Beyond Uniformity and Independence : Analysis of R-trees Using the Concept of Fractal Dimension

Christos Faloutsos[*] and *Ibrahim Kamel*

Department of Computer Science and
Institute for Systems Research (ISR)
Univ. of Maryland at College Park

February 24, 1994

Abstract

We propose the concept of fractal dimension of a set of points, in order to quantify the deviation from the uniformity distribution. Using measurements on real data sets (road intersections of U.S. counties, star coordinates from NASA's Infrared-Ultraviolet Explorer etc.) we provide evidence that real data indeed are skewed, and, moreover, we show that they behave as mathematical fractals, with a measurable, non-integer fractal dimension.

Armed with this tool, we then show its practical use in predicting the performance of spatial access methods, and specifically of the R-trees. We provide the *first* analysis of R-trees for skewed distributions of points: We develop a formula that estimates the number of disk accesses for range queries, given only the fractal dimension of the point set, and its count. Experiments on real data sets show that the formula is very accurate: the relative error is usually below 5%, and it rarely exceeds 10%.

We believe that the fractal dimension will help replace the uniformity and independence assumptions, allowing more accurate analysis for *any* spatial access method, as well as better estimates for query optimization on multi-attribute queries.

1 Introduction

In this work we study the distribution of real datasets with D-dimensional points. Sets of multi-dimensional points appear in several database applications: records in relational DBMSs can be

[*]This 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.

viewed as points in attribute space; geographic information systems (GIS) [21] contain point data, such as cities on a 2-dimensional map; medical image databases with, eg., 3-dimensional MRI brain scans, require the storage and retrieval of point-sets, such as digitized surfaces of brain structures [2], etc. For the balance of this work, the term *point-set* and *dataset* will be used interchangeably.

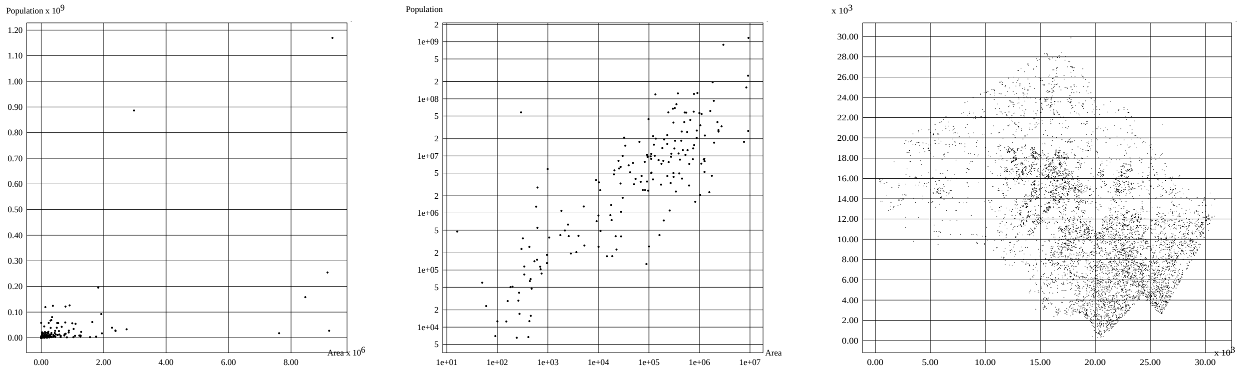
An interesting problem is the estimation of the search effort for range queries, when the points are stored in a spatial access method (SAM), such as an R-tree. A closely related problem is the estimation of selectivity (ie., number of qualifying points) for range queries: this is useful in the query optimization in an RDBMS with multidimensional queries [14] or in a GIS [1]. The traditional assumptions are the *uniformity* and *independence* assumption, which make the analysis tractable.

However, these assumptions do not hold on real data; moreover, they lead to pessimistic estimates [5]. For a single attribute, the uniformity assumption has been relaxed, eg., [10], typically using the Zipf distribution [24]. Distributions of real attributes indeed follow the Zipf distribution or the generalized Zipf distribution: for example, word frequencies in the English language (as well as other languages); salaries [24]; first names and last names of people [7], etc.

However, no attempt has been made to model multi-dimensional distributions. Theoretical analyses in such cases assume that points are uniformly distributed in the address space [6],[1], 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 [15], some sort of clustered distributions (with points clustering around uniformly distributed sites [17], or points clustering around curves, like the sinusoidal curve [4]), or even to obey the independence assumption anyway [14]. Although non-uniform, it is unclear whether these distributions resemble real distributions.

Evidence that real distributions violate both assumptions is overwhelming. Figure 1 provides a counter-example for both assumptions, using two real datasets. Figure 1(a) shows the population vs. area scatter-plot for 181 nations, from the World Factbook (available, eg., with anonymous ftp from `ocf.berkeley.edu, /ftp/pub/Library/Reference/World_Factbook`). Notice that the points follow a highly skewed distribution: there are a few large and/or highly populated nations (eg., China, India), while the vast majority of points is close to the origin. Figure 1(b) shows the same scatter plot in doubly logarithmic scales, to highlight the strong correlation between area and population.

