Selectivity Estimation in Spatial Databases

Swarup Acharya **Viswanath Poosala** **Sridhar Ramaswamy***
Information Sciences Research Center
Bell Laboratories, Lucent Technologies
600, Mountain Avenue, Murray Hill, NJ, USA
{swarup,poosala}@research.bell-labs.com, {sridhar}@epiphany.com

Abstract

Selectivity estimation of queries is an important and well-studied problem in relational database systems. In this paper, we examine selectivity estimation in the context of Geographic Information Systems, which manage spatial data such as points, lines, poly-lines and polygons. In particular, we focus on point and range queries over two-dimensional rectangular data. We propose several techniques based on using spatial indices, histograms, binary space partitionings (BSPs), and the novel notion of spatial skew. Our techniques carefully partition the input rectangles into subsets and approximate each partition accurately. We present a detailed experimental study comparing the proposed techniques and the best known sampling and parametric techniques. We evaluate them using synthetic as well as real-life TIGER datasets. Based on our experiments, we identify a BSP based partitioning that we call *Min-Skew* which consistently provides the most accurate selectivity estimates for spatial queries. The Min-Skew partitioning can be constructed efficiently, occupies very little space, and provides accurate selectivity estimates over a broad range of spatial queries.

1 Introduction

Geographic Information Systems (GISs) have generated enormous interest in the commercial and research database communities over the last decade. GISs typically store and manage spatial data such as points, lines, poly-lines, polygons, and surfaces and hence are often referred to as spatial databases. Several commercial database systems that manage spatial data are now available. These include ESRI's ARC/INFO [ARC93], InterGraph's MGE [Int97], MapInfo [Map98] and Informix [Ube94]. Most other leading database vendors either offer some kind of support for spatial data or are in the process of providing such support. GISs have also been the focus of much research, mostly towards efficient manipulation and access[Sam89a, Sam89b] and more recently, towards research prototypes[DeW94, GRSS97].

As in relational database systems, there are many modules of a spatial database system that require accurate estimates of query result sizes. Such estimates are used in a variety of ways. For example, query optimizers use query result size estimates to determine the most efficient way to execute queries [SAC$^+$79]. Estimates are also used by database systems to give users feedback about the running times of their queries before the queries are actually executed. Since it is impractical to run the entire query to compute the result sizes, most commercial systems use some form of statistics to approximate the underlying data and estimate result sizes based on these statistics.

A variety of techniques have been proposed in the literature to compute estimates of query result sizes in *relational* databases. The most common ones use *histograms*[Koo80, Poo97], *samples*[LNS90], or are based on *parametric techniques* that model the data via a standard mathematical distribution[CR94, BF95]. Of the various techniques, histograms, in particular, have proved very popular in database systems because they can be computed efficiently, use very little space (on the order of a few hundred bytes per relation), and do not require that the input distribution be known in advance.[a] They work by partitioning the input into a small number of subsets called "buckets", and by using approximations for each bucket to model the distribution of tuples within. Query result estimates are then obtained by processing the query against the buckets and the approximations used therein.

The problem of selectivity estimation for spatial data is very different from the relational selectivity

*Work done while the author was at Bell Labs. Current affiliation is Epiphany Inc., 2300 Geng Road, Suite 200, Palo Alto CA 94303.

[a]However, one can construct distributions that cannot be approximated well using histograms in a small amount of space.

Appears in the Proceedings of ACM SIGMOD Conference, Philadelphia, June 1999.

estimation problems that have been studied extensively in the database literature. Most previous works have focused on approximating the distributions of single numerical attributes. Even the ones studying multi-dimensional data [PI97, MPS99] have concentrated on approximating the *frequencies* of *points* in space. Many of these techniques are aimed at data with highly skewed frequencies. However, they do not perform as well when the frequency domain is relatively uniform but the value domain (i.e., placement of points in space) is skewed.

Spatial selectivity estimation differs in two important aspects from traditional selectivity estimation: (i) The individual spatial entities may differ in shape and size; (ii) The distribution of frequencies over the input domain does not vary dramatically in spatial data, whereas the values are spread non-uniformly in space. (This corresponds to the fact that not many spatial entities cover the same point.) Hence, the problem of approximating spatial data requires techniques for accurately approximating the value domain of a distribution. Even in the context of uni-dimensional point data, this has been known to be one of the most difficult problems in selectivity estimation [HNSS95, GG82, Poo97]. To the best of our knowledge, only Belussi and Faloutsos [BF95] have addressed the problem of spatial selectivity estimation and they concentrate only on point data (issue (ii) above).

Motivated by the above reasoning, we study the problem of selectivity estimation of point and range selection predicates over spatial data. In particular, we concentrate on *two-dimensional rectangular* data. This is an important problem because it is customary in spatial database systems to approximate spatial objects using their *minimum bounding rectangles(MBRs)* and perform query processing with the MBRs as much as possible. (Spatial objects themselves often need such large representations that it becomes very cumbersome and inefficient to manipulate them directly.) Though we concentrate primarily on rectangular data in this paper, our techniques are applicable to point and linear data as well.

Our contributions are as follows:

- We present several novel *grouping* techniques for approximating spatial data. In addition to analogues of techniques used in relational databases, we propose new techniques based on using spatial indexes for selectivity estimation. We also propose novel techniques based on the notions of *spatial density* and *spatial skew*. These features of the data capture the underlying input data distribution in a concise manner and allows us to devise accurate techniques for selectivity estimation.
- We provide a detailed experimental study comparing the various techniques on both synthetic and real-life datasets. Our results show that a density-based technique that we call *Min-Skew* outperforms the other techniques over the entire data and query spectra. The technique is not only computationally efficient but also has low memory requirements even for large dataset sizes. Our studies also show that straightforward application of techniques from the one-dimensional world, such as sampling, are not effective in the spatial domain.
- We also identify a seemingly counter-intuitive fact in spatial selectivity estimation: using the data at too fine a level of detail sometimes result in poor performance when answering large queries! We use this observation to develop a technique called *progressive spatial refinement* to improve *Min-Skew* further.

The rest of this paper is organized as follows. Section 2 contains a formal problem description. Section 3 offers a high-level discussion of possible solutions to the problem and presents some basic techniques for spatial selectivity estimation. Section 4 presents more sophisticated techniques for spatial selectivity estimation. Section 5 presents the results from an extensive series of experiments on real-life and synthetic data. We identify *Min-Skew* as the clear winner and suggest further improvements to it. We present our conclusions in Section 6.

2 Problem Formulation

In this section, we formally define the specific class of selectivity estimation problems addressed in this paper and develop a notation for describing spatial data.

