Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension

Alberto Belussi
Dipartimento di Elettronica e Informatica
Politecnico di Milano
Milano (Italy)
e-mail: `belussi@elet.polimi.it`

*Christos Faloutsos**
Institute for Systems Research (ISR) and
Dept. of Computer Science
Univ. of Maryland, College Park
e-mail: `christos@cs.umd.edu`

February 24, 1995

Abstract

We examine the estimation of selectivities for range and spatial join queries in real spatial databases. As we have shown earlier [FK94a], real point sets: (a) violate consistently the "uniformity" and "independence" assumptions, (b) can often be described as "fractals", with non-integer (fractal) dimension. In this paper we show that, among the infinite family of fractal dimensions, the so called "Correlation Dimension" D_2 is the one that we need to predict the selectivity of spatial join.

The main contribution is that, for all the real and synthetic point-sets we tried, the average number of neighbors for a given point of the point-set follows a power law, with D_2 as the exponent. This immediately solves the selectivity estimation for spatial joins, as well as for "biased" range queries (i.e., queries whose centers prefer areas of high point density).

We present the formulas to estimate the selectivity for the biased queries, including an integration constant ($K_{`shape'}$) for each query shape. Finally, we show results on real and synthetic point sets, where our formulas achieve very low relative errors (typically about 10%, versus 40%-100% of the uniform assumption).

1 Introduction

The goal of this paper is to illustrate the power of the concept of fractal dimension as a succinct description for the distribution of real k-dimensional point-sets. Specifically, we show how to use the so-called "Correlation" fractal dimension, to estimate the selectivity of several types of spatial queries (spatial joins, range queries, etc..). Multi-dimensional point-sets have numerous applications in databases:

- traditional relational databases, where records with k-attributes become points in k-d spaces; e.g. a relation with patient data (age, blood pressure, etc.) becomes a cloud of points;

*On leave at AT&T Bell Laboratories, Murray Hill, NJ. His research was partially funded by the National Science Foundation under Grants IRI-9205273 and IRI-8958546 (PYI), with matching funds from EMPRESS Software Inc. and Thinking Machines Inc.

- geographical information systems (GIS) [Sam90] contain point data, such as cities on a two-dimensional map;
- medical image databases with, for example, three-dimensional MRI brain scans, require the storage and retrieval of point-sets, such as digitized surfaces of brain structures [ACF+93], etc.
- multimedia databases, where multi-dimensional objects can be represented as points in feature space [FRM94] e.g., 2-d color images correspond to points in (R,G,B) space (where R,G,B are the average amount of red, green and blue [FBF+94]);
- time-sequences analysis and forecasting [CE92], where k successive values are treated as a point in k-d space; correlations and regularities in this k-d space help in characterizing the dynamical process that generates the time series.

In all the above applications the distribution of k-d points is seldom (if ever) uniform [Chr84], [FK94b]. Thus, it is important to be able to characterize the deviation from uniformity in succinct way (e.g. as a sum of gaussians, or something like that). Such a description is vital for the following two requirements:

1. selectivity estimation and, in general, query optimization: "Given a range query, or a spatial join, estimate the amount of effort for a variety of alternative query plans". This is increasingly important, as the size of spatial databases increases (19Gb for the GIS data of the TIGER system of the U.S. Bureau of Census, > 1Gb and up to several Terabytes for the spatial data of the Sequoia benchmark [DKPY94], [SFGM93]).
2. "data mining" [AIS93],[AS94], with hypothesis testing and rule discovery. A succinct description of a k-d point-set could help reject quickly some non-promising hypotheses, or could help provide hints about hidden rules. For example, consider a medical database, with patient records, with k numerical attributes (eg., age, blood-pressure, cholesterol-level, etc.); in this case, a fast, positive test for uniform distribution of points in k-d space would indicate that there are no correlations or rules to search for.

We argue that the theory of fractals provide powerful tools to solve the above problems. The paper is organized as follows. Section 2 gives past work in spatial databases query optimization and analysis of spatial access methods (SAMs). Section 3 gives a survey of the main ideas in the theory of fractals. Section 4 shows how to use fractals to describe real point-sets and how to obtain accurate estimates for the selectivities of several spatial queries. Section 5 gives experimental results on real and synthetic data sets, illustrating the accuracy of our proposed formulas. Section 6 discusses the practical use of the obtain results. Section 7 lists the conclusions and directions for future research.

2 Past work in databases

There are two areas within the databases field that are related to our present work:

(a) query optimization and, specifically, selectivity estimation in multi-attribute queries;

(b) analysis of spatial access methods.

Within query optimization, a set of records with k-attributes can be seen as a set of points in k-d space. To estimate selectivities for range queries, one typically makes the *uniformity* and *independence* assumptions. These assumptions, however, do not hold on real data; moreover, they lead to pessimistic estimates [Chr84] For a single attribute, the uniformity assumption has been relaxed [IC91], typically through the use of the Zipf distribution [Zip49]. Distributions of real attributes do indeed follow the Zipf distribution or the generalized Zipf distribution: for example, word frequencies in the English language (as well as other languages); salaries [Zip49]; first names and last names of people [FJ92], etc. For multi-dimensional distributions, though, the deviations from uniformity and independence are difficult to model. The general practice is to divide the address space in cells and to keep statistics with their occupancy, in the form of histograms [IC94],[MD88].

Similar assumptions are made in the analysis of spatial access methods. Theoretical analysis in such cases assumes that points are uniformly distributed in the address space [FSR87],[AS91], which also implies that the attributes are uncorrelated. Even in simulation studies, researchers on spatial access methods and multi-attribute query optimization are forced to use ad-hoc, non-uniform distributions, such as the Gaussian distribution [NS86], some sort of clustered distributions (with points clustering around uniformly distributed sites [Ore86], or points clustering around curves, like the sinusoidal curve [BKSS90]). Although these distributions are non-uniform, they suffer from two drawbacks:

1. it is unclear whether these distributions are related to real-world distributions;
2. they do not help make the analysis tractable.

In a recent paper [FK94b] we proposed an alternative viewpoint to modeling real-world point-sets, which alleviates both of the above problems. The idea was to use concepts from the theory of fractals. Using real data sets, we showed that they indeed behave as fractals and we showed how to accurately predict the performance of R-trees [Gut84], [BKSS90] using the *Hausdorff fractal dimension* (D_0) of the target point-set. Next we introduce the basic concepts from the theory of fractals. Later on we show how to use a different fractal dimension, the 'Correlation' one, to estimate the selectivities of spatial queries.

3 Introduction to fractals

Intuitively, a set of points is a fractal if it exhibits self-similarity over all scales. This is illustrated by an example: Figure 1 shows the first few steps in constructing the so-called *Sierpinski triangle*. Figure 2(a) gives 5,000 points that belong to this triangle. Theoretically, the Sierpinski triangle

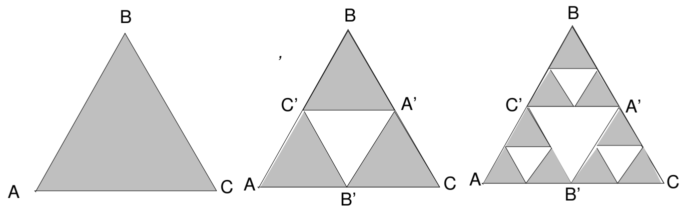
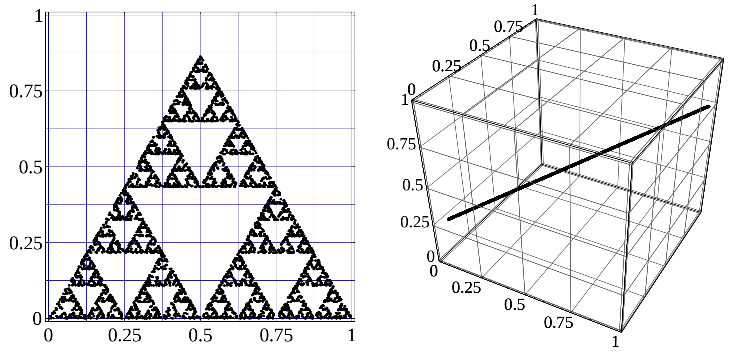


Figure 1: *Sierpinski triangle: the first 3 steps of its recursive construction.*

is derived from an equilateral triangle ABC by excluding its middle (triangle A'B'C') and by recursively repeating this procedure for each of the resulting smaller triangles. The resulting set of points exhibits 'holes' in any scale; moreover, each smaller triangle is a *miniature replica* of the whole triangle. In general, the characteristic of fractals is this *self-similarity* property: parts of the fractal are similar (exactly or statistically) to the whole fractal. For our experiments we use 5,000 sample points from the Sierpinski triangle (termed *'Sierpinski5K'* dataset from now on), using Barnsley's algorithm of Iterated Function Systems [BS88] to generated these points quickly.