Figure 1 (c) shows the road intersections for the Montgomery County of Maryland. Notice that the distribution is clearly non-uniform. Also notice that it does not resemble a gaussian distribution either : there are several high-concentration areas (one for each city, like Bethesda, Rockville, Gaithersburg etc).



(a) (b) (c)

Figure 1: *(a) WORLD dataset (b) WORLD dataset - log-log scale (c) Cross-roads of Montgomery County of Maryland*

In conclusion, we need a way to describe distributions of real multi-dimensional points. The problem we want to solve is the following:

GIVEN real distributions of multi-dimensional points

FIND a way to characterize their deviation from uniformity

TO PREDICT the search performance of a spatial access method (SAM).

The description should allow us to analyze several types of queries, such as 'range' queries, 'nearest neighbor' queries, 'spatial joins' etc. In this paper we focus on range queries.

For the above problem, we propose the concept of *fractal dimension* as the solution. We show that several real distributions indeed exhibit *fractal* behavior, that is, they are self-similar: portions of the point-set are statistically similar to the whole set (see Figure 3 for an example). We describe how to compute the fractal dimension of a set of points, and then we show how to use it to predict the performance of spatial access methods that store this point-set. Specifically, we examine the R-trees and provide the first known analysis for them, for real distributions.

The paper is organized as follows: Section 2 provides the background information about R-trees. Section 3 gives the definition of the fractal dimension, along with examples from real data. Section 4 presents the analysis. Section 5 gives the experimental results and Section 6 lists the conclusions.

2 Background

In this section, we provide a brief description of R-trees [9], which will be the SAM that we use to illustrate our analysis. Additional SAMs include the quad-tree based methods [8, 17] and grid file and its variants [16]; see [20] for a survey. We focus on the R-tree because it seems to be one of the most successful methods.

The original R-tree [9] was suggested by Guttman; it is the extension of the B-tree for multidimensional objects. A geometric object is represented by its minimum bounding rectangle (MBR). Non-leaf nodes contain entries of the form (ptr,R) where ptr is a pointer to a child node in the R-tree; R is the MBR that covers all rectangles in the child node. Leaf nodes contain entries of the form (obj-id, R) where obj-id is a pointer to the object description, and R is the MBR of the object. The main innovation in the R-tree is that father nodes are allowed to overlap. This way, the R-tree can guarantee at least 50% space utilization and at the same time remain balanced. Figure 2 illustrates data rectangles (in black), organized in an R-tree with fanout 3.

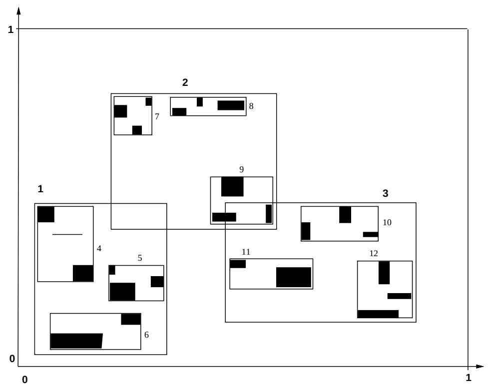


Figure 2: *Data (dark rectangles) organized in an R-tree with fanout=3*

Subsequent work on R-trees includes the packed- [19] and Hilbert-packed R-trees [12], the R^+-tree [23], R-trees using Minimum Bounding Polygons [11] and the R^*-tree [4]. The latter seems to give the best search times, mainly thanks to the idea of deferring the splits by 'force-reinserting' some of the entries of the overflowing nodes.

The last part in this section is a summary of previous attempts to analyze the R-trees. In [6] we assumed that the points are uniformly distributed in the address space. Recently, we provided formulas [12] that assume that the R-tree has been built and that we can measure the MBR of each node of the R-tree. The results are as follows. Consider the n-th node of the tree, and let its MBR be $x_{1,n} \times \ldots \times x_{D,n}$. We denote it compactly as:

$$\vec{x_n} = (x_{1,n}, \ldots, x_{D,n}) \tag{1}$$

Similarly, let

$$\vec{q} = (q_1, \ldots, q_D) \tag{2}$$

denote a range query with side q_i on the i-th dimension. Then, we have the following theorem:

Theorem 1 *The average number $P(\vec{q})$ of nodes ($\equiv$ pages) accessed by a query of sides $\vec{q}$ is given by*

$$P(\vec{q}) = \sum_n \prod_{i=1}^{D} (x_{i,n} + q_i) \tag{3}$$

where the summation extends over all the nodes of the tree.

Proof: See [12].

A similar analysis, mainly focusing on square queries, was published independently in [18]. Notice that Theorem 1 allows us to calculate the node accesses for *any* level of the tree: we only need to restrict the summation over the nodes of the level(s) of interest.

