Deflating the Dimensionality Curse using Multiple Fractal Dimensions

Bernd-Uwe Pagel
SAP AG
Walldorf, Germany
`Bernd-Uwe.Pagel@sap-ag.de`

Flip Korn
AT&T Labs - Research
Florham Park, NJ
`flip@research.att.com`

Christos Faloutsos[*]
CMU
Pittsburgh, PA
`christos@cs.cmu.edu`

Abstract

*Nearest neighbor queries are important in many settings, including spatial databases (*Find the k closest cities*) and multimedia databases (*Find the k most similar images*). Previous analyses have concluded that nearest neighbor search is hopeless in high dimensions, due to the notorious "curse of dimensionality". However, their precise analysis over real data sets is still an open problem.*

The typical and often implicit assumption in previous studies is that the data is uniformly distributed, with independence between attributes. However, real data sets overwhelmingly disobey these assumptions; rather, they typically are skewed and exhibit intrinsic ("fractal") dimensionalities that are much lower than their embedding dimension, e.g., *due to subtle dependencies between attributes. In this paper, we show how the Hausdorff and Correlation fractal dimensions of a data set can yield extremely accurate formulas that can predict I/O performance to within one standard deviation.*

The practical contributions of this work are our accurate formulas which can be used for query optimization in spatial and multimedia databases. The theoretical contribution is the 'deflation' of the dimensionality curse. Our theoretical and empirical results show that previous worst-case analyses of nearest neighbor search in high dimensions are over-pessimistic, to the point of being unrealistic. The performance depends critically on the intrinsic ("fractal") dimensionality as opposed to the embedding dimension that the uniformity assumption incorrectly implies.

1 Introduction

The problem we consider is that of nearest neighbor queries. For example, the query *'Find the closest city to San Diego,'* might return *'Los Angeles'*. More formally, a 1-nearest neighbor query is defined as follows. Given a data set $\mathcal{S}$ of N points and a query q, find the closest object n, *i.e.*,

$$NN(q) = \{n \in \mathcal{S} \mid \forall p \in \mathcal{S} : ||q - n|| \leq ||q - p||\}.$$

A k-nearest neighbor query finds the $k \leq N$ closest points. Henceforth, we shall use the term nearest neighbor query. Nearest neighbor queries are useful in several applications: GIS, where we want to find, *e.g.*, the k nearest hospitals from the place of an accident. Such queries are useful in urban planning, decision making, *etc.*; information retrieval, where we want to retrieve similar documents; multimedia databases, where objects (*e.g.*, time series) are transformed into n-d points, by extracting n features (such as the first n Discrete Fourier Transform (DFT) coefficients [1]). Thus, a query of the form *find k objects similar to the query object Q* becomes the query *find the k nearest neighbors to the query point q*. This approach has been used in several other settings: for medical images [24], video [23], *etc.*; and DNA databases [32].

Analyzing query performance in spatial access methods is important for query optimization and for evaluating access method designs. Unlike more complex queries such as spatial join, nearest neighbor queries are I/O-bound. Most I/O cost models unrealistically assume uniformity and independence to make the analysis tractible (for a recent example, see [7]). However, real data overwhelmingly disobey these assumptions; they are typically skewed and often have subtle dependencies between attributes, causing most cost models to give inaccurate, pessimistic estimates [12]. In this paper, we derive formulas to estimate the number of I/Os for nearest neighbor search in an R-tree [17] and its variants (*e.g.*, R*-tree [2]), for real world data sets. Furthermore, our analysis incorporates a model of a typical (and nonuniform) workload using the so-called *biased*

[*]This material is based upon work supported by the National Science Foundation under Grants No. IRI-9625428, DMS-9873442, IIS-9817496, and IIS-9910606, and by the Defense Advanced Research Projects Agency under Contract No. N66001-97-C-8517. Additional funding was provided by donations from NEC and Intel. Additional funding was provided by donations from NEC and Intel. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA, or other funding parties.

query model [27], which assumes that queries are more likely to be posed in heavily populated regions.