Consider a relation R containing an attribute A whose domain is the set of (two-dimensional) rectangles. The *distribution* $\mathcal{T}$ of A is the set of rectangles $\{r_1, r_2, \ldots, r_N\}$ where $r_i = [(x_i^1, y_i^1), (x_i^2, y_i^2)]$. The two components of rectangle r_i specify its lower-left and upper-right corners respectively. The x_i's and y_i's are assumed to come from an integer or real domain. We denote the area of the minimum bounding rectangle (MBR) of the input rectangles in $\mathcal{T}$ by $Area(\mathcal{T})$. W_{avg} and H_{avg} denote the *average* width and height of the rectangles contained in $\mathcal{T}$ respectively. Finally, denote the sum of areas of all the rectangles in $\mathcal{T}$ by TA.

A range query predicate is specified as a rectangle $Q = [(qx^1, qy^1), (qx^2, qy^2)]$. Note that this definition can be used to specify point queries as well by setting $qx^1 = qx^2$ and $qy^1 = qy^2$. The *result size* $|Q|$ is the number of rectangles in the input that have a non-empty intersection with the query rectangle. The *selectivity* of query Q is the fraction $|Q|/N$. Computing the exact selectivity of a query requires us to either scan the input dataset or to use an index to find the number of input rectangles that intersect with the query. Clearly, both of these options are too expensive to be useful in most contexts (including query optimization), because they may require several disk accesses and involve a significant amount of processing. Hence, one is forced to summarize or approximate the input data so that it can be represented compactly. The approximations are then used to estimate selectivities. We define the problem of selectivity estimation for rectangular data to be that of estimating $|Q|$ for an arbitrary range query Q given an input distribution $\mathcal{T}$. This problem is the focus of this paper.

3 Techniques for Approximating Spatial Data

In this section, we examine various techniques for approximating spatial data. Our main focus is on the broad class of *non-parametric* techniques which approximate the data using a small amount of space, typically in tabular form. First we present a very simple technique that is analogous to the traditional *uniform distribution assumption* made in relational databases [SAC+79].

3.1 Uniformity Assumption for Spatial Data

The simplest technique for approximating spatial data is to assume that the input rectangles are of identical width and height, and are distributed uniformly in the input space. The formulas for estimating the result sizes of point and range queries under this assumption are given below.

Point Queries: For point queries, one has to return the number of rectangles that overlap a particular point. One applies the uniformity assumption by calculating the *average* number of rectangles hit by a point query within the input MBR. The average here is over all the point queries within the MBR. It is easily shown that this average can be obtained by taking the ratio of the total area of the all the input rectangles to the area of the input's MBR ($\frac{TA}{Area(\mathcal{T})}$).

Range Queries: For range queries, the natural uniformity assumption is that rectangles are of identical width and height uniformly distributed within the input MBR. The number of rectangles that a query $Q = [(qx^1, qy^1), (qx^2, qy^2)]$ intersects can be calculated as follows: Define $qx'^1 = \min(x^1_{\mathcal{T}}, (qx^1 - W_{\text{avg}}))$. This extends the left side of the query by the average width subject to the constraint that the left side cannot cross the left input boundary. Define qx'^2, qy'^1 and qy'^2 similarly. Then, the extended area of the query, $Area(Q)$ is: $(qx'^2 - qx'^1) \times (qy'^2 - qy'^1)$. The number of rectangles intersecting with Q is then estimated as $n \times Area(Q)/Area(\mathcal{T})$. Note that simply using the area of the query Q without extending it is inaccurate because this assumes that only rectangles whose centers lie in the query area intersect with the query. That does not account for input rectangles with centers outside the query area that intersect the query.

Of course, in real-life data, there is often considerable skew in the placement of input rectangles in the input space. There is also skew in the size of the input rectangles[b]. Hence, the application of the uniformity assumption can result in highly inaccurate selectivity estimates.

At a high level, both placement skew and size skew problems can be mitigated by "grouping" the input rectangles into smaller subsets or *buckets* and then applying the uniformity assumption to each subset individually. If, as a result of the grouping, the skew within each subset is small, it is clear that making the uniformity assumption over the subsets incurs substantially less error than making the uniformity assumption over the entire input. We therefore focus on developing accurate grouping techniques for spatial data.

3.2 Approximating Spatial Data by Grouping

There are two issues to be addressed in developing a grouping-based technique[c] for approximating spatial data: a) the criteria for grouping rectangles into buckets and b) the technique for using the resulting set of buckets to estimate the result sizes. The latter is solved by observing that, once the buckets are identified, the problem of selectivity estimation reduces to solving selectivity estimation over the individual buckets (because the buckets are disjoint). We apply the uniformity assumption (and the corresponding formulae developed in Section 3.1) individually to each bucket.

The primary issue in developing a grouping based technique, therefore, is the criterion for grouping rectangles into buckets. We have identified the following three classes of fundamentally different groupings that can be applied to approximate spatial data.

- *Equi Partitionings:* In these partitionings, the goal is to partition the input space such that the resulting buckets are identical under some measure: e.g., they all have equal areas or equal number of rectangles.
- *Index Partitionings:* These partitionings are produced by a spatial index structure like the R-tree.
- *Skew-Aware Partitionings:* These partitionings, which are more sophisticated, use binary space partitionings (BSPs) and the notion of *spatial skew* (defined

[b]However, placement skew tends to dominate size skew in real-life datasets.

[c]Throughout the paper, we use the terms *grouping* and *partitioning* interchangeably.

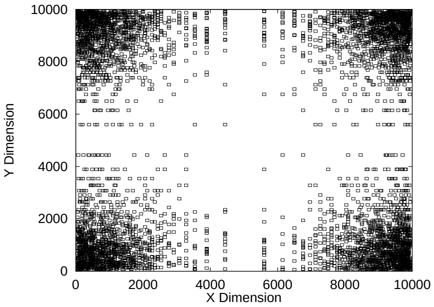


Figure 1: *Charminar* Dataset

in Section 4) to group rectangles into buckets such that the negative effects of making the uniformity assumption are minimized.

Next, we present the equi-partitioning and index-partitioning techniques and identify their advantages and drawbacks. We then present the fundamentally different skew-aware approach.

3.3 Equi-Partitioning of Spatial Data

In these partitionings, the goal is to partition the input space such that some property is identical across the resulting partitions. We studied the following two criteria for equi-partitionings, namely *Equi-Area* and *Equi-Count*, which are analogous to the *Equi-Width* and *Equi-Height* histograms in relational data approximations [Koo80, PSC84].