In the next sections we shall see how to *predict* the size of the MBRs of the nodes of the tree, *before* even the tree is built. To do that, we show that we only need the *fractal dimension d* of the point-set, and, of course, the number of points N and the fanout of the tree.

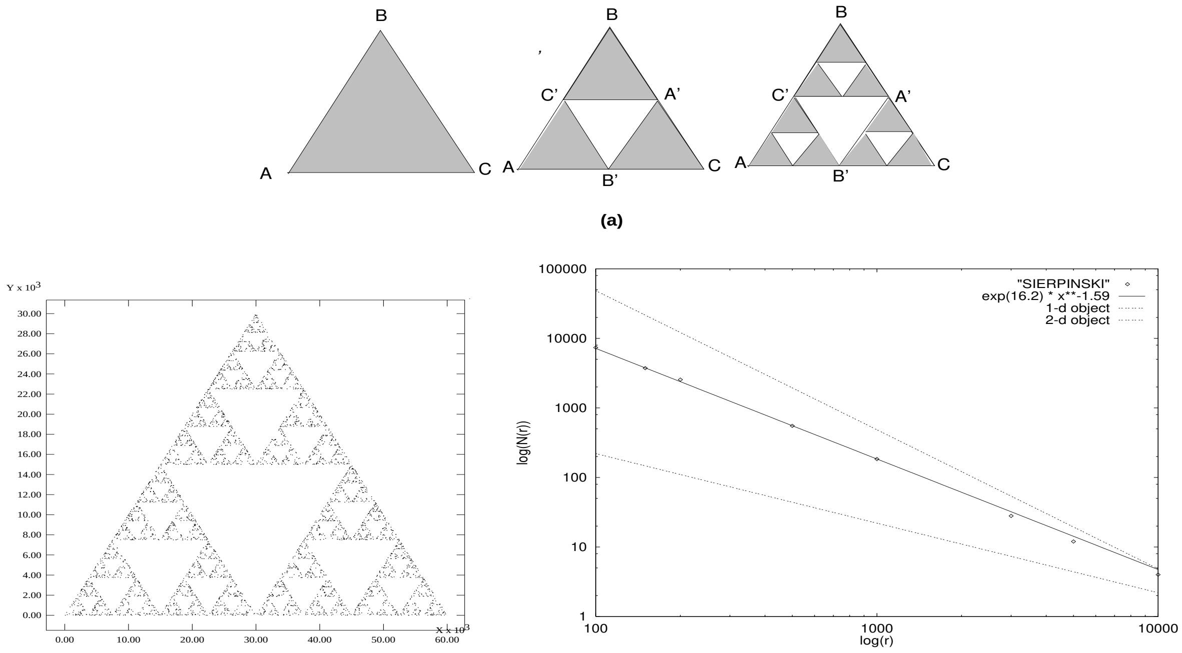
3 Fractal Dimension

Intuitively, a set of points is a fractal if it exhibits self-similarity over all scales. This is illustrated by an example: Figure 3(a) shows the first few steps in constructing the so-called *Sierpinski triangle*. Figure 3(b) gives 5,000 points that belong to this triangle. Theoretically, the Sierpinski triangle is derived from an equilateral triangle ABC, by excluding its middle (triangle A'B'C') and 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 50,000 sample points from the Sierpinski triangle ('SIERPINSKI' dataset).

The Sierpinski triangle gives an example of points which follow a highly non-uniform distribution; yet, the distribution is deterministic, and easy to describe, at least in English. There should be an equally easy, way to describe it mathematically.

3.1 Formal definitions and Measurements

Consider a geometrical object (eg., like the Sierpinski triangle) with the 'self-similarity' property, consisting of a set of points in D-dimensional space. The dimensionality D of the address space is



(b) (c)

Figure 3: *(a)Three steps in generating sierpinski triangle (b) 5000 points forming Sierpinski triangle (c) The slope of the solid line gives the fractal dimension of Sierpinski triangle* $\approx$ *1.59*

defined as the *embedding* dimension.

The *fractal dimension* d, (or, more accurately, the *box-counting fractal dimension*) is defined as follows [22]: Divide the D-dimensional space into (hyper-)cubic grid cells of side r. Let $N(r)$ denote the number of cells that are penetrated by the fractal (i.e., contain 1 or more points of it). Then the (box-counting) fractal dimension d of a fractal is defined as

$$d \equiv \lim_{r \to 0} \frac{\log N(r)}{\log(1/r)} \tag{4}$$

This definition is useful for mathematical fractals, that consist of infinite number of points. For a finite sample of points that belong to a fractal, we use the *boxcount-side* plot to estimate the fractal dimension. By *boxcount-side* plot we mean the plot of $\log(N(r))$ vs $\log(r)$. Its usefulness lies in the fact that, if the point-set is self-similar, the plot is almost a straight line. Moreover, the *slope* of the line is the (negated) fractal dimension of the point-set. This is the standard way to measure the fractal dimension for real point-sets [22]:

Definition 1 *For a point-set that has the self-similarity property, its fractal dimension d is measured as*

$$d \;=\; -\frac{\partial \log(N(r))}{\partial \log(r)} \;=\; constant \tag{5}$$

Corollary 1 *For a point-set with the self-similarity property, we have*

$$N(r) = K/r^d \tag{6}$$

where K is an integration constant.