(a): *'Sierpinski5K'* dataset (b): *'StrLine3D'* dataset

Figure 2: *Theoretical fractals: (a) sample of the Sierpinski triangle (b) a line in 3-d space)*

Like all fractals, the Sierpinski triangle is a rich source of paradoxes: it is a point-set

- with area zero (being proportional to: $\lim_{i\to\infty}(3/4)^i$),
- and with infinite-length perimeter (proportional to $\lim_{i\to\infty}(1+1/2)^i$).

Thus, it is not a 1-dimensional Euclidean object (otherwise it would have finite length perimeter), but it is not a 2-dimensional Euclidean object either (since it has zero area). The way to resolve the issue is to consider *fractional* dimensionalities, which are called *fractal dimensions*. As we shall see shortly, there are several definitions; among them, we first choose the *Hausdorff* fractal dimension, or box-counting dimension or D_0, because it is easier to describe.

The *Hausdorff fractal dimension* D_0 for a given point-set in an E-dimensional address space is defined as follows [Sch91]. Divide the E-dimensional space into (hyper-)cubic grid cells of side

r. Let $N(r)$ denote the number of cells that are penetrated by the set of points (i.e., that contain one or more of its points). Then the (box-counting) fractal dimension D_0 of a fractal is defined as

$$D_0 \equiv \lim_{r \to 0} \frac{\log N(r)}{\log(1/r)} \tag{1}$$

This definition is useful for mathematical fractals, that consist of infinite number of points. For a finite number of points, we avoid the limit $r \to 0$; instead we restrict our attention to a suitable range of scales $r \in (r_1, r_2)$, in which the point-set exhibits (statistical) self-similarity. More specifically, we use the *box-count* plot, which plots $\log(N(r))$ vs. $\log(r)$ (e.g., see Figure 4(a), top row, for the box-count plot of the *'Sierpinski5K'* dataset). If the point-set is self-similar for $r \in (r_1, r_2)$, then its box-count plot will be a straight line for this range. The (negated) slope of this line is the Hausdorff fractal dimension D_0 of the point-set for the range of scales (r_1, r_2):

Definition 1 (Hausdorff fractal dimension) *For a point-set that has the self-similarity property in the range of scales (r_1, r_2), its Hausdorff fractal dimension D_0 for this range is measured as*

$$D_0 \; = \; - \frac{\partial \log(N(r))}{\partial \log(r)} \; = \; \text{constant} \qquad \textit{for } r_1 < r < r_2 \tag{2}$$

An interesting observation is that the above definition encompasses traditional Euclidean objects:

Observation 1 *For Euclidean objects, their fractal dimension equals their Euclidean dimension.*

Thus, lines, line segments, circles, and all the standard curves have D_0=1; planes, disks and standard surfaces have D_0=2; Euclidean volumes in E-dimensional space have $D_0 = \mathrm{E}$.

Figure 4(a)(top row) shows the box-count plot for D_0 for the *'Sierpinski5K'* dataset. Notice that the slope for $r \in (e^{-4.5}, e^{-1})$ is 1.574, very close to the theoretical value of $\log 3/ \log 2 = 1.585$ [Man77]. Also notice that the horizontal parts of the plot are perfectly explainable:

- For very fine granularities (i.e., $r \to 0$), each of the 5,000 points eventually is in a cell by itself. Thus, it becomes clear that the point-set is a finite collection of points (each with fractal dimension $D_0 = 0$), and therefore, the collective Hausdorff fractal dimension is also 0. Notice that the limit value $\lim_{r \to 0} N(r)$=5,000.
- For very coarse granularities (i.e., $r \to \infty$), the whole point-set fits in a single cell ($\lim_{r \to \infty} N(r) = 1$), and thus behaves like a single point.

As mentioned earlier, there are more than one fractal dimensions; in fact, there is an infinite family of them. For a finite point set, the generalized fractal dimension D_q (where q is a real number) is as follows. Consider again a grid with cells of side r, and let p_i denote the frequency with which points fall into the i-th cell of the grid.

Definition 2 (Generalized fractal dimension) *For a point-set that has the self-similarity in the range of scales* (r_1, r_2)*, the generalized fractal dimension* D_q *is defined as*

$$D_q \equiv \frac{1}{q-1}\frac{\partial \log \sum_i p_i^q}{\partial \log r} = \text{constant} \qquad q \neq 1,\ \text{for } r_1 < r < r_2 \tag{3}$$

$$D_1 \equiv \frac{\partial \log \sum_i p_i \log p_i}{\partial \log r} = \text{constant} \qquad q = 1,\ \text{for } r_1 < r < r_2 \tag{4}$$

Clearly, the plot of $\log \sum_i p_i^q$ versus $\log r$ is vital for the estimation of a generalized fractal dimension D_q. For the rest of this paper, we shall refer to it by *the generalized box-count plot* or simply *the box-count plot*. In the definition of the generalized fractal dimension, notice that:

- for $q = 0$, we have the Hausdorff fractal dimension D_0;
- for a strictly self-similar point set (eg., like the Sierpinski triangle), we have $\forall q : D_q = D_0$ (see proof in [Sch91]).
- for $q = 2$, we have the so-called 'correlation' fractal dimension D_2, which is the one we shall use next:

$$\boxed{D_2 \equiv \frac{\partial \log \sum_i p_i^2}{\partial \log r} = \textit{constant} \qquad r \in (r_1, r_2)} \tag{5}$$

Thus, for the rest of this paper, we focus on $q = 2$. To make the discussion more clear, we introduce the term *"sum of squared occupancies"* $S_2(r)$:

Definition 3 *For a point-set* $\mathcal{P}$ *in a grid of cell-side r, the sum of squared occupancies* $S_2(r)$ *is defined as:*

$$S_2(r) \equiv \sum_i p_i^2 \tag{6}$$

We close this introduction to fractals with a formula which is valuable for selectivity estimation: by integrating Equation 3, we have that the sum of squared occupancies follows the power law:

$$\boxed{S_2(r) \propto r^{D_2}} \tag{7}$$

where the symbol '$\propto$ ' stands for 'proportional'. Combined with the upcoming Lemma 1 this power law is the first stepping stone towards the desired selectivity estimation formulas.

Symbols	Definitions
D_q	general Fractal Dimension
D_0	Hausdorff Dimension
D_2	Correlation Dimension
E	Dimensionality of address space ('Embedding Dimension')
$\mathcal{P}$	set of points
$\vec{q}$	single multi-dimensional point
N	cardinality of the considered point set
r	length of the side of a grid cell
p_i	frequency with which points fall into each grid cell
$S_2(r)$	sum of squared occupancies
ϵ	radius of the spatial join or size of the query region
$Sel_{r-range}(\epsilon)$	selectivity of random range queries
$Sel_{b-range}(\epsilon)$	selectivity of biased range queries
$Sel_{join}(\epsilon)$	selectivity of spatial join
$\overline{nb}(\epsilon)$	average number of neighbors
$'shape'$	shape of the query region
$step(x)$	function which return 1 if $x > 0$, 0 otherwise
$dist(\vec{q_i}, \vec{q_j})$	Euclidean distance between two points $\vec{q_i}$ and $\vec{q_j}$

Table 1: *Definition of symbols*

3.1 Description and fractal dimensions of sample datasets

The reader might be wondering whether real datasets behave like fractals, with linear box-count plots. In this subsection we give (a) a description of 4 datasets (2 real and 2 synthetic), that we shall use throughout this paper and (b) their box-count plots, for the Hausdorff (D_0) and correlation (D_2) fractal dimensions.