***Equi-Area* Groupings:** *The goal of the* Equi-Area *grouping is to create buckets whose MBRs have the same area.* This approach clearly minimizes the maximum area among the buckets. Since large buckets potentially incur higher errors (though this is not always true), the *Equi-Area* groupings can be seen as an attempt to minimize the worst case errors. We construct the partitioning by starting with a single bucket consisting of the MBR of all the input rectangles. The MBR of the bucket is split along the longer dimension into two equal halves. Rectangles are grouped into the two halves based on where their centers lie. MBRs are calculated for the two new buckets and once again the longest dimension (among the four choices available now) is chosen and the corresponding bucket split. The process is repeated until the desired number of buckets are obtained. The recalculation of the MBRs ensures that the buckets produced try to approximate the input data distribution rather than simply sub-divide the MBR of the whole input into regions of equal size. This is useful when there is a lot of empty space in the MBR of the input.

***Equi-Count* Groupings:** *In an* Equi-Count *grouping, the goal is to create buckets containing the same number of rectangles.* Since buckets with a large number of rectangles potentially incur large errors (this is not always true either), the *Equi-Count* groupings can also be seen as an attempt to bound the worst case errors. The algorithm for obtaining this grouping is similar to the algorithm for *Equi-Area* with one difference: the dimension with the highest *projected rectangle count* is chosen for splitting. The projected rectangle count of a dimension d in bucket B is the number of distinct centers of all the rectangles in the bucket when projected on dimension d.

The above partitionings are illustrated in the following example.

Example 1: Consider the dataset depicted in Figure 1. This dataset, which we call the *Charminar* set[d], is described in more detail in Section 5 as part of our experiments. Note that this set contains more rectangles in the corners than in the center. Figures 2 and 3 depict the *Equi-Area* and *Equi-Count* groupings on this dataset, with each grouping using 50 buckets. As expected, *Equi-Area* has nearly identical buckets distributed more or less uniformly, whereas *Equi-Count* contains more buckets in the "denser" areas. ∎

3.4 R-tree Index Based Grouping

The R-tree[Gut85, SRF87, BKSS90] is a popular index structure for spatial data. While building an index structure, an R-tree insert algorithm tries to minimize one or more of the area, margin, and/or the overlap, etc., of the summed over the bounding boxes of the internal nodes. It stands to reason that partitions produced by R-trees can be used to summarize the input data well using the MBRs of the internal nodes, effectively giving us another way to construct a spatial partitioning (The idea of using indexes to obtain summary information about a dataset has been examined by other researchers before. See [Ant92, Aok97].) We used the R*-tree[BKSS90], which is known to be one of the most efficient members of the R-tree family of data structures. The partitioning resulting from using the R-tree on the *Charminar* dataset is given in Figure 4. Note that this partitioning is drastically different from those given by the Equi-Partitioning techniques.

3.5 Disadvantages of Equi-Partitioning and Index-Based Grouping Schemes

The equi-partitioning and index-based schemes suffer from fundamental disadvantages. These disadvantages can be broadly classified based on how they affect their accuracy and computation cost.

Accuracy: The key drawback of the equi-partitioning schemes is that they often produce buckets that have significant placement and size skew within. The *Equi-Area*

[d] The name is derived from a famous structure in India called the *Charminar* which has four minarets (pillars) in its corners.

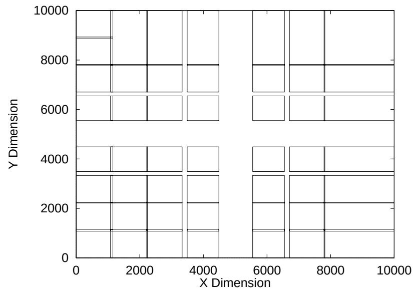


Figure 2: *Equi-Area* Partitioning

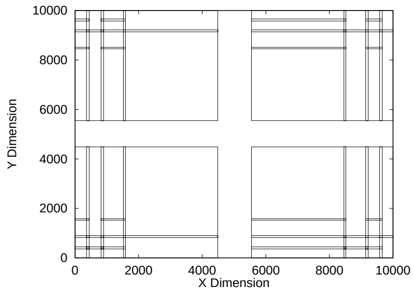


Figure 3: *Equi-Count* Partitioning

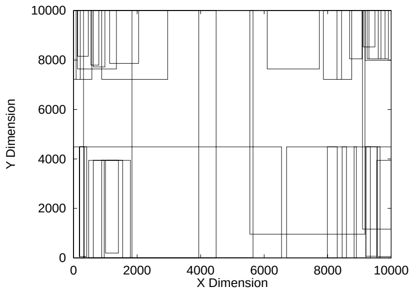


Figure 4: *R-Tree* Partitioning

grouping does not at all consider skew in the distribution of input rectangles when producing buckets. The *Equi-Count* grouping does attack skew by producing more buckets in regions where there are a lot of rectangles. However, if the region is relatively uniform, this is not beneficial and wastes buckets. R-tree insertion algorithms essentially make local decisions about creating new nodes, with the result that the final buckets produced are often fairly skewed. However, recent proposals to minimize the number of disk reads performed by the R-tree by taking the data distribution into account can be expected to produce partitions which are more conducive to selectivity estimation [TS96].

Computation: A further disadvantage of the three grouping schemes above is that they are computationally expensive. The equi-partitionings require that all the input data be kept in memory. (They can be modified to use less memory, but they still make several passes over the input data.) The R-tree technique does not require the index and data to fit in memory, but still makes the equivalent of several passes over the input data. A naive algorithm for constructing an R-tree based on repeated insertion will take $O(N \log_B N)$ I/O's to insert N items while a more sophisticated bulk-loading algorithm will take $O(\frac{N}{B} \log_B N)$ I/O's. (Here, B is the disk block size.)

Motivated by the deficiencies of these techniques, we developed a new class of techniques that directly address skew and computational efficiency. We describe these techniques in the next section.

4 Advanced Spatial Grouping Techniques

Any technique for addressing the skew problem has to focus on reducing placement and size skew within a bucket. Since we make the uniformity assumption within a bucket, the more uniform the intra-bucket distribution, the more accurate the estimation will be. We formalize this notion as follows.

Define the *spatial density* of any point in the space to be *the number of rectangles containing that point*. (Note that the uniform distribution assumption essentially replaces all the spatial densities within a bucket by a single number, equal to their average.) Then, it is clear that an accurate approximation groups rectangles such that all points in a bucket have similar spatial densities. Similar to the notion of V-Optimality in histogram literature [PIHS96], we define the following metric to capture this notion of grouping error.