Proof: Since d remains constant with r, we obtain Eq. 6 by integrating Eq. 5.

Next we give some examples to illustrate how the method works.

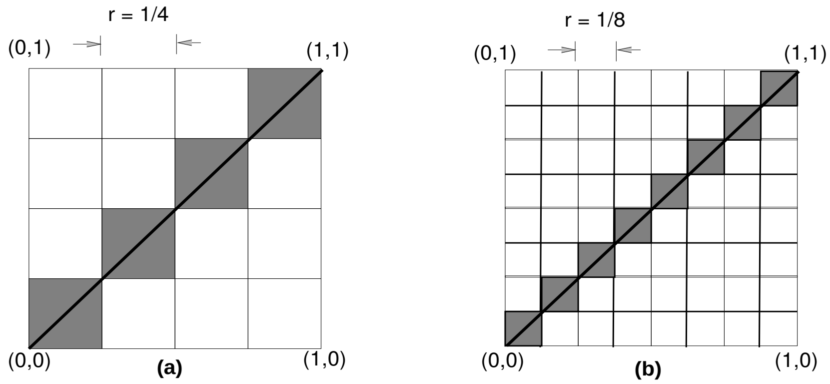


Figure 4: *Computing the fractal dimension of a line segment*

Example 1: Figure 4 illustrates the method when the object is the diagonal line segment (0,0)-(1,1). Figure 4(a) shows that the line segment penetrates 4 boxes of side $r = 1/4$, and Figure 4(b) shows it penetrating 8 boxes of side $r = 1/8$. Thus, we have

$$N(r) = (1/r) = (1/r)^1 \quad r \leq 1 \tag{7}$$

and, therefore,

$$d = -\frac{\partial \log((1/r)^1)}{\partial \log(r)} = 1 \tag{8}$$

which is intuitively expected, since a line segment is a 1-dimensional object.

The fact that the fractal dimension of a line segment reduces to its Euclidean dimension is not a coincidence: Notice that Euclidean objects, like lines, circles, planes etc. trivially fulfill the self-similarity requirement: for example, a part of a line segment is a miniature replica of the whole segment. Based on the above, we have [13]:

Observation 1 *For Euclidean objects, their fractal dimension reduces to their Euclidean dimension.*

Thus, lines, line segments, circles, and all the standard curves have d=1; planes, disks and standard surfaces have d=2; euclidian volumes in D-dimensional space have $d = D$.

Example 2. This is an example of a fractal with a non-integer d. Figure 3-(c) shows the boxcount-side plot for the SIERPINSKI set of points of Figure 3-(b) The line has slope (-)1.59, while the theoretical number is $\log 3/\log 2 = 1.5849$ [13]. The plot also contains two lines. The former corresponds to a 2-dimensional object (like unit square) and the latter corresponds, to a 1-dimensional object, such as the line segment of example 1, with slopes -2 and -1 respectively.

3.2 Fractal dimensions of real datasets

In the above examples, all the involved point-sets are known to have the self-similarity property, and therefore it is expected that their boxcount-side plots will be straight lines. The question is whether *real* datasets exhibit self-similarity. Table 1 shows the characteristics of the datasets we used:

The 'MGCounty' and 'LBCounty' datasets are part of the TIGER database of the US Bureau of Census and they contain the road intersections of the Montgomery county, MD and Long Beach county, CA, respectively. The 'IUE' contains observation points (latitude and longitude of stars) from the International Ultraviolet Explorer (IUE) satellite of NASA. The WORLD dataset has been discussed in the introduction (see Figure 1).

In the upcoming plots we also used synthetic datasets: In addition to the SIERPINSKI one that we have discussed we also used the 2D-UNIFORM, 1D-Uniform point sets. As their names reveal, they consisted of points uniformly distributed in the unit square and on a line, respectively

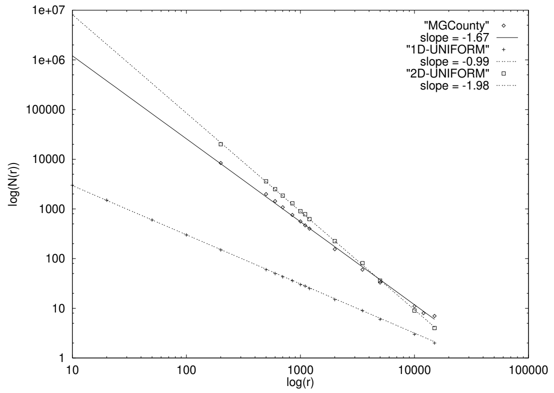
Figure 5 (a)-(d) shows the boxcount-side plots for all these datasets. Notice that the plots are indeed straight lines, confirming that real point-sets exhibit fractal behavior. This implies that we can use Eq. 6 to do useful predictions, as we show next. In all of figures 5 (a)-(d) we also give the boxcount-side plot for the two uniform synthetic datasets, 2D-UNIFORM and 1D-UNIFORM. The reason is that we want to highlight the fact that the slopes for the real sets are non-integers. This means that the real datasets are self similar, but clearly non-uniform.

