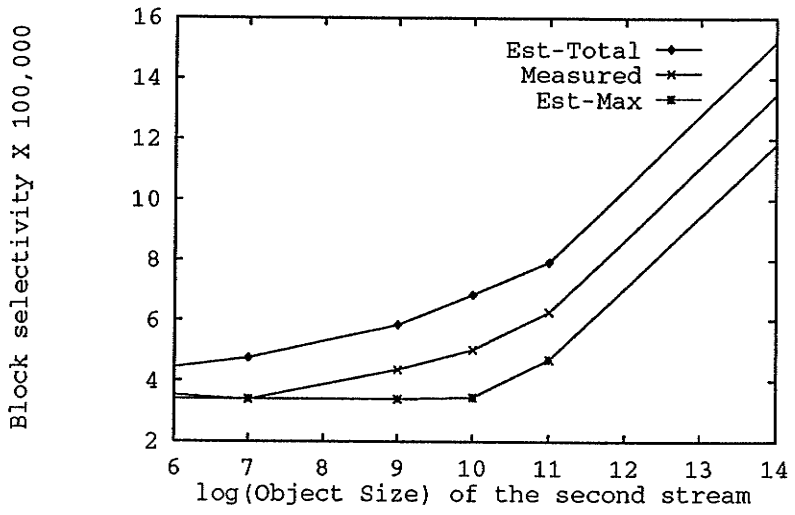
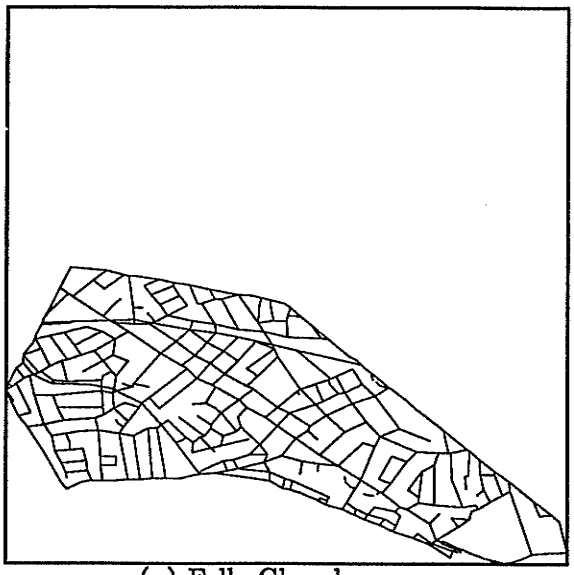


Figure 7: A comparison of the measured vs. the estimated block selectivity for the spatial join operation as a function of different average object sizes.. The data coverage of both input streams of the join is 100%. The average area of the objects in the first stream is 1024. The x-axis corresponds to the average area of the objects of the second stream.

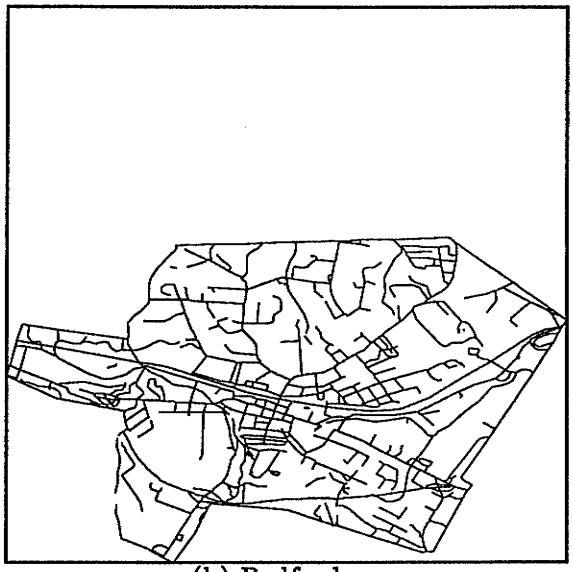
Table 4 gives the relative error in the estimated value of the object selectivity factor of the spatial join over the measured value. In the table, a negative percentage indicates that the measured value was smaller than the estimated value while a positive percentage indicates that the estimated value was smaller. On the other hand, Tables 5 and 6 give the relative error in the estimated value of the block selectivity factors of the spatial join: $BS_{j\text{-}total}$ and $\bar{BS}_{j\text{-}max}$, respectively, over the measured value of the block selectivity. The total number of data set pairs that we spatially joined together was 15. Figure 9 summarizes 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.