Definition 4.1 *Consider a grouping $\mathcal{G}$ with buckets $B_i, 1 \leq i \leq \beta$. Let n_i be the number of points in B_i. The* spatial-skew *s_i of a bucket B_i is the statistical* variance[e] *of the spatial densities of all points grouped within that bucket. The* spatial-skew *S of the entire grouping is the weighted sum of spatial-skews of all the buckets: $\sum_1^\beta n_i \times s_i$.* ∎

It is clear that a partitioning with small spatial-skew is likely to be highly accurate in approximating the given data. Unfortunately, building *optimal* partitionings, which *minimize the spatial-skew using a given amount of space*, is a difficult problem that is provably NP-hard for even simple instances [MPS99, KMS97]. One technique for reducing the complexity of constructing good partitionings is to restrict ourselves to *binary space partitionings (BSP)*. The partitioning of a region is said to be a BSP if we can find a vertical or horizontal line that divides the input region into two sub-regions such that the partitionings of the two sub-regions are also BSPs.

The best known algorithms for constructing BSPs use dynamic programming and have a complexity of at least $O(N^{2.5})$ [MPS99] and also require the input to be in memory. Clearly this is infeasible for large GIS data. To make the problem of finding good BSPs tractable, we propose the following two heuristics:

1. Use a compact approximation of the input data in place of the original in order to build the grouping in memory.

[e] variance v of a set of numbers $f_1, f_2, .., f_n$ is equal to $\frac{\sum_{i=1}^{N}(f_i - \bar{f})^2}{N}$, here $\bar{f}$ is the average of all f_is.

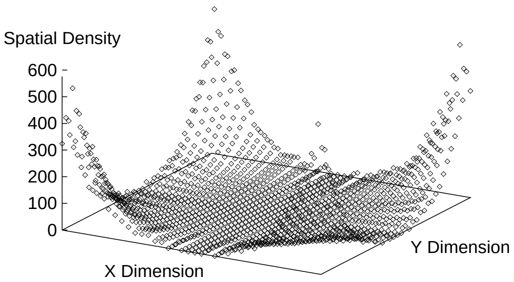


Figure 5: Spatial Densities in *Charminar*

2. Use a greedy approach and reduce the complexity of the computation by making locally optimal decisions, thus achieving linear time complexity for the partitioning.

In the rest of this sub-section, we focus on the problem of reducing the effective input size. In the next section, we develop a novel algorithm for constructing good BSPs based on the above two points.

One approach to reduce the input size is to divide the input MBR evenly into a uniform grid of *rectangular regions*. Each grid region is associated with its *spatial density*, the number of input rectangles that intersect with it. We then use the grid regions and their spatial densities as inputs to a BSP construction algorithm. The grid size is chosen such that the all the grid regions and the spatial densities can easily fit in memory. Note that the spatial densities can be obtained easily in a single sweep of the input data. We illustrate these concepts in the following example.

Example 2: Consider an approximation of the *Charminar* dataset of Figure 1 using a 50×50 grid. Figure 5 depicts the spatial densities of these regions along the z-dimension. ∎

Choosing the correct granularity for the grid of rectangular regions presents a tradeoff. On one hand, a fine grid captures the details of the underlying distribution and lets us approximate it well. On the other hand, a fine grid implies a higher processing and memory cost. We examine this issue empirically in Section 5 and show that in certain situations, a very fine grid can actually be detrimental to the construction of a good partition.

4.1 The Min-Skew Partitioning

We now propose a novel technique for constructing skew-resistant binary space partitionings. This technique uses a uniform grid of regions and their spatial densities as input. The algorithm partitions the grid into buckets while trying to minimize the spatial-skew (Definition 4.1) of the grouping. We call the resulting partitioning the *Min-Skew Partitioning*.

The construction algorithm for the Min-Skew partitioning repeatedly partitions the given set of regions such that the spatial-skew is minimized at each step. Since it always partitions an existing region into two, the result is a BSP partitioning. The pseudo-code for the algorithm is below:

> **Algorithm Min-Skew**
> Compute the aggregate density of the input rectangle distribution by using a uniform grid of rectangles and their spatial density.
> Start with a single bucket consisting of all the regions.
> **while** there are less buckets than needed
>
> > For each current bucket do
> >
> > > Compute the spatial skew of the bucket (Definition 4.1) and the split point along its dimensions that will produce the maximum reduction in spatial-skew.
> >
> > Pick the bucket whose split will lead to the greatest reduction in spatial-skew. Split the bucket into two and assign regions from the old buckets into the new buckets.
>
> **endwhile**
> Assign each rectangle in the input to the bucket whose MBR contains the center of the rectangle.

In our implementation, we further reduce the computational complexity at each step of the greedy approach by basing the splitting decisions on marginal frequency distributions along each dimension rather than the full two-dimensional input distribution. Figure 6 illustrates the preprocessing of the spatial inputs and highlights one iteration of the algorithm. Also, the 50-bucket partitioning of the *Charminar* data set is also given in Figure 7.

The Min-Skew partitioning has many desirable features. Since the construction algorithm tries to minimize spatial-skew (albeit in a local, uni-dimensional manner), the partitioning tends to have significantly less skew overall than other algorithms. Further, the construction algorithm does not require the entire data distribution to fit in main memory, which is a significant advantage. Thus it addresses both the issues of skew and computation time that were the disadvantages of the grouping techniques considered in the last section.

5 Experimental Results

In this section, we study the performance of the various techniques in estimating spatial selectivity. The results

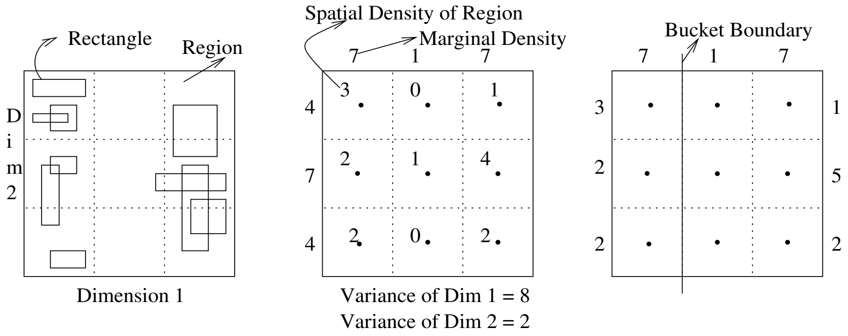


Figure 6: Min-Skew Illustration

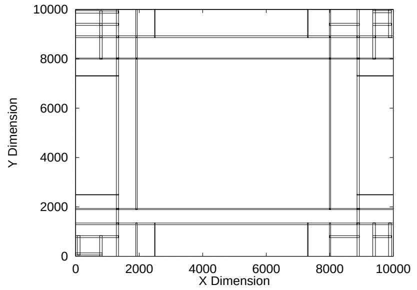


Figure 7: Min-Skew Partitioning for the *Charminar* Dataset

presented in this section are based on an extensive experimental study on both real-life and synthetic spatial data.

This section is organized as follows. We first describe the datasets and the query model. This is followed by the performance evaluation of the various techniques. Finally, we explore the champion technique, *Min-Skew*, in greater detail and study some of its tradeoffs.