The two real point-sets are road intersections of U.S. counties, from the TIGER database of the U.S. Bureau of Census:

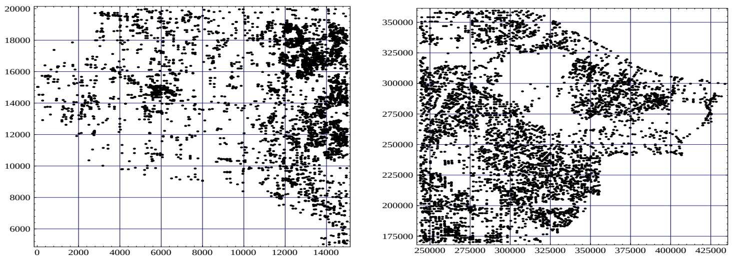
- 'MCnty' *dataset*: road intersections from the Montgomery County, Maryland, with N =9,552 points (see Figure 3(a));
- 'LBCnty' *dataset*: road intersections from the Long Beach County, California, with N=10,377 points (see Figure 3(b)).

As a 'sanity check' for our formulas and algorithms, we also used synthetic point-sets, which are self-similar and have known fractal dimensions:

- 'Sierpinski5K' *dataset*: a 5,000 point sample from the Sierpinski triangle ($D_q = 1.585\ \forall q$); see Figure 2(a)
- 'StrLine3D' *dataset*: a 5,000 point sample from a straight line in 3 dimensional space (D_q=1 $\forall q$); see Figure 2(b).

Next, we give the box-count plots for the above 4 datasets, for both the Hausdorff and correlation fractal dimension. For the two synthetic datasets (Figures 4(a, b)) the plots are indeed straight lines in the suitable range of scales, and the slope is very close to the theoretically expected ones (the relative error is less than 2%).

For the real datasets, the box count plots are shown in Figure 4(c,d). Notice that they, too, have plots that are straight lines for suitable ranges of scales. The slopes are *smaller* than the embedding dimension (E=2), reflecting the visually obvious fact (see Figures 3(a,b)) that the point sets are *not* uniformly distributed in the address space.



(a) *'MCnty'* dataset (b) *'LBCnty'* dataset

Figure 3: *Real data sets: road intersections from (a) the Montgomery county, MD; and (b) and the Long Beach county, CA.*

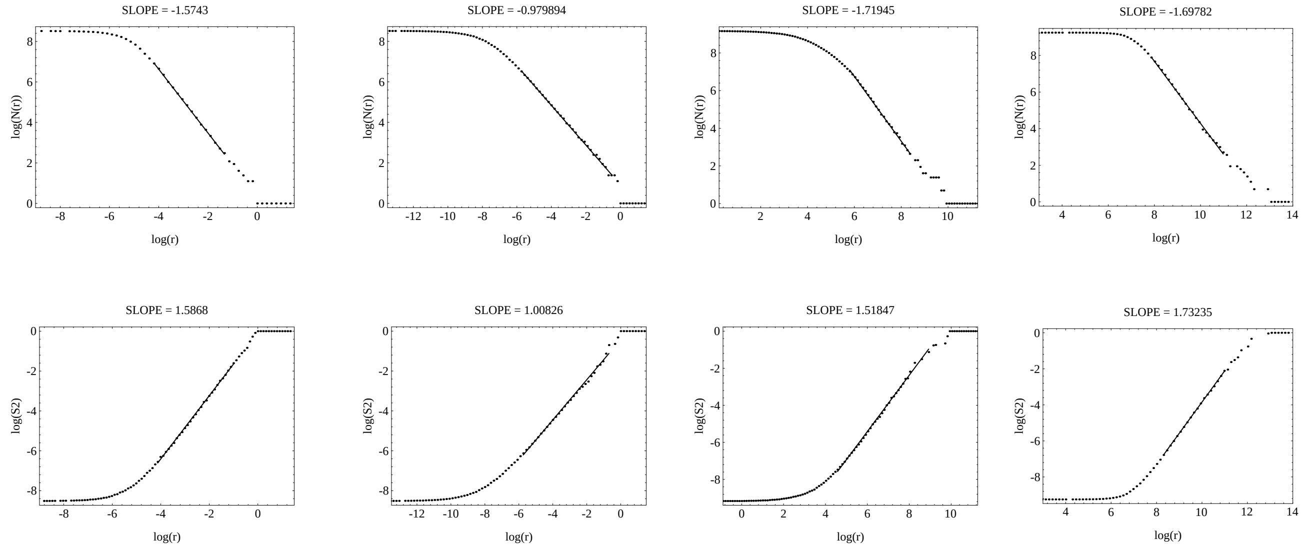
Table 2 lists the measured D_0 and D_2 for each dataset. The last column lists the theoretical fractal dimension, whenever known. We shall use the results of this Table in the experiments (Figures 7 and 5).

Data Sets	Measured		Theoretical
	D_0	D_2	D_q
'Sierpinski5K' dataset	1.574	1.587	1.585
'StrLine3D' dataset	0.979	1.008	1.000
'MCnty' dataset (road int. map)	1.719	1.518	N/A
'LBCnty' dataset (road int. map)	1.697	1.732	N/A

Table 2: *Summary of measured fractal dimensions (D_0 and D_2), for all datasets*

4 Selectivity estimation

In the previous section we saw the definitions of several fractal dimensions, and specifically, the 'correlation' fractal dimension D_2 of a point-set. Next, we show how to use this machinery, in order



(a) *'Sierpinski5K'* dataset (theor. $D_0 = D_2$=1.585) (b) *'StrLine3D'* dataset (theor. $D_0 = D_2$=1.000) (c) *'MCnty'* dataset (d) *'LBCnty'* dataset

Figure 4: *Box count plots for all four datasets, for the D_0 (top row) and D_2 (bottom row)*

to estimate the selectivities for spatial queries, and specifically (a) for range queries and (b) for spatial joins.

First, we give some preliminary definitions and the problem description; then we show that the desired selectivities follow a power law, with exponent the correlation fractal dimension D_2; next we give a formula to estimate the constant of proportionality; and finally we combine everything in the main theorem, in Eq. (34). Each subsection corresponds to each of the above steps, respectively.

4.1 Definitions and problem description

As mentioned before, we focus on range queries and on spatial joins. A range query specifies a region in the address space and requires all the data items that intersect this region. Thus, we can describe a range query as a triplet:

$$< \text{`}shape', \epsilon, \vec{q} >$$

where:

- '*shape*' represents the shape of the query region (e.g.: square, circle, diamond, etc.); without loss of generality, we designate the center of gravity of the query region as the '*anchor*' point.
- $\vec{q}$ is the position of the 'anchor' in the address space,
- ϵ is the extent of the shape, that is, the distance of the center-of-gravity to the most remote point of the shape on the positive x-axis (e.g., in the case of a circle, ϵ is the radius).

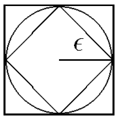


Figure 5: Query shapes with radius ϵ.

Figure 5 shows some shapes (square, circle, diamond) of the same radius ϵ.
We distinguish between two models [PSTW93] for the probability distribution of the anchors $\vec{q}$:

1. the *Random model*, which assumes that the anchors are uniformly distributed in the address space and
2. the *Biased model*, which assumes that queries are more probable in high-density areas of the address space. For example, in a GIS/transportation application with a map of cities, we would expect few queries on deserts and bodies of water, and more queries on highly populated areas. In the 'biased model', we assume that the anchors are allowed to land only on data points; thus, high-density areas attract more queries.

A '*self spatial join*', or simply a '*spatial join*', of a set of points $\mathcal{P}$ is a query that requests all unique pairs of distinct points, whose relative distance is less than a given radius 'ϵ'. We typically use the Euclidean distance as the distance metric.