The "curse of dimensionality" is a notorious problem in high dimensional indexing whereby performance degrades exponentially as a function of dimensionality. It has attracted a lot of recent interest [25, 9, 5, 22, 6, 8, 34]. Although no convincing lower-bound has been shown, the conventional wisdom in the theory community is that a solution does not exist for worst-case search time polynomial in d with linear storage [11]. Some empirical studies have observed barely sublinear search time in practice; in fact, in high dimensions, performance can degrade to worse than exhaustive search [9].

As we argue in this paper, these results are over-pessimistic when the intrinsic dimensionality of a data set is significantly lower than its embedding dimension. We show both analytically and experimentally that the fractal dimension, rather than the dimension of the space containing the data set, is the real determinant of performance. In this sense, the dimensionality curse is 'deflated' when the intrinsic dimensionality is significantly lower than the embedding dimension. Our formulas take as input five parameters: the number of data points N, the number of neighbors k, the effective bucket capacity C, the dimension of the embedding space E, and two measures of the intrinsic dimensionality, namely, the Hausdorff dimension D_0 and the Correlation dimension D_2.

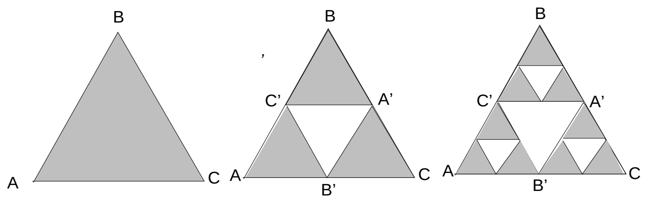
The paper is organized as follows. Section 2 gives some background on fractals and fractal dimensions and reviews some formulas for spatial selectivity estimation. Sections 3 and 4 give our formulas, with associated derivations and proofs. Section 5 presents empirical results from experiments. Section 6 lists the conclusions.

2 Background

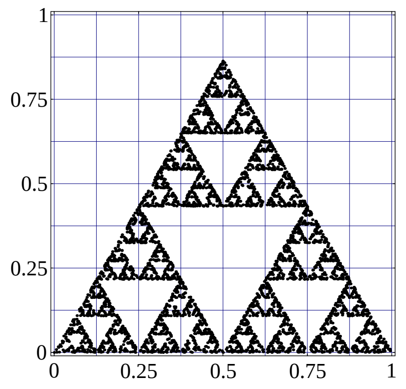
Here we give a brief introduction to fractals and review some work in the analysis of spatial access methods and spatial selectivity estimation.

2.1 Fractals and Fractal Dimensions

A set of points is a fractal if it exhibits self-similarity over all scales. This is illustrated in Figure 1(a), which shows the first few steps in the recursive construction of the so-called *Sierpinski triangle*. Figure 1(b) gives 5,000 points that belong to this triangle. 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 essence of fractals is this *self-similarity* property: parts of the fractal are similar (exactly or statistically) to the whole fractal.



(a) recursive construction



(b)`Sierpinski` data set

Figure 1. Sierpinski triangle: (a) first 3 steps of its recursive construction; (b) sample of 5,000 points based on recursive rule.

Next we present two measures of the so-called fractal dimension of a point set. For algorithms to compute them, see [4].

Definition 1 (Hausdorff fractal dimension) *Given a point set embedded in an E-dimensional space, divide this 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). For a point set that has the self-similarity property in the range of scales $r \in (r_1, r_2)$, its Hausdorff fractal dimension D_0 for this range is measured as*

$$\boxed{D_0 \equiv -\frac{\partial \log(N(r))}{\partial \log(r)} = \text{constant} \quad r \in (r_1, r_2)} \tag{1}$$

Definition 2 (Correlation fractal dimension) *For a point set that has the self-similarity property in the range of scales (r_1, r_2), its Correlation fractal dimension D_2 for this range is measured as*

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

where p_i is the percentage of points which fall inside the ith cell.

An interesting observation is that the above definitions encompass traditional Euclidean objects, that is, the fractal dimension of Euclidean objects equals their Euclidean dimension. Thus, lines, line segments, circles, and all the standard curves have D=1; planes, disks and standard surfaces have D=2; *etc.*

2.2 Spatial Access Methods

Spatial access methods and specifically R-trees [17] have been the subject of many papers, *e.g.*, R+-trees [31], R*-trees [2], and Hilbert R-trees [21]. See [16] for a survey. Recently, the dimensionality curse has attracted a lot of interest, and thus there have been many proposals for spatial access methods geared specifically towards high dimensional indexing [25, 9, 5, 6, 8, 34].