In comparing the various techniques, we used the following metrics:

- **Average Relative Error**: This is the ratio of the error in an estimate to the actual size of the query result averaged over a set of queries. If e_i is the estimated answer and r_i is the actual answer for a given query q_i, the average relative error[f] for a query set Q is given as: $(\sum_{q_i \in Q} |r_i - e_i|)/(\sum_{q_i \in Q} r_i)$.
- **Preprocessing Time**: This is the time taken by the construction algorithm for each technique to preprocess the data, build appropriate data structures, and generate the buckets used to process spatial queries.

5.1 Datasets

We studied the performance of the various techniques on both real life and synthetic datasets, which are described below.

5.1.1 Real-Life Datasets

For real-life data, we used two datasets widely used in spatial database research: TIGER [tig92] and the Sequoia dataset [SFGM93]. In this paper, due to space constraints, we only present results from the *NJ Road* dataset from TIGER, which gives the road data for the state of New Jersey as line segments. For our experiments, we computed the bounding boxes of all the line segments ($414,442$ in this set) and used them as the spatial inputs. The results using the other data sets are available in the full paper [APR99].

[f]This metric is undefined if all queries in the query set produce no output. However, this is not a case we encounter in our datasets.

5.1.2 Synthetic Datasets

We systematically generated several synthetic datasets varying in size, sparsity, placement skew, and size skew. Sparsity was controlled by adjusting the dataset size relative to the total input area. Size skew was modeled by generating widths and heights from the Zipf Distribution [Zip49]. Placement skew was modeled using two-dimensional Zipf distributions. In this paper, we present results from one set, the *Charminar* set, which was introduced in Section 3.3 (Figure 1). It contains 40000 rectangles of identical height and width of 100 units distributed in a 10000×10000 space. As can be seen, most of the rectangles are concentrated in the four corners creating areas of varying levels of spatial densities.

5.2 Query Sets

The query sets consist of a large number (10000) of rectangles lying within the MBR of the input. The centers of the rectangles were chosen randomly from the set of centers of the input rectangles. The average width (height) of the query rectangle (referred to as parameter `QSize` in the experiments) was varied from 2% to 25% of the width (height) of the input bounding box, which varies the query area from 0.04% to 6.25% of the input MBR size. A desired average area, a, for the query rectangles generated is achieved by setting the height and width of the rectangles to be uniformly distributed in the range $[0.5 \times \sqrt{a}, 1.5 \times \sqrt{a}]$.

5.3 Techniques Studied

In addition to studying the four new techniques that we propose, namely, *Equi-Count*, *Equi-Area*, *R-Tree* and *Min-Skew*, we also included sampling and the parametric technique due to Belussi and Faloutsos [BF95] in our experiments.

We apply sampling as follows. We collect a sample of the input rectangles. Given a query, we compute the selectivity of the query on the sample. We then scale the result appropriately to obtain an estimate for the query selectivity. That is, if the size of the sample if n, the input size is N, and the number of sample rectangles that satisfy the given predicate is m, then the estimated result size is $m \times \frac{N}{n}$. This technique is referred to as *Sample* in the graphs.

The fractal-based parametric technique in [BF95] was proposed for point data only. The paper shows that spatial data can be described using fractals having a non-integer fractal dimension. In that context, selectivity for such point sets can be described using a power law with the correlation fractal dimension as the exponent. For comparison, we extended this technique to rectangle data by using the centroids of the rectangles as representatives.

In addition to these techniques, we also used the uniform assumption (Section 3.1) over the entire input, i.e., a single bucket approximation and call it the *Uniform* technique.

5.4 Space Allocation

The key parameter that influences the cost and performance of the non-parametric techniques is the number of buckets (or samples) allowed in the approximation. Typically, query processors allocate a few hundred bytes for statistics on each attribute (which translates into having 50 to 200 buckets). In this study, we consider allocating between 50 and 750 buckets.

The space overhead of each of the bucket-based techniques (*Equi-Count*, *Equi-Area*, *R-Tree* and *Min-Skew*) is eight times the number of buckets — four words for the bounding box of the bucket and the average density of the bucket, and the number, the average width and the average height of the rectangles in the bucket. The *Sample* technique requires half that since it needs to only store the bounding box of each sample rectangle. Consequently, in terms of space overhead, $2n$ rectangles for the *Sample* technique correspond to n buckets for the bucket-based techniques. However, in the following experiments, we liberally give *Sample* twice the fair amount. In other words, when *Sample* is compared to the other techniques, it is given twice the space that is given to the other techniques (and thus, four times as many rectangles as buckets).

A complication with using an R-tree to build partitions is the difficulty in controlling the number of buckets produced by it. We addressed this problem by tweaking the branching factor to produce close to the number we desired but ensuring we never exceeded the allocated quota (in order to be fair to the other techniques).

5.5 Experimental Results

In this section, we compare the performance of the various techniques for spatial selectivity estimation. We first study the performance of the various techniques with respect to the *query size* and the *number of buckets*. In each experiment, we vary one of the parameters over its entire set of values and keep the remaining parameters fixed at their default values. For these experiments, the number of regions used by the *Min-Skew* construction algorithm was set to $10,000$.

5.5.1 Experiment 1: Impact of Query Size

This experiment studies the performance of the various techniques for different query sizes (on x-axis). Figure 8 shows relative error as a function of query size when using 100 buckets to approximate the *NJ Road* dataset. The results for the other datasets are qualitatively similar. Here, we chop the y-axis in both the graphs to keep them in scale. The error for the *Sample* technique is 82% for 2% `QSize`.

The general observation from the graph is that the relative error decreases with increasing query size. This is because the error in an estimation arises only from those buckets that are partly contained in the query. Since each bucket stores the exact number of rectangles belonging to it, those buckets that fall entirely within the query rectangle contribute no error. Since more buckets are fully covered by bigger queries, *relatively* fewer buckets contribute to the error. Thus, all the techniques show better accuracy as we move to the right of the graph.

The fractal technique of [BF95] was close to being the least effective technique for all the datasets that we studied. Average relative error for the *NJ Road* dataset was consistently close to 90% for most query sizes. It must be said in defense of the technique that it was developed for point data, while we have examined it for rectangle data. In the rest of the experiments, we do not show the performance of the fractal technique since it did not perform competitively.

The numbers for *Uniform* were also very poor. For the *NJ Road* dataset, *Uniform* had errors in the range 80% (`QSize` = 2%)—57% (`QSize` = 25%). This high range of errors was observed across all the data sets. This shows that real-life spatial data is inherently skewed and thus, cannot be captured by a trivial single bucket approximation. In the rest of the experiments, we do not show the performance of the *Uniform* technique as well.