For any query, its selectivity is defined as the *proportion of records* that it retrieves.

For range queries under the random model, the problem is easy: the selectivity $Sel_{r-range}(\epsilon, \text{`}shape')$ for a query of the specified shape with radius $\epsilon$ is the relative volume $Vol(\epsilon, \text{`}shape')$ of the query shape:

$$Sel_{r-range}(\epsilon, \text{`}shape') = \frac{Vol(\epsilon, \text{`}shape')}{(\text{volume of address space})} \tag{8}$$

Since Eq. (8) solves the problem, we will not examine the random model any further.

Before we give the formulas to express the selectivities of biased range queries ($Sel_{b-range}(\epsilon)$) and of spatial joins ($Sel_{join}(\epsilon)$), we define two auxiliary quantities, which will make the presentation and the proofs more clear: (a) the total number $UniquePairs(\epsilon)$ of unique pairs of points that lie within distance ϵ from each other, and (b) the average number of neighbors $\overline{nb}(\epsilon)$ within distance ϵ from a data point.

Definition 4 *The total number of unique pairs $UniquePairs(\epsilon)$ of a point set $\mathcal{P}$ is defined as:*

$$UniquePairs(\epsilon) \equiv \sum(\text{ unique pairs of points } (\vec{q_i}, \vec{q_j}) \text{ within Euclidean distance } \epsilon\ ) \quad \vec{q_i}, \vec{q_j} \in \mathcal{P},\ i \neq j \tag{9}$$

where 'unique' means that $(\vec{q_i}, \vec{q_j})$ and $(\vec{q_j}, \vec{q_i})$ are counted once. More formally:

$$UniquePairs(\epsilon) \equiv \sum_{i=1}^{N} \sum_{j=i+1}^{N} step(\epsilon - dist(\vec{q_i}, \vec{q_j})) \quad \vec{q_i}, \vec{q_j} \in \mathcal{P} \tag{10}$$

where: $dist(\vec{q_i}, \vec{q_j})$i is the Euclidean distance between two points $\vec{q_i}$, $\vec{q_j}$ and $step(x)$ =1 if x is positive, =0 otherwise.

Definition 5 *The average number of neighbors $\overline{nb}(\epsilon)$ is defined as*

$$\overline{nb}(\epsilon) \equiv \text{(avg. \# of neighbors within distance } \epsilon)$$

or

$$\overline{nb}(\epsilon) = 1/N \ \textstyle\sum_i^N (\ \text{\# points within distance } \epsilon \text{ from the } i\text{-th point}) \tag{11}$$

where the summation is over all the N points of the point-set $\mathcal{P}$. Notice that this summation is exactly twice as large as the summation of Eq. (10), because it counts each pair of points twice. Thus

$$\overline{nb}(\epsilon) = 1/N \times 2 \times UniquePairs(\epsilon) \tag{12}$$

We can generalize the above definition, to include other distance measures, or even other query shapes, like diamonds, squares, etc.:

Definition 6 *The average number of neighbors $\overline{nb}(\epsilon, \text{'shape'})$ for a specified query shape is defined as*

$$\begin{aligned} \overline{nb}(\epsilon, \text{'shape'}) &\equiv \text{(avg. \# of neighbors within a '}\textit{shape}\text{' of radius } \epsilon) \\ &\equiv 1/N \sum_i^N (\text{\# points within a shape of radius } \epsilon \\ &\qquad \text{anchored at the } i\text{-th point}) \end{aligned} \tag{13}$$

We designated the default shape to be the 'circle' (corresponding to Euclidean distances). That is

$$\overline{nb}(\epsilon) \equiv \overline{nb}(\epsilon, \bigcirc) \tag{14}$$

Based on that, the selectivities for the range query and the spatial join can be expressed as follows:

$$Sel_{b-range}(\epsilon) = \overline{nb}(\epsilon) + 1 \tag{15}$$

and

$$
\begin{aligned}
Sel_{join}(\epsilon) &= \frac{UniquePairs(\epsilon)}{N \times (N-1)/2} \\
&= \frac{\overline{nb}(\epsilon)}{N-1} \qquad (16)
\end{aligned}
$$

Thanks to Eq. (15,16), we only need to study the average number of neighbors $\overline{nb}(\epsilon)$.

Before we proceed with the major results, we mention the concept of '*Correlation Integral*', which has been studied in the theory of fractal dimension, and, specifically, in connection to the 'correlation' fractal dimension. The correlation integral can be defined in two ways, depending on whether we count the 'self-pairs' (like $(\vec{q_i}, \vec{q_i})$) or not. Here we follow the definition by Richard L. Smith in [Smi92], where self-pairs are not counted. Then, the 'Correlation Integral' $C(\epsilon)$ coincides exactly with our definition for the selectivity of spatial joins $Sel_{join}(\epsilon)$:

Definition 7 *The Correlation Integral $C(\epsilon)$ of a point set $\mathcal{P}$ is defined as:*

$$
C(\epsilon) \equiv \frac{\sum \text{(unique pairs of points } (\vec{q_i}, \vec{q_j}) \text{ within Euclidean distance } \epsilon)}{N \times (N-1)/2} \qquad \vec{q_i}, \vec{q_j} \in \mathcal{P},\ i \neq j \quad (17)
$$

which is identical to our definition of spatial join selectivity:

$$
\boxed{C(\epsilon) \equiv Sel_{join}(\epsilon)} \qquad (18)
$$

After these preliminary definitions and observations, we are ready for the major results.

4.2 A power law for selectivity estimation

Our goal is to find a formula to estimate the average number of neighbors $\overline{nb}(\epsilon, \text{'}shape')$. An important stepping stone towards our goal is provided by the following Lemma:

Lemma 1 (Schuster) *Given a point set $\mathcal{P}$ and the sum of squared occupancies $S_2(\epsilon)$ ($=\sum_i p_i^2$) on a grid with cells of side ϵ, we have:*

$$
\boxed{C(\epsilon) \propto S_2(\epsilon)} \qquad (19)
$$

Proof: See [Sch88].

An obvious consequence of the above Lemma is the following power law:

Lemma 2 *Given a set of points $\mathcal{P}$ with finite cardinality N and its Correlation Dimension D_2, the average number of neighbors within radius ϵ follows the power law:*

$$\overline{nb}(\epsilon) \propto \epsilon^{D_2} \tag{20}$$

Proof: Trivially, from Eqs. (19) and (7). QED

Using Eq. (20), we know how to estimate the average number of neighbors, except for a constant of proportionality. Our experiments (see Section 5) led to an observation that simplifies greatly the estimation of this constant. This observation states that *other* query shapes, too, obey the same power law, with the *same* (!) exponent D_2:

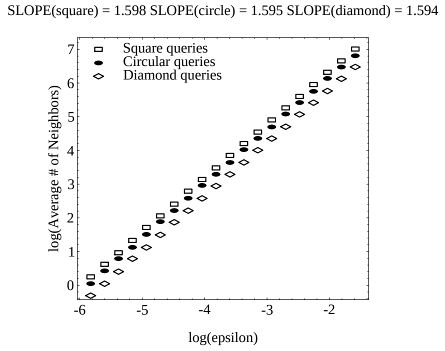


Figure 6: 'Sierpinski5K' *dataset: measured average number of neighbors for different query shapes.*

Observation 2 *The measured average number of neighbors for any query shape follows a power law with exponent the correlation fractal dimension:*

$$\boxed{\overline{nb}(\epsilon, \text{`shape'}) = K_{\text{`shape'}} \times \epsilon^{D_2}} \tag{21}$$

where $K_{`shape'}$ depends on the specific query shape.

The observation is illustrated in Figure 6, which plots the measured values of $\overline{nb}()$ for several query shapes (square, circle and diamond). The point-set was the *''Sierpinski5K''* dataset. The horizontal axis is the radius ϵ; both axes are logarithmic. Notice that all the plots are straight lines for a long range of scales, and that their slopes are extremely close to the measured Correlation fractal dimension D_2= 1.587 of the point-set.