The conclusion is that we have confirmed once more that real data sets are far from uniformly distributed - the only difference is that, this time, we have a usable measure of their skewness!

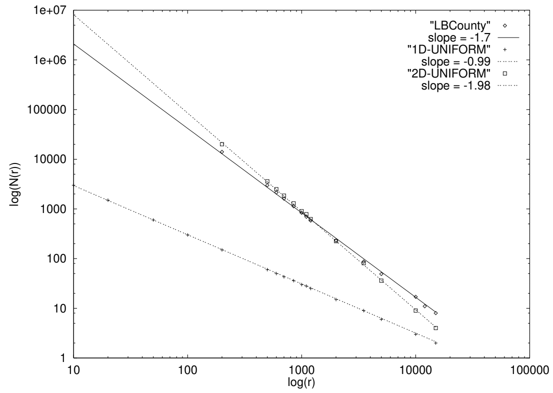
Before we continue with the analysis, we list a few more observations:

Observation 2 *The fractal dimension can model point-sets with highly correlated attributes, even if the correlation is non-linear.*

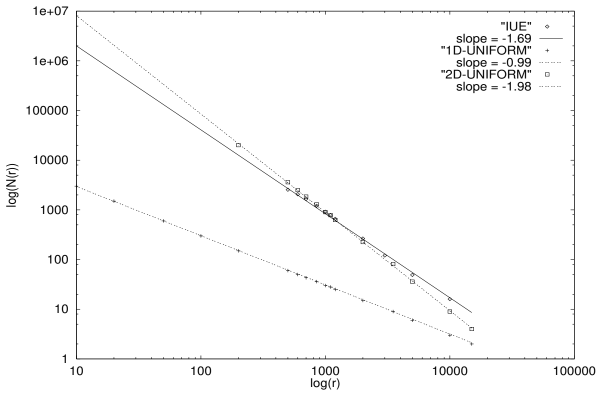
The reason is that, if two attributes are strongly correlated (even in a quadratic, logarithmic or some other *non-linear* fashion), the resulting set of points in attribute space will be a curve, with d=1, as we just discussed in Observation 1. The input point-set will be correctly characterized as a linear object, as opposed to a 2-dimensional object.



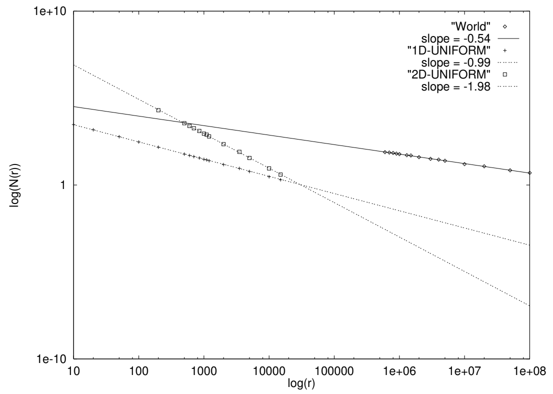
(a) MGCounty - slope = -1.67



(b) LBCounty - slope = -1.70



(c) IUE - slope = -1.69



(d) WORLD - slope = -0.54

Figure 5: *'boxcount-side' plots for real datasets (solid lines); slopes -1 and -2 (dotted lines)*

Observation 3 *A set of D-dimensional points with attributes that obey the 'uniformity' and 'independence' assumptions has fractal dimension equal to the number of attributes:*

$$d = D \tag{9}$$

This is justified, because the points cover the whole space, forming practically a D-dimensional solid. Thus, a data set with 'uniform' and 'independent' attributes can be trivially modeled as a fractal, with fractal dimension $d = D$.