Formal worst-case analyses of performance of R-tree queries in high dimensions have yielded some very pessimistic results [18, 34, 10]. The focus of the latter two are on nearest neighbor queries, where the analysis in [34] makes assumptions of uniformity and independence, and where the analysis in [10] applies to very restricted conditions of dependence between dimensions.

2.3 Selectivity Estimation

Selectivity estimation in R-trees has also attracted much interest. The focus of [15] and [33] is on estimating disk accesses for range queries in R-trees. Selectivities for spatial joins using fractals are studied in [4]. Closer work to ours is that of [7] and [13], in which cost models for nearest neighbors are presented. In the former, assumptions of independence are implicit in the formulas. In the latter, a cost function was provided, assuming that the distance distribution $F(x)$ from an 'anchor' object to the rest of the objects is given, and is the same for *any* object that we choose as anchor. However, their formula needs statistical estimation of $F(x)$, and is unable to make any comment with respect to the embedding dimensionality E. In contrast, our analysis shows that E does not play a role; rather, it is the Hausdorff and Correlation fractal dimensions that really matter. Thanks to their simplicity, our formulas allow for easy extrapolations and asymptotic analysis that the formulas in [7, 13] cannot.

To the best of our knowledge, the only nearest neighbor analysis that uses fractals is in [29], which gives upper and lower bounds for 1-nearest neighbor queries. In contrast, here we tackle the problem of k-nearest neighbor queries and give a single and more accurate average value. Moreover, our formulas can be simplified, and thus lead to fundamental observations that deflate the 'dimensionality curse'.

symbol	*definition*
N	number of points
k	number of nearest neighbors
E	embedding dimension
D	intrinsic dimensionality
D_0	Hausdorff fractal dimension
D_2	Correlation fractal dimension
h	height of R-tree
C	effective bucket capacity
σ_j	avg side length of R-tree node on level j
$d_{nn}^{L_\infty}(k)$	avg L_∞ distance to k-th nearest neighbor
$d_{nn}^{L_2}(k)$	avg L_2 distance to k-th nearest neighbor
$an_{mbr}^{L_\infty}$	avg number of MBR's sensitive anchors wrt L_∞-norm
$an_{mbr}^{L_2}$	avg number of MBR's sensitive anchors wrt L_2-norm
$P_{all}^{L_\infty}(k)$	avg pages retrieved by k-nn query (L_∞-norm)
$P_{all}^{L_2}(k)$	avg pages retrieved by k-nn query (L_2-norm)

Table 1. Symbol table.

From the previous work, we make use of the following two results.

Lemma 2.1 *In a well-constructed R-tree, the MBRs are square-like regions of roughly the same size. The sides σ_j of these hyper-rectangles at each level j can be estimated using the Hausdorff dimension D_0 as follows:*

$$\sigma_j = \left(\frac{C^{h-j}}{N}\right)^{\frac{1}{D_0}}, \qquad j = 0, \ldots, h-1 \tag{3}$$

where h is the height of the tree and C is the effective bucket capacity.

Proof: See [14]. □

Theorem 2.2 *Given a set of points $\mathcal{P}$ with finite cardinality N embedded in a unit hypercube of dimension E and its Correlation dimension D_2, the average number of neighbors $\overline{nb}(\epsilon, \text{'}E\text{-d shape'})$ of a point within a region of regular shape and radius ϵ is given by*

$$\overline{nb}(\epsilon, \text{'}E\text{-d shape'}) = (N-1) \cdot Vol(\epsilon, \text{'}E\text{-d shape'})^{\frac{D_2}{E}} \tag{4}$$

*where $Vol(\epsilon, \text{'}E\text{-d shape'})$ is the volume of a shape (*e.g.*, cube, sphere) of radius ϵ.*

Proof: See [3]. □

3 Proposed L_∞ Nearest Neighbor Formulas

Because the analysis is more straightforward, we first consider the L_∞ distance between points, that is, $d(x,y) = \max_{1 \le i \le E} |x_i - y_i|$. Moreover, nearest neighbor search based on the L_∞-norm has been used in the literature (*e.g.*, see [24]). We then proceed to the L_2 case. Both cases are analyzed similarly.