The next subsection is devoted to the estimation of the $K_{`shape'}$ constant, to complete the estimation of selectivities and average number of neighbors.

4.3 Estimation of the constant $K_{`shape'}$

Our goal is to estimate the constant $K_{\bigcirc}$ for circles (ie., Euclidean distances). A head-on attack on this problem seems difficult. Instead, we propose to exploit Observation 2, to estimate the constant

$K_{\text{`}shape'}$ for a more convenient shape first, and, specifically, a square. Without loss of generality, we can normalize the address space to the unit hyper-cube. Then, we have:

Lemma 3 *For a square query shape, the constant of proportionality $K_{\square}$ is given by:*

$$K_{\square} = (N - 1) \times 2^{D_2} \tag{22}$$

Proof: The idea is that a query that covers the whole address space should retrieve $N-1$ neighbors. Since queries that exceed the limits of the address space are 'wrapped around' [KF94],[Fal92], a radius of $\epsilon = 1/2$ is needed to cover the address space. Thus

$$\overline{nb}(1/2, \square) = N - 1 \tag{23}$$

Combining Eq. (23) with Eq. (21), we prove Eq. (22). QED

From the above lemma, we have finally:

$$\boxed{\overline{nb}(\epsilon, \square) = (N - 1) \times (2 \times \epsilon)^{D_2}} \tag{24}$$

The formula for the constant $K_{\text{`}shape'}$ is based on the following assumption:

Assumption 1 *For a fractal point set $\mathcal{P}$, biased range queries with equal (hyper-)volumes, retrieve on the average the same number of points, regardless of the query shape.*

We do not have a solid justification for this assumption, apart from the fact that it 'sounds right'. However, in our pursuit of an accurate selectivity formula, we don't need to provide a justification: this assumption eventually leads to predictions which are consistently accurate (see Figure 8 in the experiments section). Since the assumption works, we should use it! The consequence of this assumption is the following:

Lemma 4 *For a given point set, the constant $K_{\text{`shape'}}$ is given by*

$$\boxed{K_{\text{`shape'}} \;=\; K_{\square} \times \left( \frac{Vol(\epsilon, \text{`shape'})}{Vol(\epsilon, \square)} \right)^{D_2/E}} \tag{25}$$

where D_2 is the Correlation dimension of the point set, E is the embedding dimension and $Vol(\epsilon, \text{`}shape')$ gives the volume of the specified shape with radius ϵ.

Proof: Consider a query ($Q_{\text{`}shape'}$) of the desired shape and radius ϵ, and a query ($Q_{\square}$) of square (i.e., E-d hyper-cube) shape and the same radius. Then we have

$$\overline{nb}(\epsilon, \square) = K_{\square} \times \epsilon^{D_2} \tag{26}$$

$$\overline{nb}(\epsilon, \text{`}shape') = K_{\text{`}shape'} \times \epsilon^{D_2} \tag{27}$$

The main idea in the proof is to consider a query $Q'_{\Box}$, which is a square query with the same *volume* as $Q_{`shape'}$. Let ϵ_* denote the radius of this query. Then we have

$$Vol(\epsilon, `shape') = Vol(\epsilon_*, \Box) \tag{28}$$

and, by Assumption 1

$$\overline{nb}(\epsilon, `shape') = \overline{nb}(\epsilon_*, \Box) \tag{29}$$

From Eq. (21) we have

$$\overline{nb}(\epsilon_*, \Box) = K_{\Box} \ \times \ \epsilon_*^{D_2} \tag{30}$$

For a E-dimensional hypercube, the volume is given by

$$Vol(\epsilon, \Box) = (2\epsilon)^E \tag{31}$$

$$Vol(\epsilon_*, \Box) = (2\epsilon_*)^E \tag{32}$$

Putting all of the above together, we have

$$\begin{aligned} K_{`shape'}/K_{\Box} &= \overline{nb}(\epsilon, `shape') \ / \ \overline{nb}(\epsilon, \Box) \\ &= \overline{nb}(\epsilon_*, \Box) \ / \ \overline{nb}(\epsilon, \Box) \\ &= (\epsilon_*/\epsilon)^{D_2} \\ &= ((Vol(\epsilon_*, \Box) \ / \ Vol(\epsilon, \Box))^{1/E})^{D_2} \\ &= (Vol(\epsilon, `shape') \ / \ Vol(\epsilon, \Box))^{D_2/E} \end{aligned} \tag{33}$$

QED

Table 3 gives arithmetic examples of the ratio $K_{`shape'}/K_{\Box}$ for circles and diamonds, for the four sample datasets. We use the measured value of D_2, from Table 2. The measured D_2 is repeated in the second column, for convenience.

4.4 Main result

The final conclusion of all these mathematical derivations is the formula that estimates the average number of neighbors for *any* query shape, as a function of the correlation fractal dimension D_2.

Theorem 1 *The average number of neighbors for a pointset $\mathcal{P}$ is given by*

$$\boxed{\overline{nb}(\epsilon, \text{`shape'}) \ = \ (Vol(\epsilon, \text{`shape'})/Vol(\epsilon, \Box))^{D_2/E} \times (N-1)2^{D_2} \times \epsilon^{D_2}} \tag{34}$$

Sample Sets	D_2	$K_{`shape'}/K_{\square}$	
		Circle	Diamond
Synthetic Data			
'Sierpinski5K'	1.587	$(\pi/4)^{1.587/2} = 0.826$	$(1/2)^{1.587/2} = 0.577$
'StrLine3D'	1.008	$(\pi/6)^{1.008/3} = 0.805$	$(1/6)^{1.008/3} = 0.548$
Real Data			
'MCnty'	1.518	$(\pi/4)^{1.518/2} = 0.832$	$(1/2)^{1.518/2} = 0.591$
'LBCnty'	1.732	$(\pi/4)^{1.732/2} = 0.811$	$(1/2)^{1.732/2} = 0.549$

Table 3: *Theoretical values for the ratio* $K_{`\mathrm{shape}'}/K_{\square}$

where D_2 is the correlation fractal dimension of the point-set, N is the number of points in the point-set and $Vol(\epsilon, `shape')$ is the volume of a shape of radius ϵ.

Proof: By substituting Eqs. (25) and (22) into Eq. (21). QED

From the above theorem and Equations (15) and (16), we can estimate the selectivity of the spatial join ($Sel_{join}(\epsilon)$) and of the biased range queries ($Sel_{b-range}(\epsilon)$), which was our initial goal. The question is to find out how accurate these formulas are, in a real setting. This is exactly the goal of the next section.

5 Experiments

The purpose of the experiments is to test the prediction accuracy of our main result, Eq. (34). This equation estimates the average number of neighbors $\overline{nb}(\epsilon, `shape')$ for a given query, in a point-set with N points and correlation fractal dimension D_2. For our experiments, we used the four datasets described in subsection 3.1. Recall (subsection 3.1) that we have already verified that all four datasets behave like fractals, that is, they have linear box-count plots. Their correlation fractal dimensions D_2 are in Table 2.