Map Name	Data Coverage	Avg. Area of Bound. Box	Number of Line-segs.	Number of Morton Blks.
Falls Church (FC)	0.43	192.10	587	24330
Bedford (BF)	0.34	54.17	1644	27725
Williamsburg (WB)	0.19	23.62	2112	20686
Franklin (FR)	0.35	54.46	1685	25708
Washington D.C. (DC)	0.56	7.90	18517	94934

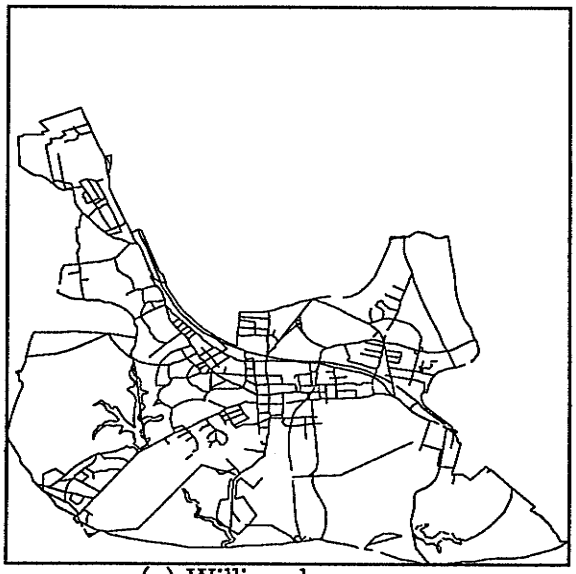
Table 3: Parameter setting for the Tiger data files. The size of all the maps is normalized to 512 x 512.



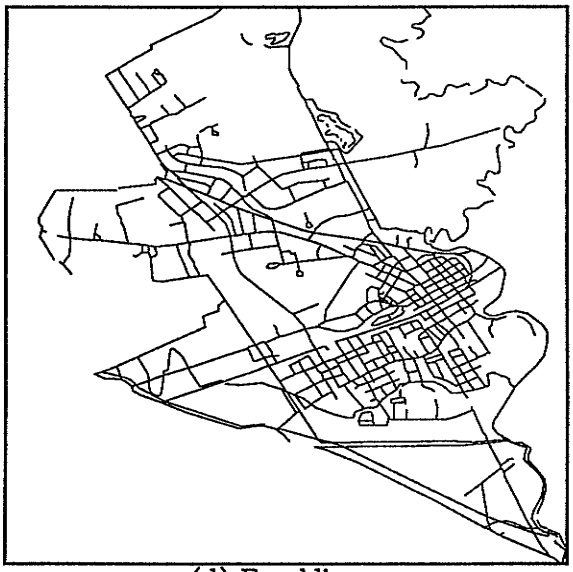
(a) Falls Church



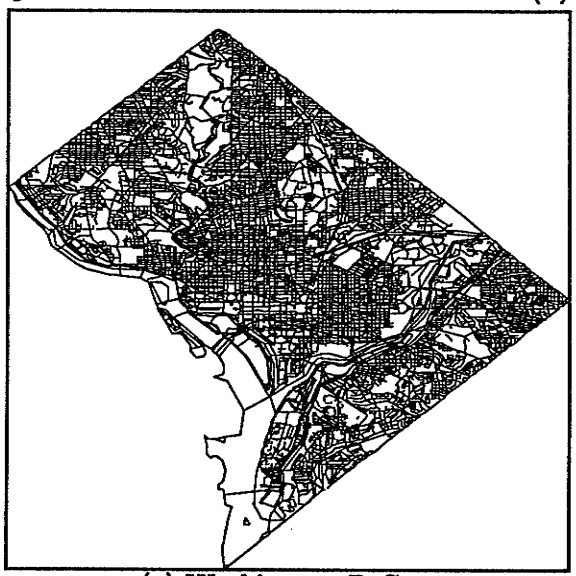
(b) Bedford



(c) Williamsburg



(d) Franklin



(e) Washington D.C.

Figure 8: The Tiger/Line spatial databases used in the experiments.