4 Analysis of R-trees

In the previous section, we claimed that real datasets will obey Eq. 6:

$$N(r) = \frac{K}{r^d}$$

Name	Description	number of points	d
MGCounty	road intersections, Montgomery Cty, MD	79,438	1.67
LBCounty	road intersections, Long Beach Cty, CA	68,849	1.70
IUE	star coordinates (NASA's Infrared-Ultraviolet Explorer)	11,281	1.69
WORLD	population vs. area of all countries - World Almanac	181	0.54
2D-UNIFORM	uniformly dist. points on a plane - synthetic	50,000	1.98
SIERPINSKI	points on Sierpinski triangle - synthetic	50,000	1.59
1D-UNIFORM	uniformly dist. points on a line - synthetic	50,000	0.99

Table 1: Datasets used

where K is a constant. Assuming that the address space is normalized to the unit (hyper)-cube, we have that $K = 1$, or equivalently

$$N(r) = \frac{1}{r^d} \tag{10}$$

where d is the (box counting) fractal dimension of the dataset. In this section we want to derive a formula to predict the number of disk accesses for range queries, when these points are stored in an R-tree. First we estimate the number of leaf nodes only; then we extend the analysis for all levels of the R-tree. We define the *effective capacity* C_{eff} of the nodes of the R-tree as the average number of entries per node:

$$C_{eff} = C \times u \tag{11}$$

where u is the average node utilization (typically, 70% for the R*-trees).

Thus the problem is as follows:

Given: N: the number of points in D-dimensional space

d: their fractal dimension

C_{eff}: the effective capacity of the nodes

Find: the expected size $\vec{s} = (s_1, s_2, \ldots, s_D)$ of the MBRs for the leaf nodes. That is, the average MBR will be a hyper-rectangle of size $s_1 \times s_2 \times \ldots \times s_D$

Once we have estimated $\vec{s}$, we can immediately use Eq. 3 to calculate the expected number of leaf accesses for any given query $\vec{q}$. The number of leaf nodes is

$$N_l = N / C_{eff} \tag{12}$$

We make the following (optimistic) assumption:

Assumption A1: The algorithms of the R-tree are 'good', in the sense that they will result in tight, square-like MBRs, roughly of the same size. That is

$$s_1 = s_2 = \ldots = s_D \equiv \sigma \tag{13}$$

From this assumption we expect that the MBRs of the leaf nodes will be roughly similar, square-like hyper-rectangles of side σ. Notice that this setting resembles very much the setting of the definition of the fractal dimension: We have N_l boxes of side σ, that cover all the points of the dataset. Assuming that the address space is normalized to the unit D-dimensional cube, we can use Eq. 10 as follows:

$$N_l = 1/\sigma^d \tag{14}$$

$$N/C_{eff} = 1/\sigma^d \tag{15}$$

$$\sigma = (\frac{C_{eff}}{N})^{1/d} \tag{16}$$

Combining Eq. 3 with the above equation, we can estimate the number of leaf accesses $P(\vec{q})$ for a query $\vec{q}$:

$$P(\vec{q}) = \sum_{all\ leaf\ nodes} \prod_{i=1}^{D} (\sigma + q_i) \tag{17}$$

or

$$P(\vec{q}) = \frac{N}{C_{eff}} \prod_{i=1}^{D} (\sigma + q_i) \tag{18}$$

We have just shown how to estimate the node accesses at the leaf level. The analysis can be similarly extended to any level of the R-tree. Assuming that the average fanout is C_{eff} at every level, we can estimate the number of nodes N_j at each level j, as well as the side σ_j of the (D-dimensional cubic) MBRs. The final formula for the total number of nodes accessed $P_{all}(\vec{q})$ is given by adding the node accesses at each level. Thus

$$P_{all}(\vec{q}) = \sum_{j=0}^{h-1} \frac{N}{{C_{eff}}^{h-j}} \prod_{i=1}^{D} (\sigma_j + q_i) \tag{19}$$

where h is the height of the tree (the root is assumed at level $j = 0$ and the leaves at level $j = h-1$); and σ_j given by:

$$\sigma_j = (\frac{{C_{eff}}^{h-j}}{N})^{1/d} \quad j = 0, \ldots, h-1 \tag{20}$$

Symbols	Definitions
C	max. number of rectangles per page(= node)
C_{eff}	effective page capacity $= C \times u$
d	fractal dimension
D	embedding dimension (= # of attributes/axes)
h	height of the R-tree
N	number of data rectangles
N_l	number of leaf nodes
$\vec{q}$	query hyper-rectangle $q_1 \times q_2 \times \ldots \times q_k$
q_i	length of the query in the i-th dimension
$P(\vec{q})$	avg. leaf pages retrieved by a query $\vec{q}$
$P_{all}(\vec{q})$	avg. pages (at all levels) retrieved by a query $\vec{q}$
σ_j	side of the ($\approx$)square MBR of a node at level j
u	avg. node utilization

Table 2: Summary of Symbols and Definitions

5 Experimental results

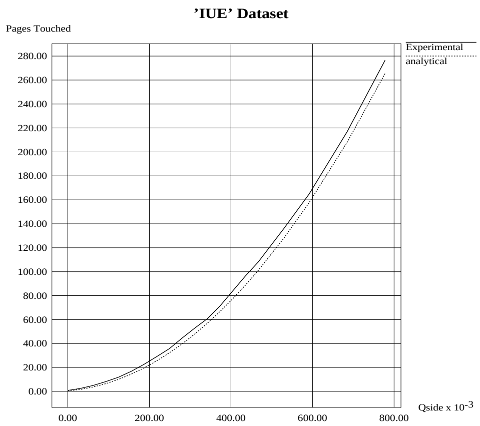
We carried out several experiments, to compare our analytical results with the results of an R*-tree [4]. The R*-tree was written in C under UNIX and the experiments ran on DEC 5000 workstation.