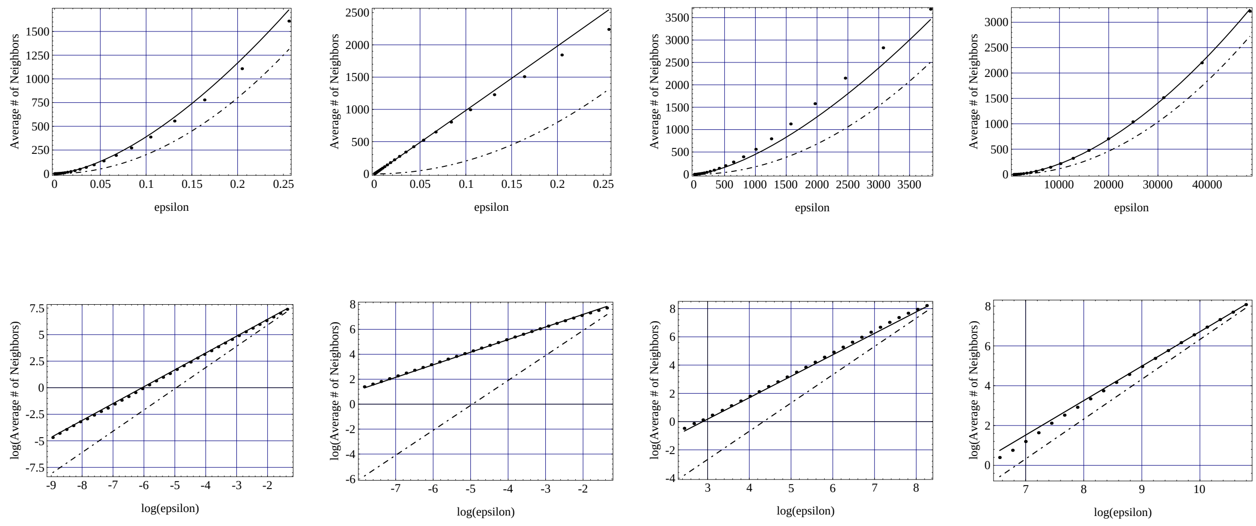
We present two sets of experiments. In the first set, we examine the accuracy of our analysis for square queries, because the derivations for square queries (Eq. (24)) required fewer assumptions than the rest of the shapes; if our analysis is inaccurate for square queries, it will be at least as inaccurate for the rest of the shapes. In the second set, we examine additional query shapes (namely, circles and diamonds).

All the upcoming plots have the same format: they give the average number of neighbors $\overline{nb}(\epsilon, `shape')$ versus the radius ϵ of the query. Our estimates (using Eq. (34)) are shown in solid line; a dashed line shows the estimates under the uniformity assumption (i.e., by Eq. (34), after forcing D_2 to be E). Actual measurements are represented by 'bullets' ($\bullet$); for each such 'bullet' we measured the number of neighbors within the desired radius ϵ for each point of the point-set, and averaged the results. To accelerate the searches, we used an R-tree spatial index. Also, we selected the radii so that they form a geometric progression.

5.1 Accuracy of predictions for square queries

Figure 7 shows the results of the experiments for square queries. Each column corresponds to the indicated dataset. Each plot follows the format mentioned above: it shows the average number of neighbors $\overline{nb}(\epsilon, \Box)$ versus the radius ϵ of the square queries; actual measurements are shown as 'bullets'; our estimates, using the measured D_2 (see Table 2) and Eq. (34) are shown with a solid line; the results of the uniformity assumption are shown with a dashed line.

The plots in the top and bottom rows use linear and logarithmic scales, respectively. The consistent conclusion is that Eq. (34) (which reduces to Eq. (24) for square shapes) gives very accurate predictions. In contrast, the uniformity assumption may give large errors; its errors increase with the discrepancy between D_2 and E, as intuitively expected. The largest error is for the *'StrLine3D'* dataset (Figure 7(b)), which has the highest such discrepancy.



(a) *'Sierpinski5K'* Measured D_2 = 1.587 (b) *'StrLine3D'* Measured D_2 = 1.008 (c) *'MCnty'* Measured D_2 = 1.518 (d) *'LBCnty'* Measured D_2 = 1.732

Figure 7: *All four datasets, with square queries. Average number of neighbors $\overline{nb}(\epsilon, \Box)$ vs. ϵ in linear (top row) and logarithmic plots (bottom row): actual measurements ('bullets'), estimates with D_2 (solid line), estimates with uniformity assumption (dashed line).*

Table 4 lists the exact values of the errors for our predictions, as well as the predictions of the uniformity assumption. Following the recommendations from statistics, we compute the *geometric* average of relative errors, for each setting. While the uniformity assumption leads to errors in the range of 40%-100%, the accuracy of our predictions is striking, typically within 10%.

Data Sets	Avg. Relative Error	
	Proposed formula	Uniformity assumption
Synthetic Data		
'Sierpinski5K'	8.9%	61.2%
'StrLine3D'	3.3%	97.1%
Real Data		
'MCnty'	12%	75.8%
'LBCnty'	4.5%	43.2%

Table 4: *Average relative error in estimating* $\overline{nb}(\epsilon, \Box)$.

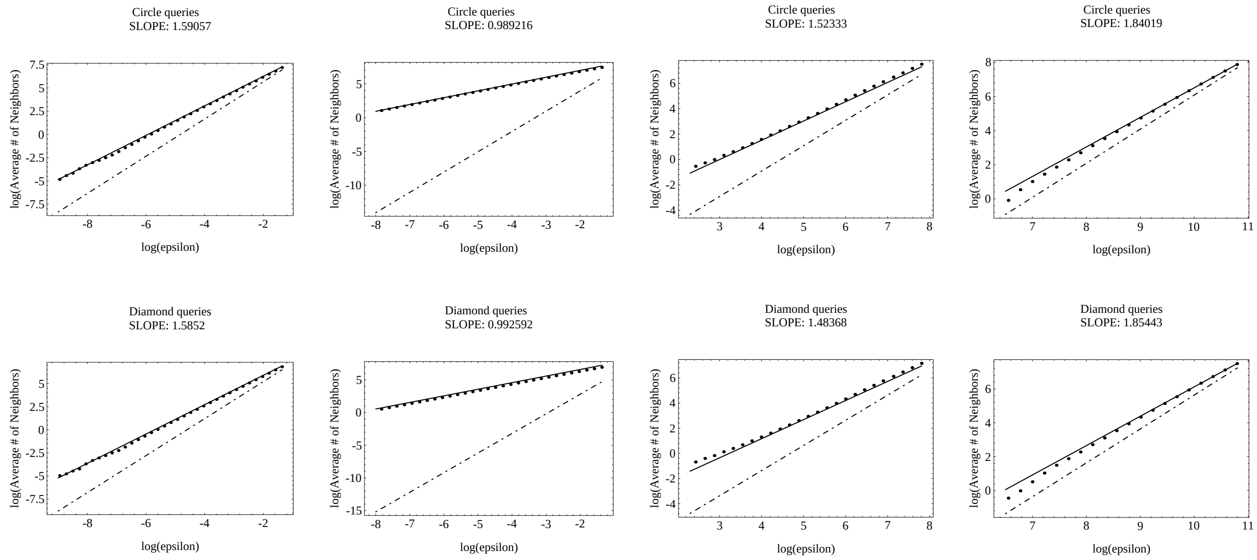
5.2 Accuracy of predictions for arbitrary query shapes.

Here we list the plots for queries of additional shapes (circles and diamonds). Figure 8 has the results. All the plots follow the same format as before: they give the average number of neighbors vs. the radius ϵ, for all 4 datasets and for circle queries (top row) and diamond queries (bottom row). The 'bullets' are the actual measurements, the solid line plots our predictions, and the dashed line plots the predictions of the uniformity assumption. The observations are as follows:

- As mentioned in Observation 2, the 'bullets' of any plot fall on a straight line, whose slope is very close to the measured correlation fractal dimension D_2 of the respective point-set.
- Our predictions are consistently good, justifying our Assumption 1, which was necessary for the estimation of the ratio $K_{`shape'}/K_{\Box}$.
- The predictions of the uniformity assumption can lead to large errors, as happened for the square queries. The error in the predictions increases with the deviation of the dataset from uniformity $(E - D_2)$, as expected: the smallest error is for the *'LBCnty'* dataset ($E - D_2$ = 2-1.732 = 0.268), while the largest error is again for the *'StrLine3D'* dataset ($E - D_2$ = 3-1.008 $\approx$2).

Finally, Tables 5 and 6 list in detail the (geometric) average of the relative errors for circles and diamonds, respectively. The observations are

- the relative errors seem insensitive to the shape of the queries (compare also with the errors for square queries, in Table 4). The only major change is in the error for our formula, for diamond queries on the *'StrLine3D'* dataset (3.3%, 9.9% and 26% for squares, circles and diamonds, respectively). The phenomenon is probably due to unlucky relative orientation of the the 3-d line with respect to the surfaces of the diamonds (ie., octahedra, in 3-d). The error for the uniformity assumption is not changed, probably because it was large to begin with (97.1%, 98% for squares and circles, respectively).
- with the above exception, the accuracy of our predictions is in the 10-15% range, while the competition remains in the 40-100% range.