In the case of estimating object selectivities, 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 object 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. For block selectivity estimates using $BS_{j\text{-}total}$, 14 out of the 15 join operations (i.e., over 93% of the join operations performed) have their estimated value of the block selectivity factor lie within 30% of the actual measured value, while 13 out of 15 (i.e., over 86%) have their estimated value lie within 25% of the actual measured value. For block selectivity estimates using $\bar{BS}_{j\text{-}max}$, only 4 out of the 15 join operations (i.e., 26% of the join operations performed) have their estimated value of the block selectivity factor lie within 30% of the actual measured value, while 9 out of 15 (i.e., 60%) have their estimated value lie within 40% of the actual measured value.

Therefore, we see that the experiments show that $BS_{j\text{-}total}$ is a better estimate for the block selectivity factor than $\bar{BS}_{j\text{-}max}$. This confirms our conclusions for synthetic data in Section 4.1. As the data coverage increases, more overlap among objects (and hence object-pieces) is expected and hence $BS_{j\text{-}total}$ will be the better estimate. The real data sets have high data coverage in the regions where data is concentrated and hence $BS_{j\text{-}total}$ is a better estimate for the overlaps that take place. The experiments in this section confirm this conclusion as well. However, all three estimates are certainly good enough to be useful for query optimization 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.

	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 object selectivity factor of the spatial join over the measured value.

5 Related work

In the past, very little attention was devoted to spatial query processing and optimization. Existing spatial database systems have basically ignored the optimization issues. Little is known about the different strategies and alternatives for processing spatial queries (i.e., when interspersing spatial and nonspatial operations) as well as about the cost of executing alternative query evaluation plans containing spatial operations. However, recently several prototype systems address the issue of spatial query optimization, (e.g.,

	FC	BF	WB	FR	DC
FC	-0.12	0.06	-0.08	-0.22	-0.27
BF		0.01	0.05	-0.07	-0.08
WB			-0.13	-0.10	0.16
FR				-0.14	-0.18
DC					0.37

Table 5: The relative error in the value of the estimated block selectivity factor of the spatial join, $BS_{j\text{-}total}$, over the measured value.

	FC	BF	WB	FR	DC
FC	0.31	0.37	0.38	0.54	0.41
BF		0.19	0.39	0.51	0.37
WB			0.24	0.46	0.28
FR				0.30	0.41
DC					0.46

Table 6: The relative error in the value of the estimated block selectivity factor of the spatial join, $\bar{BS}_{j\text{-}max}$, over the measured value.

SAND [3], Gral [11], GEOQL [24], and Geo-Kernel [27]). [3] presents several strategies for evaluating such queries. GEOQL [20, 21, 24] has an extended query optimizer that fully decomposes a given query into simpler spatial-only and nonspatial-only subqueries. The results of all the subqueries are later merged together (similar to the query decomposition technique in [28] but extended to handle spatial operations). However, the optimizer for GEOQL only optimizes the cost of nonspatial operations and does not take into account the I/O cost of spatial operations or the selectivity factors of the operations. Gral [6, 11] uses an algebraic query language at both the query description and execution levels. In particular, a rule-based optimizer is used to normalize and optimize at the descriptive algebra level. In addition, rules are used to translate a query into its executable equivalent as well as to express heuristics such as performing selects before joins. The developers of Gral place a greater emphasis on expressing query optimization techniques using rules. However, there is little emphasis on providing good cost estimates as well as selectivity factors of spatial operations. Instead, they choose ad hoc selectivities for spatial operations (similar to the selectivities of Selinger et al. [26]).

Geo-Kernel [27] uses some heuristics for query optimization and is not based on a cost-model for spatial operations. For example, the query optimizer in [27] adopts a useful heuristic which prefers to perform relational selections before spatial selections in order to reduce the cost of spatial data conversion (i.e., downloading) by reducing the number of qualifying tuples. However, this is not based on actual cost estimates.

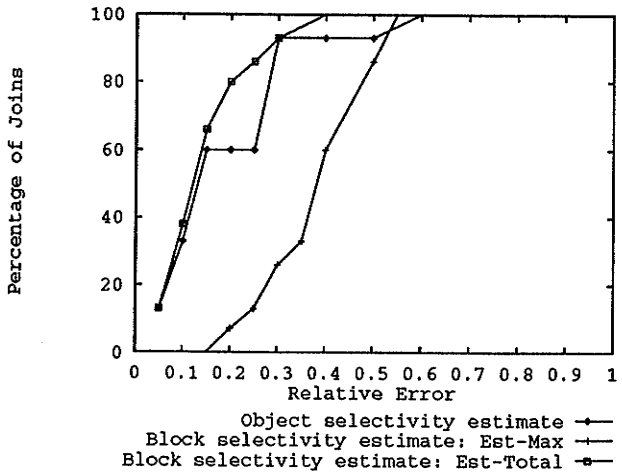


Figure 9: The number of spatial joins whose estimates were within the relative error given in the x-axis.