An outline of our analysis is given as follows. First, we determine how far away from the anchor point of the query the kth nearest neighbor is located on average, in effect, transforming a nearest neighbor query into an equivalent range query. Based on this distance, we compute the number of query anchors which cause a typical page to be accessed. Since in the biased model data points are chosen as query anchors and each point is equally likely to be taken, this directly gives us the access probability of that page. By summing up over the access probabilities of all R-tree pages, we finally get the average number of page accesses, that is, the average k-nearest neighbor performance of the structure.

Lemma 3.1 *The average L_∞-distance of a data point to its kth nearest neighbor is given by*

$$d_{nn}^{L_\infty}(k) = \frac{1}{2} \cdot \left(\frac{k}{N-1}\right)^{\frac{1}{D_2}}. \tag{5}$$

Proof: In order to reach all neighbors $\vec{n}$ of a query anchor $\vec{q}$ with L_∞-distance less than ϵ, that is, $\|\vec{q} - \vec{n}\|_\infty = \max_{i=1,\dots,E} |q_i - n_i| < \epsilon$, a hypercube of radius ϵ (of side length 2ϵ) centered at $\vec{q}$ has to be explored. On the other hand, the average number of neighbors of a data point in a hypercube of radius ϵ is given by

$$\overline{nb}(\epsilon, \text{‘}E\text{-}d\ cube\text{'}) = (N-1) \cdot Vol(\epsilon, \text{‘}E\text{-}d\ cube\text{'})^{\frac{D_2}{E}} \tag{6}$$

due to Eq. 4. Substituting $Vol(\epsilon, \text{‘}E\text{-}d\ cube\text{'})$ by the volume formula for hypercubes, $(2\epsilon)^E$, as well as $\overline{nb}(\epsilon, \text{‘}E\text{-}d\ cube\text{'})$ by k, and performing some simple algebraic manipulations to solve the resulting equation for ϵ yields the desired result. □

Lemma 3.1 provides the average search radius around an arbitrary data point in order to find its k nearest neighbors. Since $d_{nn}^{L_\infty}(k)$ is a constant, the nearest neighbor problem hence turns into a specific range query problem from the point of an average analysis, namely, asking square range queries of radius $d_{nn}^{L_\infty}(k)$, where each data point is equally likely to be chosen as square center. In [27] it is shown that all anchors of square range queries with radius ϵ which force a certain page to be accessed (*query-sensitive anchors*, for short) must be located within the page's Minimum Bounding Rectangle (MBR) dilated (à la Minkowski sum) by a frame of width ϵ. The next lemma gives the average number of data points located in such a region if the MBR is roughly a square by itself.

Lemma 3.2 *Assume the biased model, square range queries of radius ϵ (= nearest neighbor queries with L_∞ search radius ϵ). Then the average number of query-sensitive anchors of an MBR with side length l is given by*

$$an_{mbr}^{L_\infty}(l, \epsilon) = (N-1) \cdot (l + 2\epsilon)^{D_2} + 1. \tag{7}$$

Proof: We assume the center of the MBR to be a data point. Since every MBR wraps a large number of data points on average, it can be expected that there exists a data point within a very small distance to the center. The inflated MBR (with the query-sensitive anchors in it) is a square of side length $l + 2\epsilon$, so that we can directly use Eq. 4 to estimate the number of data points falling into it. Computing $Vol(l + 2\epsilon, \text{‘}E\text{-}d\ cube\text{'})$, substituting the term in Eq. 4, and adding the MBR's center as a further anchor completes the proof. □

We are ready now to estimate the number of page accesses needed to answer a k-nearest neighbor query (that is, the query performance) with respect to the L_∞-norm.