(a) *'Sierpinski5K'* Measured $D_2 = 1.587$

(b) *'StrLine3D'* Measured $D_2 = 1.008$

(c) *'MCnty'* Measured $D_2 = 1.518$

(d) *'LBCnty'* Measured $D_2 = 1.732$

Figure 8: *All four datasets, with circle (top row) and diamond queries (bottom row). Average number of neighbors* $\overline{nb}()$ *vs.* ϵ *in logarithmic plots: Actual (bullets), estimates with* D_2 *and* $K_{\text{'shape'}}$ *(solid line), estimates with uniformity assumption (dashed line).*

6 Discussion

Here we discuss some questions about the practicality of the proposed concepts and formulas.

Q1 *How often do real point-sets behave like fractals?*
Surprisingly often. Recall that Euclidean objects (smooth curves and surfaces), as well as uniformly distributed point sets behave like fractals, with fractal dimension their Euclidean dimension. There is overwhelming evidence [Man77](p. 447),[Sch91] that a huge number of real point sets behave like a fractal, for an appropriate range of scales: coast lines and country borders ($D_0 \approx 1.2 - 1.3$); the periphery of clouds and rainfall patches ($D_0 \approx 1.35$)[Sch91](p.231); the distribution of galaxies in the universe ($D_0 \approx 1.23$); the brain surface of mammals ($D_0 \approx 2.7$); the human vascular system ($D_0 = 3$, because it has to reach every cell in the body!) and so on. Thus, applications with GIS, with meteorological databases, with medical image databases, etc., will encounter fractal sets very often.

Q2 *How would a practitioner use the provided formulas?*
The setting we envision is as follows: given a point-set (e.g., a set of cities of a country, as 2-d points), the practitioner needs to compute the correlation dimension D_2. An efficient

Circle - avg. relative error		
Sample sets	Proposed formula	Uniformity assumption
Synthetic Data		
'Sierpinski5K'	8.1%	71,6%
'StrLine3D'	9.9%	98%
Real Data		
'MCnty'	12.4%	80.4 %
'LBCnty'	3.7%	42.6%

Table 5: *Shape CIRCLE: Average relative error in estimating* $\overline{nb}(\epsilon, \bigcirc)$.

Diamond - avg. relative error		
Sample sets	Proposed formula	Uniformity assumption
Synthetic Data		
'Sierpinski5K'	9.4%	74.6%
'StrLine3D'	26%	93.3%
Real Data		
'MCnty'	14.1%	82.5%
'LBCnty'	4.3%	45.8%

Table 6: *Shape DIAMOND: Average relative error in estimating* $\overline{nb}(\epsilon, \diamond)$.

($O(N \log N)$) algorithm is provided in Appendix A, by measuring the sum of squared occupancies S_2(r), for several grid sides r. Once D_2 is known, Equation (34) can be used to estimate the average number of neighbors for any query shape. Moreover, with the help of Equations (15) and (16), accurate predictions for biased range queries and for spatial joins can be made. Our analysis can also be used to provide bounds, or educated guesses, in case that a computation of D_2 is expensive.

Q3 *How expensive is the computation of* D_2 *?*
Older algorithms [Gra90] used a different definition of D_2, through the 'correlation integral' $C(\epsilon)$; this requires the enumeration of the number of pairs within distance ϵ. Given that the average number of pairs is $N(N-1)/2 \times \epsilon^{D_2}$, the complexity of such an algorithm will inevitably be O(N^2). Our algorithm (see Appendix A) reduces the complexity to O($N \log(N)$), because it uses Schuster's Lemma and computes the sum of squared occupancies ($S_2(\epsilon)$) instead; this can be achieved by a linear scanning of the points and by a (lexicographic) sorting of them. Figure 6 shows the timing results for the two approaches (elapsed time vs. database size). Both algorithms ran on a dedicated SUN SPARCstation 5. Our algorithm, in solid line, was implemented in the 'Mathematica' system. The dashed line shows the time for our implementation (in 'C') of the older algorithm, using an R-tree to accelerate the search for

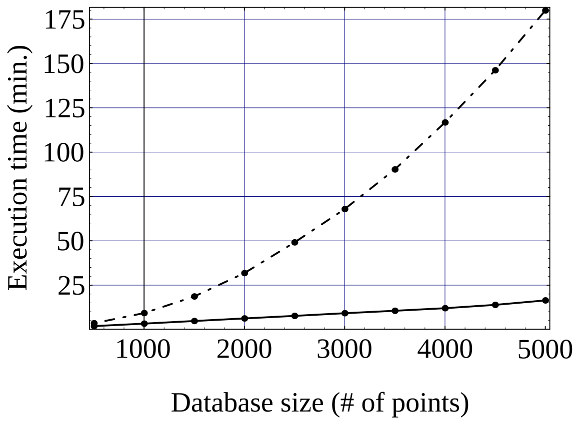


Figure 9: *Elapsed time vs. database size N, for the estimation of D_2: Our algorithm ($O(N \log N)$ - solid line); an older algorithm ($O(N^2)$ - dashed line).*

neighbors. The advantage of our algorithm is clear and it will increase with larger databases.

7 Conclusions

The major contribution of this paper is a fractal-based approach to estimate the selectivity of several spatial queries on real, *non-uniform* point sets. Query types of interest are: spatial joins, 'biased' range queries and average number of neighbors. Expanding on our previous work [FK94a] (where we used the Hausdorff fractal dimension to study range queries on R-trees), here we use the 'Correlation' fractal dimension to find accurate estimates for all the above query types. Additional, smaller contributions are:

- the discovery of Schuster's Lemma from the theory of fractals, which justifies our choice of the 'Correlation' fractal dimension D_2;
- the experimental discovery that the average number of neighbors for any query shape follows a power law (Equation (21))
- the accurate estimation of the constant of proportionality ($K_{'shape'}$) for Equation (21), justifying our Assumption 1.
- the design of a fast ($O(N \log N)$) algorithm to estimate D_2; other algorithms, by fractal theory experts [Gra90], require quadratic ($O(N^2)$) time. Our algorithm has been implemented in the Mathematica 2.2 environment and is available through ftp and mosaic (URL `ftp://olympos.cs.umd.edu/pub/SRC/fractal.dim.bundle`).

- the experimentation on real data sets, which showed that the proposed approach gives very accurate results (typically, within 10% or less), while the uniformity assumption typically leads to 40%-100% relative errors.

Future research could further exploit ideas from the theory of fractals, to solve problems in spatial databases, such as the performance of nearest neighbor queries [RKV95], and the analysis of other spatial access methods (e.g., z-ordering [Ore90]) on real, non-uniform datasets.

A Appendix: Algorithm for D_2

In this appendix we describe an algorithm for the computation of the 'correlation' fractal dimension D_2, which is based on the box-counting method proposed in [Sch91] for practical fractals (i.e. finite points-set with sef-similar behavior). We use the definition of Eq. (5) and compute the 'sum of squared occupancies' $S_2(r) = \sum {p_i}^2$ over all the cells of a grid of side r. We repeat this computation for several values of r, and we compute the slope of the line in the resulting box-count plot.

Before we give the details of the algorithm, notice that it can compute the *generalized* fractal dimension D_q, for any $q \neq 1$ (and, with some trivial modification in step 3 of $ComputeSq()$, even for $q = 1$). For the correlation fractal dimension, all we need to do is to set $q = 2$. We need the concept of '*sum of q-powered occupancies*', which is the obvious generalization of the sum of squared occupancies:

$$S_q(r) \;=\; \sum_i {p_i}^q \tag{35}$$

The algorithm for the generalized fractal dimension D_q is shown in Figure 10.