6 Extensions and Future Work

A very 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. The data distribution depends heavily on the application domain. For example, the distribution of rectangles in a VLSI circuit may be different from the distribution of plan-icons in a floor plan, and both of these distributions may be different from the distribution of line segments in a downtown city map. These distributions may vary: from simple uniform distributions, to skewed non-uniform distributions. It is obvious that some knowledge of the data distribution of the underlying spatial database is required in order to guide the selection of useful statistics for query processing and optimization purposes. It also affects the types of the parameters selected. Although this topic is not emphasized much in this paper, in this section we develop a framework for future research in this area. In addition to handling uniformly distributed data sets, we approximate data sets that have a non-uniform distribution by using a piece-wise uniform distribution. We also point out in this section how the query optimizer can benefit from knowing the distribution of the location of user queries.

6.1 Distribution of Spatial Objects and Locations of User Queries

The distribution of spatial data objects in space varies according to the underlying application (e.g., geographic database, VLSI circuit database, medical imagery database, etc.). For example, if the spatial database is to model geographic data, then the distri-

bution of geographical objects in space is known to be non-uniform [8]. In the case of geographical maps, spatial objects are usually clustered around some cluster points (e.g., cities), while in the case of VLSI circuit layouts, objects may be uniformly distributed in space.

Several factors impede the process of gathering statistics from the underlying database for the purpose of query optimization. First, incorporating an accurate distribution for the underlying spatial database inside the query optimizer is time-consuming. On the other hand, although simple to model and maintain, an over-simplified distribution for the objects in the underlying database does not help to estimate selectivity factors because the spatial objects are usually clustered. This problem is most apparent when spatial objects are non-uniformly distributed in space.

One way to model non-uniform data distributions for query optimization purposes is to assume some piece-wise uniform distribution of the spatial objects in the underlying database. This implies that the database be sub-divided into blocks, and that statistics are gathered for each block independently while assuming a uniform distribution inside each block. When a user query is directed towards a certain block, or a group of blocks, the statistics of these blocks will be involved in estimating the selectivity of the query. The number of blocks has to be kept small so that their management and access will be easy for the optimizer.

Another aspect to consider is the distribution of the user queries. Depending on the application, the user queries may be either distributed uniformly over the whole space or targeted towards particular locations in the database. Usually, users' queries are directed more towards the locations in the database that have a higher density of data. As a result, non-uniformity in user queries is likely to happen in geographic applications since the distribution of geographic data is non-uniform. For example, downtown sites are queried much more often than some other neighboring rural sites. Table 7 (columns 1–3) gives all possible combinations of spatial data distributions and user-query distributions.

There are several approaches to addressing data and query modeling inside a query optimizer for each of the cases given in Table 7. In this paper, we treat cases 1 and 2 in the same way, i.e., when the data distribution is uniform, our model for computing selectivities is insensitive to the spatial distribution of user queries. The rationale for our reasoning is that it is relatively easy to model data that is uniformly distributed. So, the optimizer will be able to estimate the outcome of the user queries quite well regardless of the nature of the user queries, i.e., regardless of whether or not the distribution of user queries are themselves uniform or non-uniform. We call this approach the uniform data model. For the case of non-uniform data distributions (cases 3 and 4), the optimizer has two choices: either to ignore this non-uniformity in the data distribution and hence assume some uniform distribution of the data, i.e., treat these cases in the same way as cases 1 and 2, or to assume some piece-wise uniform distribution as described above. We call the latter approach the *piece-wise uniform data model.* When user queries are also non-uniform (case 4 only), the optimizer can still model the data distribution as a uniform distribution in the following sense. The optimizer can collect statistics from the

dense areas only and then generalize these statistics over the whole space, thus assuming a uniform distribution over the whole space. Although the areas with low densities will result in false estimates for selectivity factors, these areas are not queried much by the user according to our assumption that user queries are also non-uniform, and that user queries are typically directed towards the areas with dense data. We call this approach the *false uniform model.* Column 4 of Table 7 summarizes the alternative approaches that can be taken by the optimizer to model spatial data and user queries in each case.