The graph also shows that sampling performs quite poorly. This result is borne out for all the cases studied. This is because it makes an implicit assumption that each rectangle chosen in the sample is representative of *both the spatial distribution and the size of the other rectangles* in its neighborhood (i.e., it makes a local uniformity assumption). Obviously, this is not the case for spatial data. Thus, these numbers show that approximating spatial data distributions requires more

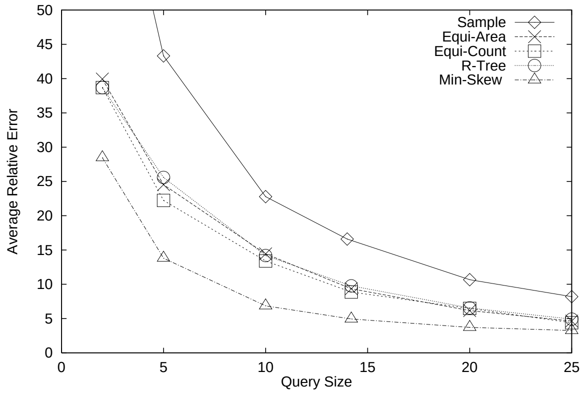


Figure 8: Performance vs. Query Size, Buckets = 100

sophisticated partitioning techniques which account for the implicit features of the data.

The *Equi-Area*, *Equi-Count* and *R-Tree* techniques have very similar error values with *Equi-Count* doing slightly better.

It is clear that, the *Min-Skew* technique is a winner by a huge margin. It improves the average relative error of its closest competitors by over 50% in most of the cases. Recall that the *Min-Skew* technique aims to create buckets that have the least variance in the density within each bucket. In other words, it explicitly accounts for the spatial skew in the data. Consequently, the error from a bucket partially covered by a query is significantly lower with the *Min-Skew* technique when compared to the other techniques that do not take skew into account.

5.5.2 Experiment 2: Impact of Bucket Size

In this experiment, we study the impact of the number of buckets on the performance of the various techniques. Figure 9 plots average relative error against the number of buckets allowed for the *NJ Road* dataset. The left and right graphs in the figure plot values for two different query sizes.

As expected, allowing more space for the approximation helps reduce the errors. The *Min-Skew* technique again gives the least errors over the entire range studied. Its performance is especially noteworthy when relatively few buckets (e.g., 50 or 100) are used. Having only a few buckets poses the greatest difficulty on any approximation technique. Performing well in this scenario is particularly good from the standpoint of query optimization, since query processors typically allow only a small number of buckets because of space constraints.

As the techniques use more buckets, they are able to approximate the input in greater detail. This lowers the effect of skew on their performance, which in turn diminishes the performance differences between the various techniques.

The performance of *Equi-Area* and *Equi-Count* are similar, with *Equi-Count* having slightly better performance in some cases. *R-Tree* is consistently worse over the entire space studied. However, the numbers for *R-Tree* are slightly pessimistic. As pointed out in earlier, it is difficult to control the number of buckets produced by the R-tree technique and thus, it often generates fewer buckets than its allocated quota. As before, *Sample* is again ineffective even when more space is available.

Based on the above experimental results, we conclude that *Min-Skew* is the technique of choice to approximate spatial distributions and the ideal technique to use for spatial selectivity estimation. As highlighted in Sec 4.1, it is fast, computationally efficient, and unlike many of the other techniques, has low memory requirements. We therefore explore the *Min-Skew* technique in greater detail and look at the various factors that affect its performance.

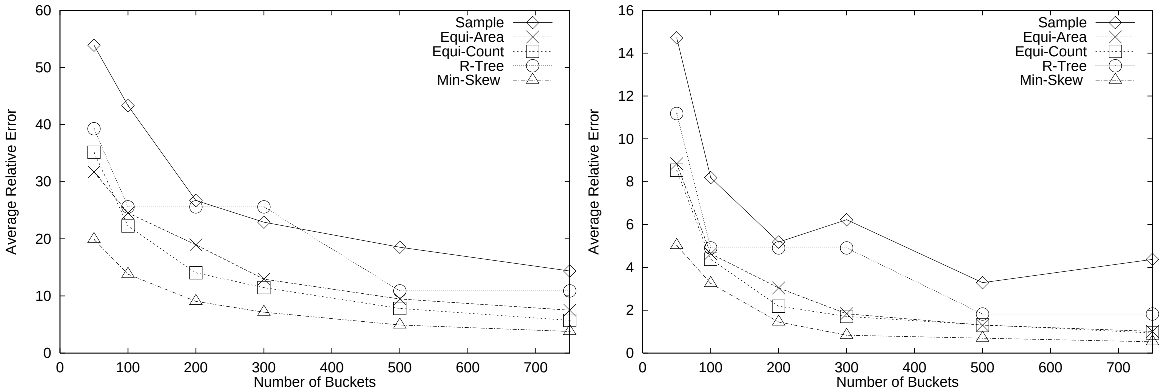
5.5.3 Experiment 3: *Min-Skew*: Sensitivity to Regions

In this experiment, we study the influence of the number of regions in the uniform grid used to approximate the underlying input space on the performance of the *Min-Skew* technique. As pointed out in Section 4, there is an inherent cost versus accuracy trade-off in choosing the correct granularity for the grid. In the following graphs, we analyze this tradeoff quantitatively.

Figure 10(a) plots average relative error versus the number of input regions for the *Min-Skew* technique. The graph is for the *NJ Road* dataset with the number of buckets set to 100. The graphs show two lines corresponding to the two query sizes in Figure 9 (5% and 25%). As we move to the right on the x-axis, the number of regions increase and the size per region drops correspondingly. This implies that the underlying input distribution is captured more accurately.

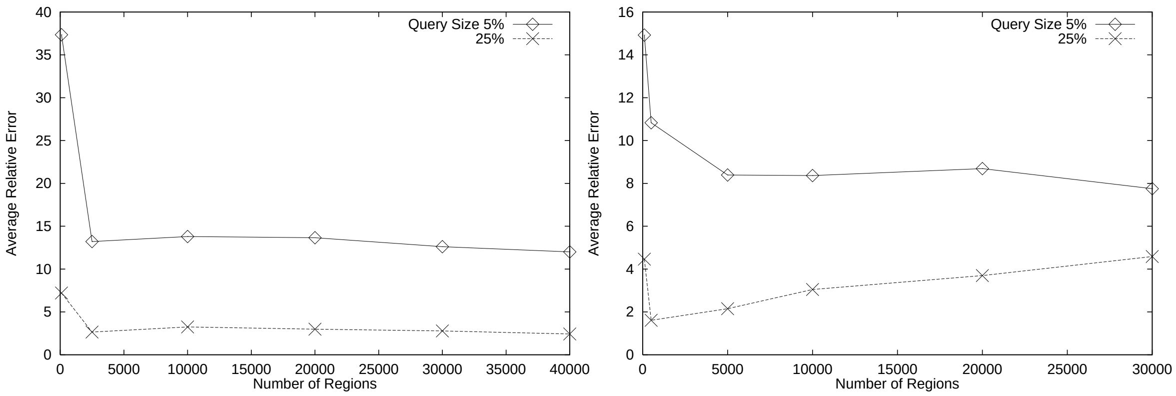
As the graphs show, increasing the number of regions decreases errors up to a point beyond which they flatten out. The flattening is due to the nature of the real-life data. They are non-uniform and have variations in density. This causes a big improvement in performance on the left side of the graph. However, since the data is not extremely skewed, further sub-dividing the space finely does not capture any additional features of the data.