Theorem 3.3 *Assume a set of N points with Hausdorff dimension D_0 and Correlation dimension D_2 indexed by an R-tree with effective page capacity C and height h. Then the average number of accesses of R-tree pages needed to answer a k-nearest neighbor query is given by*

$$P_{all}^{L_\infty}(k) = \sum_{j=0}^{h-1} \frac{an_{mbr}^{L_\infty}(\sigma_j, d_{nn}^{L_\infty}(k))}{C^{h-j}} =$$

$$\boxed{\sum_{j=0}^{h-1} \left\{ \frac{(N-1)}{2^{D_2} \cdot C^{h-j}} \left(\left(\frac{C^{h-j}}{N}\right)^{\frac{1}{D_0}} + \left(\frac{k}{N-1}\right)^{\frac{1}{D_2}} \right)^{D_2} + \frac{1}{C^{h-j}} \right\}} \tag{8}$$

Proof: According to Lemma 2.1, a page's MBR on level j has side length σ_j on average. According to Lemmas 3.1 and 3.2 the number of query-sensitive anchors forcing a page access is $an_{mbr}^{L_\infty}(\sigma_j, d_{nn}^{L_\infty}(k))$. Since in the biased model the probability of each point to be chosen as query anchor is identical, that is, $\frac{1}{N}$, a page on level j is accessed with probability $\frac{1}{N} \cdot an_{mbr}^{L_\infty}(\sigma_j, d_{nn}^{L_\infty}(k))$ on average. Eq. 8 sums up the access probabilities of every single page from the root to the leaf level (N/C^{h-j} pages on level j). □

Note that when $j = h-1$, Eq. 8 and 9 provide the average number of *leaf* accesses. We can simplify Eq. 8 under the reasonable assumptions that N is large ($N \gg 1$) and $D = D_0 = D_2$ (which holds for any mathematical fractal).
Corollary:

$$\boxed{P_{all}^{L_\infty}(k) \approx \sum_{j=0}^{h-1} \left(\frac{1}{C^{h-j}} + \left(1 + \left(\frac{k}{C^{h-j}}\right)^{\frac{1}{D}}\right)^{D} \right)} \tag{9}$$

Proof: By substituting the formulas for $an_{mbr}^{L_\infty}$, σ_j, and $d_{nn}^{L_\infty}(k)$ and performing some asymptotic manipulations as $N \to \infty$. □

With respect to the above formulas, we make the following observation. Contrary to previous results, the search effort does *not* suffer from the 'dimensionality curse' when $D \ll E$ as a result of the fact that the embedding dimensionality E does not appear in Eq. 8 and 9. Even when E is large, the k-nearest neighbor search performance is dictated by the fractal dimension of the data set.

4 Proposed L_2 Nearest Neighbor Formulas

In this section, we analyze nearest neighbor queries in the L_2 norm, that is, $d(x,y) = \left\{\sum_{i=1}^{E}(x_i - y_i)^2\right\}^{1/2}$. Fortunately, this slightly modified scenario can be handled in a manner very similar to that of the L_∞-norm. While the ideas behind the analysis remain the same, the formulas are more complicated. The first lemma determines the average Euclidean distance between neighbors.

Lemma 4.1 *The average Euclidean distance of a data point to its kth nearest neighbor is given by*

$$d_{nn}^{L_2}(k) = \frac{\left(\Gamma(1+\frac{E}{2})\right)^{\frac{1}{E}}}{\sqrt{\pi}} \cdot \left(\frac{k}{N-1}\right)^{\frac{1}{D_2}}. \qquad (10)$$

Proof: We proceed analogously to the proof of Lemma 3.1. All neighbors of a query anchor with Euclidean distance less than ϵ appear in the surrounding hypersphere of radius ϵ. Again we use Eq. 4 to estimate the number of data points located in this sphere, that is,

$$\overline{nb}(\epsilon, \text{'}E\text{-d sphere'}) = (N-1) \cdot Vol(\epsilon, \text{'}E\text{-d sphere'})^{\frac{D_2}{E}}, \qquad (11)$$

Using the volume formula for E-d hyperspheres [26],

$$Vol(\epsilon, \text{'}E\text{-d sphere'}) = \frac{\pi^{\frac{E}{2}} \cdot \epsilon^E}{\Gamma(1+\frac{E}{2})}, \qquad (12)$$

we solve Eq. 11 for ϵ. □

The next step concerns the estimation of the number of query-sensitive anchors of a certain page. In the Euclidean case, the nearest neighbor query again is transformed into a range query in the biased model, this time with spherical queries of radius $d_{nn}^{L_2}(k)$. In [28] it is shown that all anchors of spherical queries with radius ϵ are located in the page's MBR inflated by a rounded frame of width ϵ as depicted in Fig. 2. These geometric solids, however, become more difficult to describe with every additional (embedding) dimension. The following lemma provides a solution for the cases of $E = 2, 3$.