Case Study	Spatial Data Distribution	User Query Distribution	Alternative Models
1	uniform	uniform	The uniform model
2	uniform	non-uniform	The uniform model
3	non-uniform	uniform	The uniform model The piece-wise uniform model
4	non-uniform	non-uniform	The uniform model The piece-wise uniform model The false uniform model

Table 7: Possible combinations of spatial data and query distribution, and their corresponding models.

6.2 Selectivity factors for non-uniformly distributed data

The parameters described in Section 2 are computed differently in the case of non-uniformly distributed data. In the case of the uniform model, these parameters are pre-computed over the whole set of objects in the database. On the other hand, in the case when data is non-uniformly distributed, i.e., piece-wise uniform model, the database is divided into a coarse uniform grid. When the database is preprocessed, the parameters are computed 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. In the rest of this section we describe formulas that can be used for estimating selectivity factors for the spatial join operation with a join predicate that involves window intersection. Following the same approach of Section 3, we start with the case of one window intersection operation, and then use the resulting formulas to estimate the selectivity factors for the spatial join.

6.2.1 One Window Intersection

The piece-wise uniform model: the formulas in Section 3.1 assume that spatial objects are distributed uniformly in space. For the piece-wise uniform model, described in Section 6.1, data is non-uniformly distributed in space and this non-uniformity is modeled with a piece-wise uniform distribution. Assume that the database is divided into a coarse uniform grid such that each grid cell is of width and height X_g and Y_g respectively, (g stands for *g*rid). In this case, the formulas in Section 3.1 stay the same except that the precomputed parameters (i.e., X_{Avg}, Y_{Avg}, etc.) will be replaced by the parameters of the grid cell that overlaps the query window. When the database is preprocessed, parameters are computed for each grid cell separately. This results in a two-dimensional array for each precomputed 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. Assume a query window, say w, whose upper-left corner has coordinate values (x_p, y_p). Let

$$\begin{aligned}
X_{Avg}(w) &= X_{Avg}[\frac{x_p}{X_g}][\frac{y_p}{Y_g}], \\
Y_{Avg}(w) &= Y_{Avg}[\frac{x_p}{X_g}][\frac{y_p}{Y_g}], \\
NO_{db}(w) &= NO_{db}[\frac{x_p}{X_g}][\frac{y_p}{Y_g}], \\
NB_{db}(w) &= NB_{db}[\frac{x_p}{X_g}][\frac{y_p}{Y_g}], \text{ and} \\
C_{db}(w) &= C_{db}[\frac{x_p}{X_g}][\frac{y_p}{Y_g}].
\end{aligned}$$

Then,

$$\begin{aligned}
OS_w &= C_w + \frac{C_{db}(w)}{NO_{db}(w)} + \frac{Y_w X_{Avg}(w) + X_w Y_{Avg}(w)}{A_{Cell}}, \text{ and} \\
BS_w &= \frac{A_w}{A_{Cell}}
\end{aligned}$$

where A_{Cell} is the area of each grid cell and is equal to $X_g Y_g$. Notice that the number of output objects and the number of output blocks can be estimated by multiplying OS_w and BS_w by $NO_{db}(w)$ and $NB_{db}(w)$, respectively.

The false uniform model: the formulas described above for the piece-wise model assumes that the underlying space is decomposed into a coarse grid and that the preprocessing parameters are gathered for each grid cell. For the false uniform model, described in Section 6.1, in addition to the non-uniformity of the distribution of objects in space, we also assume that the distribution of the user queries is non-uniform. Furthermore, we assume that both distributions are almost alike. For example, this implies that the density of objects in a certain location in space is directly proportional to the density of the user queries to the same location. As a result of these assumptions, in the false

uniform model we neglect portions of space where no objects are contained. As a result, instead of using A_{Space} in the formulas for estimating the selectivity factors, we use $\bar{A}_{Space}$ where $\bar{A}_{Space}$ is the sum of the areas of the non-empty grid cells of the underlying space. The rest of the optimization parameters are gathered for the entire space (not per grid cell). This results in the following object and block selectivity estimates:

$$\begin{aligned} OS_w &= \bar{C}_w + \frac{C_{db}}{NO_{db}} + \frac{Y_w X_{Avg} + X_w Y_{Avg}}{\bar{A}_{Space}}, \text{ and} \\ BS_w &= \bar{C}_w \end{aligned}$$

where $\bar{C}_w = \frac{A_w}{\bar{A}_{Space}}$ In other words, the false uniform model computes the preprocessing parameters in the same way as the uniform model except that it assumes that the empty space is non-existent (since it will not be queried by the user, according to the model assumptions) and hence is eliminated. One problem with the false uniform model is that it will not produce satisfactory results when the non-empty areas of the space have large variations in the densities of the objects. In this case the piece-wise uniform model would be more appropriate.