The most complicated (and time consuming) task is the estimation of $S_q(r)$. Our approach is based on the following ideas

1. each cell is identified by the coordinates of its lower-left corner (with the obvious extension for E-d spaces). Let

$$\vec{c} = (c_1, \ldots, c_E) \tag{36}$$

be a cell identifier, with c_k denoting the k-th coordinate of a cell ($k = 1, \ldots, E$)

2. for a grid-side r, the cell that a point $\vec{x} = (q_1, \ldots, q_E)$ falls into is determined by dividing each coordinate q_k by r and taking the 'floor' function

$$c_k = \lfloor q_k / r \rfloor \quad k = 1, \ldots, E \tag{37}$$

Given the above, the algorithm to compute the sum of q-powered occupancies $S_q(r)$ for a given r is given in Figure 11.

Algorithm 1 *Compute-*D_q ($\mathcal{P}$)

begin

1) normalize the address space to the unit hyper-cube
2) select a list of r values, usually, in geometric progression, eg. (0.5), $(0.5)^2$, ..., $(0.5)^n$
3) for each value of r,

   3.1) invoke $ComputeSq(r)$ to compute the sum of squared (or q-powered) occupancies $S_q(r) = \sum {p_i}^q$

   3.2) print the values $\log r$ and $\log(S_q(r))$
4) eliminate the flat parts of the box count plot,
5) perform a linear interpolation and
6) return the slope

end

Figure 10: Algorithm to compute the generalized fractal dimension D_q

Algorithm 2 $ComputeSq(r)$

begin

1) for each point $\vec{x}$ in the point-set,

   compute the cell $\vec{c}$ it falls in using Eq. (37)

   append the cell identifier $\vec{c}$ to a list $\mathcal{L}$
2) sort lexicographically the list $\mathcal{L}$
3) count the duplicates (= occupancies) for each cell, raise them to the q-th power, and compute $S_q(r)$.

end

Figure 11: Algorithm to compute the sum of q-powered occupancies.

In these algorithms, all the steps are either $O(1)$ or $O(N)$, except for the sorting step of $ComputeSq()$. This step requires sorting of the list $\mathcal{L}$ of cell-identifiers. Therefore, the complexity of our algorithm is $O(nN\log(N))$ where N is the number of points in the database and n is the number of points in the box-count plot (typically, $n \approx$ 10-20).

References

[ACF+93] Manish Arya, William Cody, Christos Faloutsos, Joel Richardson, and Arthur Toga. Qbism: a prototype 3-d medical image database system. *IEEE Data Engineering Bulletin*, 16(1):38–42, March 1993.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. *Proc. ACM SIGMOD*, pages 207–216, May 1993.

[AS91] Walid G. Aref and Hanan Samet. Optimization strategies for spatial query processing. *Proc. of VLDB (Very Large Data Bases)*, pages 81–90, September 1991.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. *Proc. of VLDB Conf.*, pages 487–499, September 1994.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD*, pages 322–331, May 1990.

[BS88] M.F. Barnsley and A.D. Sloan. A better way to compress images. *Byte*, pages 215–223, January 1988.

[CE92] M. Casgagli and S. Eubank. *Nonlinear Modeling and Forecasting.* Addison Wesley, 1992. Proc. Vol. XII.

[Chr84] S. Christodoulakis. Implication of certain assumptions in data base performance evaluation. *ACM TODS*, June 1984.

[DKPY94] D. J. DeWitt, N. Kabra, J. M. Patel, and Jie-Bing Yu. Client-server paradise. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, September 1994.

[Fal92] C. Faloutsos. Analytical results on the quadtree decomposition of arbitrary rectangles. *Pattern Recognition Letters*, 13(1):31–40, January 1992.

[FBF+94] Christos Faloutsos, Ron Barber, Myron Flickner, Wayne Niblack, Dragutin Petkovic, and William Equitz. Efficient and effective querying by image content. *J. of Intelligent Information Systems*, 3(3/4):231–262, July 1994.

[FJ92] Christos Faloutsos and H.V. Jagadish. On b-tree indices for skewed distributions. In *18th VLDB Conference*, pages 363–374, Vancouver, British Columbia, August 1992.

[FK94a] C. Faloutsos and I. Kamel. Beyond uniformity and independece analysis of r-trees using the concept of fractal dimension. *Proc. ACM SIGACT-SIGMOD-SIGART PODS*, pages 4–13, May 1994. Also available as CS-TR-3198, UMIACS-TR-93-130.

[FK94b] Christos Faloutsos and Ibrahim Kamel. Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension. *Proc. ACM SIGACT-SIGMOD-SIGART PODS*, pages 4–13, May 1994. Also available as CS-TR-3198, UMIACS-TR-93-130.

[FRM94] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. *Proc. ACM SIGMOD*, pages 419–429, May 1994. 'Best Paper' award; also available as CS-TR-3190, UMIACS-TR-93-131.

[FSR87] C. Faloutsos, T. Sellis, and N. Roussopoulos. Analysis of object oriented spatial access methods. *Proc. ACM SIGMOD*, pages 426–439 426–439, May 1987. also available as SRC-TR-87-30, UMIACS-TR-86-27, CS-TR-1781.

[Gra90] P. Grassberger. An optimized box-assisted algorithm for fractal dimensions. *Physics Letters A*, 148(1,2):pp 63–68, August 1990.

[Gut84] A. Guttman. R-trees: a dynamic index structure for spatial searching. *Proc. ACM SIGMOD*, pages 47–57, June 1984.

[IC91] Yannis E. Ioannidis and Stavros Christodoulakis. On the propagation of errors in the size of join results. *Proc. of ACM SIGMOD*, pages 268–277, May 1991.

[IC94] Yannis E. Ioannidis and Stavros Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. *ACM TODS*, 1994. to appear.

[KF94] Ibrahim Kamel and Christos Faloutsos. Hilbert r-tree: an improved r-tree using fractals. In *Proc. of VLDB Conference,*, pages 500–509, Santiago, Chile, September 1994.

[Man77] B. Mandelbrot. *Fractal Geometry of Nature*. W.H. Freeman, New York, 1977.

[MD88] M. Muralikrishna and David J. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. *Proc. ACM SIGMOD*, pages 28–36, June 1988.

[NS86] R. Nelson and H. Samet. A population analysis of quadtrees with variable node size. Tech. Report CAR-TR-241, also CS-TR-1740, DCR-86-05557, Computer Science Department, Univ. of Maryland, College Park, December 1986.

[Ore86] J. Orenstein. Spatial query processing in an object-oriented database system. *Proc. ACM SIGMOD*, pages 326–336, May 1986.

[Ore90] J.A. Orenstein. A comparison of spatial query processing techniques for native and parameter spaces. *Proc. of ACM SIGMOD Conf.*, pages 343–352, 1990.

[PSTW93] B. Pagel, H. Six, H. Toben, and P. Widmayer. Towards an analysis of range query performance. In *Proc. PODS '93*, pages 214–221, Washington, D.C., May 1993.

[RKV95] Nick Roussopoulos, Steve Kelley, and F. Vincent. Nearest neighbor queries. In *Proc. of the 1995 ACM-SIGMOD Conference*, San Jose, CA, May 1995.

[Sam90] H. Samet. *Applications of Spatial Data Structures Computer Graphics, Image Processing and GIS*. Addison-Wesley, 1990.

[Sch88] Heinz Georg Schuster. *Deterministic Chaos*. VCH Publisher, Weinheim, Basel, Cambridge, New York, 1988.

[Sch91] Manfred Schroeder. *Fractals, Chaos, Power Laws: Minutes From an Infinite Paradise*. W.H. Freeman and Company, New York, 1991.

[SFGM93] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith. The sequoia 2000 storage benchmark. In *Proc. of the 1993 ACM-SIGMOD Conference*, Washington D.C., May 1993.

[Smi92] Richard L. Smith. Optimal estimation of fractal dimension. In *Proc. Non-linear Modeling and Forecasting, SFI Studies in the Science of Complexity*, volume 12, pages 115–135. M. Casdagli and Euban - Addison-Wesley, 1992.

[Zip49] G.K. Zipf. *Human Behavior and Principle of Least Effort: an Introduction to Human Ecology*. Addison Wesley, Cambridge, Massachusetts, 1949.

Contents