These graphs show that while increasing the number of regions used to capture the original input is necessary for performance, creating too many regions does not necessarily help. Moreover, it also increases the run time overhead. Clearly, there is a correlation between the input data distribution, the query size, the number of regions and the estimation error—finding the correct



(a) Small Query (`QSize` = 5%)   (b) Large Query (`QSize` = 25%)

Figure 9: Performance vs. Number of Buckets



(a) *NJ Road* dataset   (b) *Synthetic* dataset

Figure 10: Error vs. Regions on *Min-Skew* Performance, Buckets = 100

number of regions which provides the least error is thus an interesting problem for further exploration and part of our future work.

Studying the same problem space for the *Synthetic* dataset, however, produces a very counter-intuitive result. Figure 10(b) shows the same problem space as Figure 10(a) for the *Synthetic* dataset. With increasing number of regions, performance for the small queries (`QSize` = 5%) improves as expected. However, unlike in the last set of graphs, *the error for* Min-Skew *for the large queries actually get worse with more regions*!

When a large number of regions are used to approximate the synthetic dataset, a considerable number of regions also span the skewed corner areas (Figure 5). These regions, therefore, have highly varying spatial densities. This forces the *Min-Skew* construction algorithm to allocate many buckets to those relatively compact areas in order to reduce the skew. This in turn implies that there are very few buckets for the large interior areas which are less skewed. The net result is that such a grouping is likely to perform poorly for large queries which frequently span the relatively uniform interior areas. Allocating too many buckets to the small corners, while improving performance for small queries, is unlikely to improve performance much for large queries because those areas are likely to be included wholly within a large query. Thus, having too many regions has an adverse effect on *Min-Skew* for large queries.

In the next section, we provide a solution called *Progressive Refinement* to address this problem.

5.6 Progressive Refinement for *Min-Skew*

It is clear from the discussion in the previous section that a large number of regions mainly benefit small queries. Using a smaller number of regions helps large queries. To handle both cases correctly, and have the best of both worlds, we propose *progressive refinement* of regions in *Min-Skew*. We implement progressive refinement by starting the construction algorithm with a small number of (coarse) regions. At equal intervals of buckets, we refine the regions by splitting each region into four identical regions. The newly created regions replace the original regions. Properties of the buckets needed by the construction algorithm (skew, marginal frequencies, etc.) are recalculated using the new regions. The rest of the *Min-Skew* algorithm continues as before.

The number of refinements to be used is an important parameter that we examine empirically in the next section.

Example 3: Suppose that we want to perform 2 refinements such that the final grid size is $16,000$ regions. Let the number of buckets required be 60. Since we refine the regions by a factor of four each time, we start *Min-Skew* with a grid consisting of $16000/4^2 = 1000$ regions and run it till it produces 20 $(60/(2+1))$ buckets. At that point, we refine the number of regions to $16,000/4^1 = 4000$ and produce 20 more buckets, bringing the total number upto 40. We refine the regions again, producing $16,000$ regions and produce 20 more buckets, bringing the total number up to the 60 buckets needed. ∎

On the *Charminar* dataset, progressive refinement has the following qualitative effect. Initially, the data is being observed coarsely and hence buckets are allocated to cover even the relatively less skewed middle areas. This takes care of large queries. Towards the end, a large number of regions are produced, which highlights the high skew in the four corners. This causes *Min-Skew* to allocate the remainder of the buckets to those areas. This takes care of small queries. In effect, progressive refinement allocates buckets uniformly to the entire space and then selectively drills-down and allocates more buckets to the high-skew regions which require them.

5.6.1 Experiment 4: *Min-Skew*: Impact of Progressive Refinement

In this section, we quantitatively study the accuracy improvement produced by progressive refinement. Figure 11 plots error for the data point from Figure 10(b) that had $30,000$ regions. The number of refinements is shown on the x-axis and the numbers are for the large query (`QSize` = 25%). The horizontal line in the figure shows the minimum value of error that was achieved for large queries in Figure 10(b).

At a high level, it is clear that refinements help considerably. They cause the error to drop by over 55%. However, they do not cause the error to drop to the absolute minimal level achievable by picking the correct region size, though they do come close. Note that the error starts increasing after a few refinements. This is because having too many refinements will leave too few buckets towards the end for approximating the skewed regions, thus causing some error. Given a query and data distribution, an interesting open question is to determine the optimal number of refinements and/or regions. A detailed sensitivity study on progressive refinements is beyond the scope of this paper and is part of our future work. In our experiments, we found that the best number of refinement varies from 2 to 6 depending on the query size and the input data.

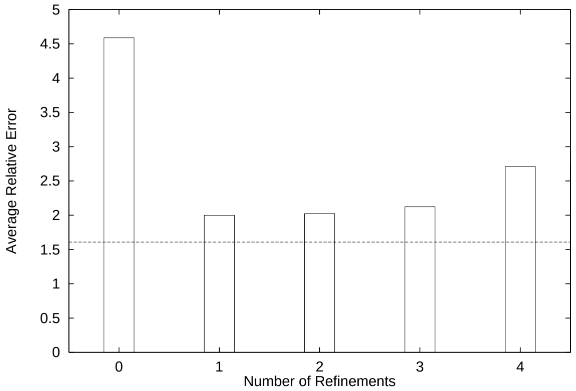


Figure 11: Impact of Progressive Refinement

Partitioning Technique	Time Taken (in seconds)			
	Input Size = $50K$		Input Size = $400K$	
	$\beta = 100$	$\beta = 750$	$\beta = 100$	$\beta = 750$
Min-Skew	5.2	15.9	20.8	33.1
Equi-Area	9.1	15.2	140.9	180.5
Equi-Count	8.1	11.3	140.8	190.3
R-Tree	3.9	6.0	57.7	891.7
Uniform	0.5	0.6	0.9	0.9

Table 1: Time for Computing Various Partitionings

5.7 Experiment 5: Construction Times

In this experiment, we look at the construction time taken by the different techniques. Table 1 shows these times for different input sizes and bucket counts (β). These times were measured on a Sparc ULTRA-30 machine with $256MB$ memory. It can be seen that the number of buckets has only a minor effect on construction time. However, all techniques, except *Min-Skew* and *Uniform*, take significantly more time with increasing data size. As mentioned earlier, this is one of the benefits of *Min-Skew*. Recall also that *Equi-Area*, *Equi-Count*, and *R-Tree* require the entire dataset to fit in memory which the *Min-Skew* technique does not.