6.2.2 The Spatial Join Operation

The piece-wise uniform model: For a spatial join with an intersection predicate, objects inside a grid cell g in one stream only join with objects that are within the same corresponding grid cell in the second stream (for simplicity, we assume that the grid cells are of the same size in both streams). As a result, the formulas for estimating OS_j and BS_j need to be computed locally for each grid cell, using the formulas of Section 3.2, and then the results must be summed over all the grid cells. This results in the following formulas:

$$OS_j = \sum_{g \in \text{grid cells}} \frac{C_{db_1}(g)}{NO_{db_1}(g)} + \frac{C_{db_2}(g)}{NO_{db_2}(g)} + \frac{X_{Avg1}(g)Y_{Avg2}(g) + X_{Avg2}(g)Y_{Avg1}(g)}{A_{Cell}}$$

or equivalently,

$$OS_j = \sum_{g \in \text{grid cells}} \frac{AO_{Avg1}(g)}{A_{Cell}} + \frac{AO_{Avg2}(g)}{A_{Cell}} + \frac{X_{Avg1}(g)Y_{Avg2}(g) + X_{Avg2}(g)Y_{Avg1}(g)}{A_{Cell}}$$

$$BS_{j\text{-}total} = \sum_{g \in \text{grid cells}} \frac{C_{db_1}(g)}{NB_{db_1}(g)} + \frac{C_{db_2}(g)}{NB_{db_2}(g)}$$

or equivalently,

$$BS_{j\text{-}total} = \sum_{g \in \text{grid cells}} \frac{AB_{Avg1}(g)}{A_{Cell}} + \frac{AB_{Avg2}(g)}{A_{Cell}}$$

$$\bar{BS}_{j\text{-}max} = \sum_{g \in \text{grid cells}} \max\left(\frac{C_{db_1}(g)}{NB_{db_1}(g)}, \frac{C_{db_2}(g)}{NB_{db_2}(g)}\right)$$

or equivalently,

$$\bar{BS}_{j\text{-}max} = \sum_{g \in \text{grid cells}} \frac{AB_{Avg1}(g)}{A_{Cell}} + \frac{AB_{Avg2}(g)}{A_{Cell}}.$$

The false uniform model: this is similar to what we performed for the false uniform model in the case of the one-window selectivity formulas (Section 6.2.1). In other words, the empty regions (i.e., the ones containing no spatial objects) of the underlying space are excluded from being considered in the selectivity formulas. As a result, instead of using A_{Space} in the formulas for estimating the selectivity factors, we use $\bar{A}_{Space}$ where $\bar{A}_{Space}$ is the sum of the areas of the non-empty grid cells of the underlying space. The rest of the optimization parameters are gathered for the entire space (not per grid cell). These parameters along with $\bar{A}_{Space}$ are directly substituted in the formulas for object-level and block-level selectivity factors of the spatial join that were derived in Section 3.2. This results in the following formulas.

$$\begin{aligned}
OS_j &= \frac{AO_{Avg1}}{\bar{A}_{Space}} + \frac{AO_{Avg2}}{\bar{A}_{Space}} + \frac{X_{Avg1}Y_{Avg2} + X_{Avg2}Y_{Avg1}}{\bar{A}_{Space}} \\
BS_{j\text{-}total} &= \frac{AB_{Avg1}}{\bar{A}_{Space}} + \frac{AB_{Avg2}}{\bar{A}_{Space}} \\
\bar{BS}_{j\text{-}max} &= \max\left(\frac{AB_{Avg1}}{\bar{A}_{Space}}, \frac{AB_{Avg2}}{\bar{A}_{Space}}\right)
\end{aligned}$$

7 Concluding Remarks

In estimating selectivity factors for spatial operations, several factors must be considered: the nature of the operation, the distribution of the underlying data set, and the distribution of the user queries. 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 estimated in this paper: object-level, and block-level selectivity factors. The object-level selectivity formulas that were presented are applicable regardless of the 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 our discussion has been for this class of data structures, a similar framework can be developed for other classes of data structures.