Except for the 'WORLD' dataset, which was too small, we used all the real and synthetic datasets that we used in section 3. Their characteristics are summarized in Table 1.

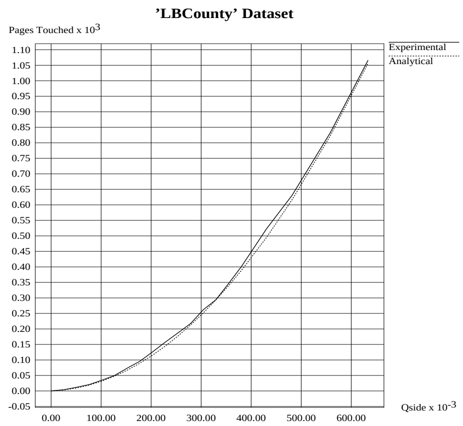
In all cases, the address space was normalized to the unit square. The queries were squares with side varying from 0 to 0.8. For each query side we report the average response time over 1000 uniformly distributed queries. Queries that were not completely inside the address space were 'wrapped around'.

We ran two sets of experiments: in the first, we measured only the leaf accesses and in the second we measured node accesses at *all* levels.

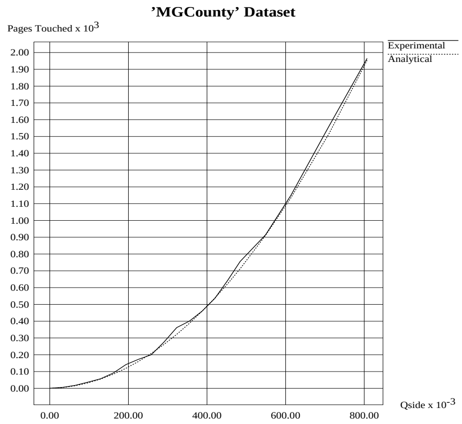
The results of the first set are plotted in Figures 6(a)-(e). For the predictions we used Eq. 18. We put more emphasis on the leaf accesses for two reasons: (a) the majority of the node accesses will be on leaf nodes and (b) most of the non-leaf nodes are likely to fit in main memory. Figures 6 (a), (b) and (c) show the number of leaf accesses vs. the query side q, for the real datasets (IUE, LBCounty and MGCounty respectively). They plot the actual results with a solid line and the predicted ones with a dotted line. Figures 6 (d) and (e) show the same measurements for the synthetic datasets (2D-UNIFORM and SIERPINSKI respectively). The common observation in all



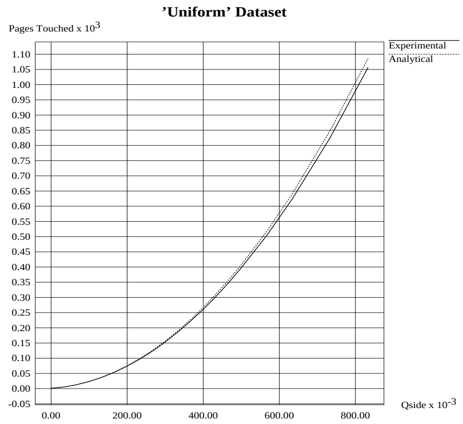
(a) IUE - Leaf accesses vs. query side



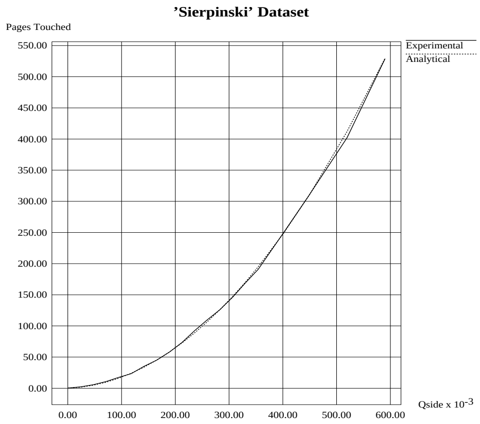
(b) LBCounty - Leaf accesses vs. query side



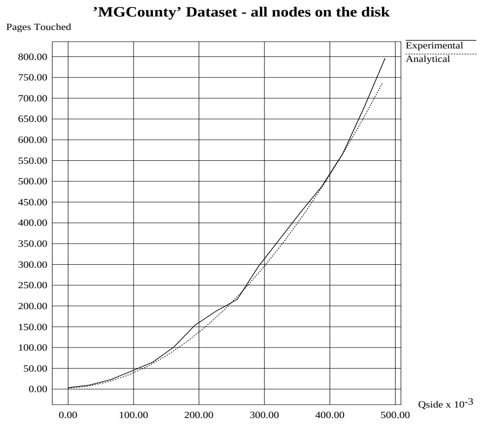
(c) MGCounty - Leaf accesses vs. query side



(d) 2D-UNIFORM - Leaf accesses vs. query side



(e) SIERPINSKI - Leaf accesses vs. query side



(f) MGCounty - total node accesses vs. query side

Figure 6: *Real response time vs. analytical for different query sizes*

the graphs is that the analytical estimate is very close the actual result: the relative error is usually below 5% and rarely above 10%.