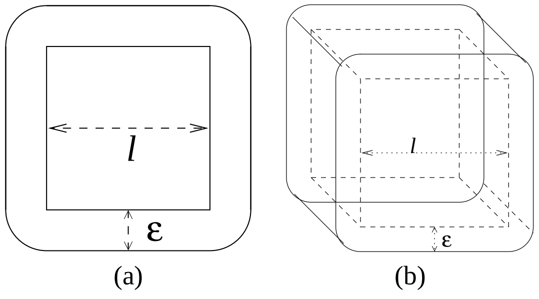


Figure 2. MBR of side length l dilated by rounded frame of width ϵ in (a) 2-d and (b) 3-d.

Lemma 4.2 *Assume the biased model, spherical queries of radius ϵ (= nearest neighbor queries with Euclidean search radius ϵ). Then the average number of query-sensitive anchors of an MBR with side length l is given by*

$$an_{mbr}^{L_2}(l,\epsilon) = \begin{cases} (N-1)\cdot\left(l^2 + 4l\epsilon + \pi\epsilon^2\right)^{\frac{D_2}{2}} + 1, \\ \quad \textit{if } E = 2 \\ (N-1)\cdot\left(l^3 + 6l^2\epsilon + 3\pi l\epsilon^2 + \frac{4}{3}\pi\epsilon^3\right)^{\frac{D_2}{3}} + 1, \\ \quad \textit{if } E = 3 \end{cases} \qquad (13)$$

Proof: The proof is identical to that of Lemma 3.2, except that the volume for the 2- and 3-dimensional solids, as depicted in Fig. 2, are used instead of the cube volume. Note that Theorem 2.2 is still applicable to these solids due to their regularity. □

Using Eq. 13 we can now derive an exact formula for $P_{all}^{L_2}(k)$, just as we did in Eq. 8:

$$P_{all}^{L_2}(k) = \sum_{j=0}^{h-1} \frac{an_{mbr}^{L_2}(\sigma_j, d_{nn}^{L_2}(k))}{C^{h-j}} \qquad (14)$$

For higher than 3 dimensions we come up with an upper and lower bound of the average number of query-sensitive anchors rather than the exact value. Since the L_∞-variation of the problem can be solved more easily, we reduce the Euclidean case appropriately.

Lemma 4.3 *For each MBR-size l, search distance ϵ, and dimension E holds*

$$an_{mbr}^{L_\infty}(l, \epsilon/\sqrt{E}) \le an_{mbr}^{L_2}(l,\epsilon) \le an_{mbr}^{L_\infty}(l,\epsilon). \qquad (15)$$

Proof: Assume an arbitrary MBR. Then for this MBR the region of possible query-sensitive anchors corresponding to $an_{mbr}^{L_2}(l,\epsilon)$ includes the region corresponding to $an_{mbr}^{L_\infty}(l,\epsilon/\sqrt{E})$, since

$$\|\vec{x}\|_\infty = \max_i |x_i| = \frac{\epsilon}{\sqrt{E}} \Rightarrow \|\vec{x}\|_2 \le \sqrt{E\cdot\left(\frac{\epsilon}{\sqrt{E}}\right)^2} = \epsilon. \qquad (16)$$

Therefore,

$$\|\vec{x}\|_\infty \le \|\vec{x}\|_2, \tag{17}$$

so that the same property holds for the regions corresponding to $an_{mbr}^{L_\infty}(l,\epsilon)$ and $an_{mbr}^{L_2}(l,\epsilon)$. These inclusion relationships of the regions, however, provide an ordering of the region volumes, which is preserved by the number of query anchors located in it (see Theorem 2.2 for the proportionality of volumes and number of anchors). □