References

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In David B. Lomet, editor, *Foundations of Data Organization and Algorithms*, pages 69–84, Berlin, 1993. Lecture Notes in Computer Science 730. Springer-Verlag.

[2] W. G. Aref. *Query Processing and Optimization in Spatial Databases.* PhD thesis, University of Maryland, College Park, MD, July 1993. Also available as Technical Report CS-3097, University of Maryland, College Park, MD, 1993.

[3] W. G. Aref and H. Samet. Optimization strategies for spatial query processing. In *Proceedings of the 17th International Conference on Very Large Databases (VLDB)*, pages 81–90, Barcelona, Spain, September 1991.

[4] W. G. Aref and H. Samet. Uniquely reporting spatial objects: Yet another operation for comparing spatial data structures. In *Proceedings of the 5th International Symposium on Spatial Data Handling*, pages 178–189, Charleston, SC, August 1992.

[5] W. G. Aref and H. Samet. Duplicate elimination using proximity in spatial databases. Technical Report CS-3067, University of Maryland, College Park, MD, May 1993.

[6] L. Becker and R. H. Güting. Rule-based optimization and query processing in an extensible geometric database system. *ACM Transactions on Database Systems*, 17(2):247–303, June 1992.

[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] A. U. Frank. Properties of geographic data: Requirements for spatial access methods. In O. Günther and H. J. Schek, editors, *Advances in Spatial Databases - 2nd Symposium, SSD'91. Also Lecture Notes in Computer Science 525*, pages 225–234. Springer-Verlag, Berlin, 1991.

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

[10] 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.

[11] R. H. Güting. Gral: An extensible relational system for geometric applications. In *Proceedings of the 15th International Conference on Very Large Databases (VLDB)*, pages 33–44, Amsterdam, August 1989.

[12] 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.

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

[14] 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.

[15] Richard J. Lipton and Jeffrey F. Naughton. Query size estimation by adaptive sampling. In *Proceedings of the 9th. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Nashville, TN, April 1990.

[16] 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.

[17] G.M. Morton. A computer oriented geodetic data base, and a new technique in file sequencing. Technical Report (Unpublished), IBM Ltd., Ottawa, Canada, 1966.

[18] 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.

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

[20] B. C. Ooi. *Efficient Query Processing for Geographic Information Systems.* PhD thesis, Monash University, Victoria, Australia, 1988. (Lecture Notes in Computer Science 471, Springer-Verlag, Berlin, 1990).

[21] B. C. Ooi and R. Sacks-Davis. Query optimization in an extended DBMS. In W. Litwin and H.-J. Schek, editors, *Foundations of Data Organization and Algorithms*, pages 48–63, Berlin, 1989. Lecture Notes in Computer Science 367. Springer-Verlag.

[22] 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.

[23] 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.

[24] R. Sacks-Davis, K. J. McDonell, and B. C. Ooi. GEOQL - A query language for geographic information systems. Technical Report 87/2, Monash University, Victoria, Australia, July 1987.

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

[26] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, pages 23–34, Boston, MA, June 1979.

[27] A. Wolf. The DASDBS GEO-Kernel: Concepts, experiences, and the second step. In A. Buchmann, O. Günther, T. R. Smith, and Y.-F. Wang, editors, *Design and Implementation of Large Spatial Databases, Proceedings of the First Symposium SSD'89. Also Lecture Notes in Computer Science 409*, pages 67–88. Springer-Verlag, Berlin, 1990.

[28] E. Wong and K. Youssefi. Decomposition - A strategy for query processing. *ACM Transactions on Database Systems*, 1(3):223–241, September 1976.