The second set of experiments test the accuracy of Eq. 19, which computes the number of node accesses at all levels. This will translate to the actual number of disk accesses, in case that all the levels of the tree reside on disk. The results were similar for all the datasets. For brevity, we present only the experiments with the MGCounty dataset, in Figure 6(f). Again, our analysis gives accurate predictions, with relative error usually below 7% and rarely above 12%.

6 Discussion - Conclusions

There are two contributions in this work. The major one is the proposal to use the fractal dimension to quantify the skewness of real point sets. Up to now, the 'uniformity' and 'independence' assumptions have been (rightly) challenged; however no satisfactory multi-dimensional distribution models have been proposed to replace them. We showed that the fractal dimension provides an excellent measure of the deviation from the above assumptions.

The fractal dimension has several desirable characteristics:

- it constitutes a simple way to describe the non-uniformity of the data set, using just a single number.
- it is applicable to real point-sets, as our experiments showed.
- it includes the uniform distribution as a special case ($d = D$).
- it is based on a well developed theory [13, 22]

In addition to the above theoretically pleasing properties, we showed that the fractal dimension has practical applications in the performance analysis of spatial access methods on real datasets. Using it, we provided the *first* analysis of R-trees on real data; the resulting formula is simple, and, as showed experimentally, it is very accurate, usually within 5% of the experimental results. This is the second contribution of this work: Despite the fact that R-trees are known for almost a decade, there has been only one attempt for their analysis [6]; even that one used the uniformity assumption, pressumably leading to pessimistic estimates. The current analysis superseeds the old one, since, according to (Observation 3), the uniform case is just a special case of a fractal distribution.

We believe that the fractal dimension will become a powerful modeling tool for multi-variate distributions in relational and spatial databases. Future work could examine its potential applications, such as:

- Analysis of other spatial access methods, such as quadtrees/octrees, grid files etc. For example, in [3] we stored the 3-dimensional MRI-scans of human brains using an oct-tree decomposition; we observed that the number of octants required to cover the surface of human brains increased exponentially with the resolution, with an exponent of 2.6 (close to the fractal dimension 2.7 of mammal brains [13],p.113). Thus, knowledge of the fractal dimension of a surface (or set of points, in general), is useful in the prediction of the storage requirements for the resulting quadtrees/octrees.

- Query optimization, for multi-attribute queries and for geometric/geographic databases. For example, Eq. 18 can be used to predict the selectivity of a range query (number of qualifying points) by setting the leaf capacity C=1.

- Generation of synthetic but realistic data, to study the performance of spatial access methods: instead of generating points that follow the uniform distribution, or gaussian or some other ad-hoc distribution, we propose to generate points that have the desirable fractal dimension. Methods to generate point-sets with a given fractal dimension are described, eg., in [13](chapter 32), using the so-called 'Lèvy flights'.

Acknowledgement : we would like to thank Erik Hoel for providing MGCounty and LBCounty datasets.

References

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

[2] 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.

[3] Manish Arya, William Cody, Christos Faloutsos, Joel Richardson, and Arthur Toga. Qbism: Extending a dbms to support 3d medical images. *Tenth Int. Conf. on Data Engineering (ICDE)*, February 1994. (to appear).

[4] 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.

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

[6] 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.

[7] 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.

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

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

[10] 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.

[11] H. V. Jagadish. Spatial search with polyhedra. *Proc. Sixth IEEE Int'l Conf. on Data Engineering*, February 1990.

[12] Ibrahim Kamel and Christos Faloutsos. On packing r-trees. *Second Int. Conf. on Information and Knowledge Management (CIKM)*, November 1993. to appear.

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

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

[15] 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.

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

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

[18] 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.

[19] N. Roussopoulos and D. Leifker. Direct spatial search on pictorial databases using packed r-trees. *Proc. ACM SIGMOD*, May 1985.

[20] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1989.

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

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

[23] T. Sellis, N. Roussopoulos, and C. Faloutsos. The r+ tree: a dynamic index for multi-dimensional objects. In *Proc. 13th International Conference on VLDB*, pages 507–518, England,, September 1987. also available as SRC-TR-87-32, UMIACS-TR-87-3, CS-TR-1795.

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

Contents