6 Conclusions

Selectivity estimation is a critical component of query processing in databases. Despite the increasing popularity of spatial databases, there has been very little work in providing accurate and efficient techniques for spatial selectivity estimation. Spatial data differs so significantly from relational data that relational techniques simply do not perform well in this domain. In this paper, we have proposed several new techniques for spatial selectivity estimation. These techniques are based on spatial indices, binary space partitionings, and the novel notion of spatial skew. Based on our extensive experimental analysis of the new techniques and adaptations of previously known techniques, we are able to show that: (a) Sampling and parametric techniques which work well in the relational one-dimensional world do not work well for spatial data. (b) A BSP based partitioning that we call *Min-Skew* outperforms the other techniques over a broad range of query workloads and datasets. A *Min-Skew* partitioning can be constructed efficiently and has the added advantage of having low memory requirements during construction. In summary, our results show that spatial selectivity estimation can be solved accurately and efficiently for large spatial databases.

Acknowledgments: The authors would like to thank Steve Blott for help with his code for computing fractal dimensions. We obtained the Mathematica program for computing fractal dimensions from Christos Faloutsos' web site. His help is gratefully acknowledged.

References

[Ant92] G. Antoshenkov. Random sampling from pseudo-ranked B+ trees. In *Proceedings of the 18th Conference on Very Large Databases, Morgan Kaufman pubs. (Los Altos CA), Vancouver*, August 1992.

[Aok97] Paul M. Aoki. Generalizing "search" in generalized search trees. Technical Report CSD-97-950, University of California, Berkeley, June 26, 1997.

[APR99] Swarup Acharya, Viswanath Poosala, and Sridhar Ramaswamy. Selectivity estimation in spatial databases. Technical report, Bell Labs, 1999. Full version of the paper appearing in SIGMOD'99.

[ARC93] ARC/INFO. *Understanding GIS—the ARC/INFO method*. ARC/INFO, 1993. Rev. 6 for workstations.

[BF95] A. Belussi and C. Faloutsos. Estimating the selectivity of spatial queries using the correlation's fractal dimension. In *Proceedings of the 21st International Conference on Very Large Data Bases, Zurich*, 1995.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proc. SIGMOD Intl. Conf. on Management of Data*, 1990.

[CR94] C. M. Chen and N. Roussopoulos. Adaptive selectivity estimation using query feedback. *Proc. of ACM SIGMOD Conf*, pages 161–172, May 1994.

[DeW94] David J DeWitt. Client-server paradise. In *VLDB 94*, Computer Sciences Department 1210 West Dayton Street Madison, WI 53706, 1994. University of Wisconsin-Madison.

[GG82] Erol Gelenbe and Daniele Gardy. The size of projection of relations satisfying a functional dependency. *Proc. of the 8th Int. Conf. on Very Large Databases*, (1021):325–333, Sept 1982.

[GRSS97] S. Grumbach, P. Rigaux, M. Scholl, and L. Segoufin. Dedale, a spatial constraint database. In *DBPL 1997*, 1997.

[Gut85] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM-SIGMOD Conf. on Management of Data*, pages 47–57, 1985.

[HNSS95] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. *Proc. of the 21st Int. Conf. on Very Large Databases*, pages 311–322, 1995.

[Int97] Intergraph Corp. *MGE 7.0*, "http://www.intergraph.com/iss/products/mge/mge-7.0.htm", 1997.

[KMS97] S. Khanna, S. Muthukrishnan, and S. Skiena. Efficient array partitioning. *Proc Intl Colloq on Languages, Automata and Programming (ICALP)*, 1997.

[Koo80] R. P. Kooi. *The optimization of queries in relational databases*. PhD thesis, Case Western Reserver University, Sept 1980.

[LNS90] R. J. Lipton, J. F. Naughton, and D. A. Schneider. Practical selectivity estimation through adaptive sampling. *Proc. of ACM SIGMOD Conf*, pages 1–11, May 1990.

[Map98] MapInfo Corp. *The MapInfo Story*, "http://www.mapinfo.com/mapinfo/mapinfostory.html", 1998.

[MPS99] S. Muthukrishnan, Viswanath Poosala, and Torsten Suel. On rectangular partitionings in two dimensions: Algorithms, complexity, and applications. *7th International Conference on Database Theory*, January, 1999.

[PI97] Viswanath Poosala and Yannis Ioannidis. Selectivity estimation without the attribute value independence assumption. *Proc. of the 23rd Int. Conf. on Very Large Databases*, August 1997.

[PIHS96] Viswanath Poosala, Yannis Ioannidis, Peter Haas, and Eugene Shekita. Improved histograms for selectivity estimation of range predicates. *Proc. of ACM SIGMOD Conf*, pages 294–305, June 1996.

[Poo97] Viswanath Poosala. *Histogram-based estimation techniques in databases*. PhD thesis, Univ. of Wisconsin-Madison, 1997.

[PSC84] Gregory Piatetsky-Shapiro and Charles Connell. Accurate estimation of the number of tuples satisfying a condition. *Proc. of ACM SIGMOD Conf*, pages 256–276, 1984.

[SAC+79] P. 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, Massachusetts, June 1979.

[Sam89a] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison Wesley, MA, 1989.

[Sam89b] H. Samet. *The Design and Analyses of Spatial Data Structures*. Addison Wesley, MA, 1989.

[SFGM93] Michael Stonebraker, Jim Frew, Kenn Gardels, and Jeff Meredith. The SEQUOIA 2000 storage benchmark. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 22(2):2–11, June 1993.

[SRF87] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R^+-tree: A dynamic index for multi-dimensional objects. In *Proc. IEEE International Conf. on Very Large Databases*, 1987.

[tig92] Tiger/line files (tm), 1992 technical documentation. Technical report, U. S. Bureau of the Census, 1992.

[TS96] Yannis Theodoridis and Timos K. Sellis. A model for the prediction of r-tree performance. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, 1996, Montreal, Canada*, 1996.

[Ube94] M. Ubell. The montage extensible datablade architecture. In *Proc. SIGMOD Intl. Conf. on Management of Data*, 1994.

[Zip49] G. K. Zipf. *Human behaviour and the principle of least effort*. Addison-Wesley, Reading, MA, 1949.