8 Appendix

In this Appendix, we give a detailed derivation of the following formula that was used in Section 3.1 to compute the average area of intersection between two rectangles:

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

where w and o are the corresponding rectangles; A_w, X_w, and Y_w are the area, width and height, respectively, of w, and A_o, X_o, and Y_o are the area, width and height, respectively, of o. Also, let p be the point corresponding to the upper-left corner of w, and (x_p, y_p) be the coordinate values of p. In order to compute A_{ow}, (refer to Figure 10), we observe that p is equally likely to be at any location inside the rectangle $[x_p, x_p + X_w + X_o] \times [y_p, y_p + Y_w + Y_o]$, i.e., with probability $\frac{1}{(X_w+X_o)(Y_w+Y_o)}$. We use the following approach. First, we compute the area of intersection when p is at some arbitrary location (x, y) and multiply it by its probability of occurrence at this location. Next, we integrate the area over all possible values of x and y inside the rectangle $[x_p, x_p + X_w + X_o] \times [y_p, y_p + Y_w + Y_o]$. From Figure 10, the upper left corner of w must lie somewhere inside the shaded area so that w can possibly intersect o. For simplicity in developing the formulas, we assume that the origin of the space is located at point C. We also assume that o is larger than w. If this happens to be false, then the roles of o and w can be exchanged. This is possible since we are interested in their common area of intersection. To compute A_{ow}, we divide this shaded area into four region classes 1, 2, 3, and 4. We will develop the formulas for each region class separately as there is a difference in the way the area of intersection is computed in each case.

1. If p lies anywhere in regions of class 1, then the area of intersection (A_1) is computed as follows: $A_1 = xy$, where x and y range from $0 \cdots X_w$ and $0 \cdots Y_w$, respectively.

2. If p lies anywhere in regions of class 2, then the area of intersection (A_2) is computed as follows: $A_2 = X_w y$, where x and y range from $X_w \cdots X_o$ and $0 \cdots Y_w$, respectively.

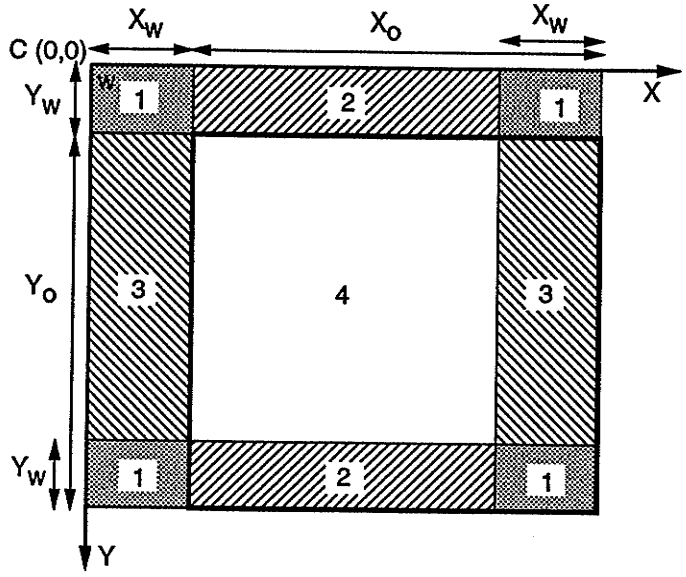


Figure 10: Computing the average area of intersection between two rectangles.

3. If p lies anywhere in regions of class 3, then the area of intersection (A_3) is computed as follows: $A_3 = xY_w$, where x and y range from $0 \cdots X_w$ and $Y_w \cdots Y_o$, respectively.

4. If p lies anywhere in regions of class 4, then the area of intersection (A_4) is computed as follows: $A_1 = X_w Y_w$, where x and y range from $X_w \cdots X_o$ and $Y_w \cdots Y_o$, respectively.

Now, in order to compute A_{ow}, we integrate each of $A_1 \cdots A_4$ over the ranges above and divide by the probability of occurance at the given locations. As mentioned above, p is equally likely to be anywhere in the shaded region with probability $\frac{1}{(X_w+X_o)(Y_w+Y_o)}$. Therefore, the average area of intersection A_{ow} can be computed as follows:

$$A_{ow} = \frac{1}{(X_w + X_o)(Y_w + Y_o)}(4\int_0^{X_w}\int_0^{Y_w} A_1 + 2\int_{X_w}^{X_o}\int_0^{Y_w} A_2 + 2\int_0^{X_w}\int_{Y_w}^{Y_o} A_3 + \int_{X_w}^{X_o}\int_{Y_w}^{Y_o} A_4)$$

Notice that the multiplicative factors are due to the number of times each region class exists, i.e., there are four instances of class 1, two instances of classes 2 and 3, and one instance of class 4. By integration and simple cancellation of terms, we get the final formula for A_{ow}.