Lemma 4.4 *Nearest neighbor searching according to the two distance functions is related by*

$$d_{nn}^{L_2}(k) = d_{nn}^{L_\infty}(k') \; for k' = \left(\frac{2}{\sqrt{\pi}} \cdot \left(\Gamma(1+\frac{E}{2})\right)^{\frac{1}{E}}\right)^{D_2} \cdot k. \tag{18}$$

Proof: Solve $d_{nn}^{L_\infty}(k') = d_{nn}^{L_2}(k)$ for k'. □

Theorem 4.5 *Let $k' = (\frac{2}{\sqrt{\pi}} \cdot (\Gamma(1+\frac{E}{2}))^{\frac{1}{E}})^{D_2} \cdot k$. Then the average number of accesses of R-tree pages needed to answer a k-nearest neighbor query with respect to the Euclidean distance, is estimated as*

$$\sum_{j=0}^{h-1} \frac{an_{mbr}^{L_\infty}(\sigma_j, \frac{d_{nn}^{L_\infty}(k')}{\sqrt{E}})}{C^{h-j}} \le P_{all}^{L_2}(k) \le \sum_{j=0}^{h-1} \frac{an_{mbr}^{L_\infty}(\sigma_j, d_{nn}^{L_\infty}(k'))}{C^{h-j}} \tag{19}$$

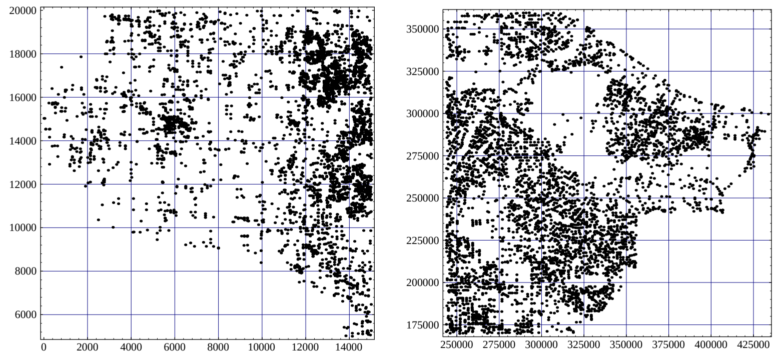
Proof: Directly by applying Theorem 2.2 and Lemmas 4.3 and 4.4. □

Analogously to the last section, we perform some asymptotic manipulations to simplify Eq. 19. Here k' depends on the embedding dimension E and so do both the upper and lower bound of $P_{all}^{L_2}(k)$. However, $k' \to (\frac{2}{\sqrt{\pi}})^{D_2}$ for $E \to \infty$, and thus both bounds monotonously decrease as E becomes large. In addition, by assuming $N \gg 1$ and $D_0 = D_2$ and using Eq. 9, we get the following estimation for $P_{all}^{L_2}(k)$, $E \to \infty$:

$$h + \frac{1}{C} \le P_{all}^{L_2}(k) \le \sum_{j=0}^{h-1} \left(\frac{1}{C^{h-j}} + \frac{2}{\sqrt{\pi}} \left(1 + \left(\frac{k}{C^{h-j}}\right)^{\frac{1}{D}} \right)^{D} \right) \tag{20}$$

5 Experiments

In this section, we present results from nearest neighbor experiments on both real and synthetic data sets. We designed experiments to determine the accuracy of our formulas for increasing nearest neighbors (k) and for increasing dimensionalities (D and E).



(a) `MGcty` data set (b) `LBcty` data set

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

Data sets: We used the following data sets:

- `MGcty` - road intersections in Montgomery County from the TIGER database of the U.S. Bureau of Census (E=2, N= 27,282, D_0 = 1.719 and D_2 = 1.518);
- `LBcty` - road intersections in Long Beach County from TIGER census data (E=2, N= 36,548, D_0 = 1.728, and D_2 = 1.732);
- `Sierpinski` - a synthetic, non-uniform fractal depicted in Fig 1, known as Sierpinski's triangle [30]. It was generated by embedding a Sierpinski triangle in $E = 10$ dimensional space, with $D_0 = D_2$ = 1.585. The Sierpinski triangle was on a two dimensional plane, which was deliberately *not* aligned with any of the axes;
- `plane` - uniformly generated points for a range of sizes N (from 1-500K) lying on a 2-d plane (*i.e.*, $D_0 = D_2 = 2$), embedded in E-d space (E ranges from 2-100). Two orthonormal vectors were randomly generated (via Gram-Schmidt) and points were uniformly sampled from the plane spanned by these vectors;
- `manifold` - uniformly generated points lying on a randomly oriented, $D_0 = D_2$)-dimensional linear manifold in E-space. Thus, for a fixed E, the intrinsic dimensionality $D_0 = D_2$ can be adjusted.

Queries: For each data set, we ran 100 queries. The workloads were composed of query anchor points randomly sampled from each data set in accordance with the biased query model. For the first set of experiments, we ran each workload to produce a variety of result sizes (*i.e.*, nearest neighbors) k. For the second set, we ran each workload data sets of fixed intrinsic dimensionality with embedding dimensions E ranging from 2-100, and of fixed embedding dimension with D_0 and D_2 ranging from 1-6. Below we tabulate and plot I/O results from the experiments.

Software: For optimized performance, we used an implementation of `libGiST`[1] [19] with R*-tree heuristics (see [2]) and the nearest neighbor algorithm from [20]. We presorted the points into an insertion order ensuring tight, square-like MBRs.

5.1 Accuracy

The plot in Fig. 4 demonstrates the accuracy of our formulas for both the L_∞ and L_2 norms on the `MGcty` data set, plotting I/O as a function of k nearest neighbors retrieved.[2] Fig. 4(a) plots the predicted (based on the formula in Eq. 8) and observed number of *leaf* page accesses for the L_∞ norm; Fig. 4(b) plots *total* page accesses. Fig. 4(c) plots the predicted (Eq. 14) and observed number of leaf accesses for L_2 norm; Fig. 4(d) plots total accesses. The plots graphically depict that the predicted values fell within one standard deviation of the observed values. As Fig. 4 shows, performance in the two norms is approximately the same.

To see how our formulas compared with others, we looked at the results reported in [29]. Table 2(a)-(b) presents a comparison with $k = 1$, the only value of k that the previous formulas can estimate, for `plane` with $E = 2$. Table 2(c) presents a comparison with $k = 1$ for a random sample of `MGcty` with $N = 9,552$. Note that, whereas the previous results are reported as a range, our formulas give a single number.

N	observed	our estimate	lower	upper
1K	1.63	1.44	1.34	4.66
10K	1.70	1.44	1.34	4.66
20K	1.80	1.44	1.34	4.66
50K	2.04	1.44	1.34	4.66
100K	1.88	1.44	1.34	4.66
200K	2.28	1.44	1.34	4.66

C	observed	our estimate	lower	upper
10	2.68	2.13	1.84	5.55
20	2.19	1.77	1.56	5.07
50	2.03	1.43	1.34	4.66
100	1.90	1.29	1.23	4.46

C	observed	our estimate	lower	upper
10	3.06	3.12	2.70	7.01
20	2.36	2.61	2.33	6.24
50	2.27	2.16	1.98	5.44
100	1.89	1.92	1.77	4.94

Table 2. Comparison to the results in [29]. The first column gives the average I/O observed in experiments, the second gives the estimate according to our formula, and the third and fourth give lower- and upper-bound estimates from the formulas in [29].

5.2 Effect of Dimensionality

In this set of experiments, our goal was to understand the effects of embedding dimension and intrinsic dimensionality. Figure 5(a) shows leaf accesses as a function of embedding dimension E for the `plane` data set with N=100K and $k = 50$, in both graphical and tabular form. We increased E over a wide range, from 2 to 100 dimensions, while keeping D_0 and D_2 constant. Table 3 displays the results from a similar set of experiments performed on the `Sierpinski` data set. To ensure fair comparisons, we increased the page size by the same factor that E was increased, in order to achieve approximately constant branching factor (= effective bucket capacity). The average results are given, along with one standard deviation. To gain some perspective, we also present values that the uniformity and independence assumptions would predict for the same queries. Note that these values are overpessimistic.

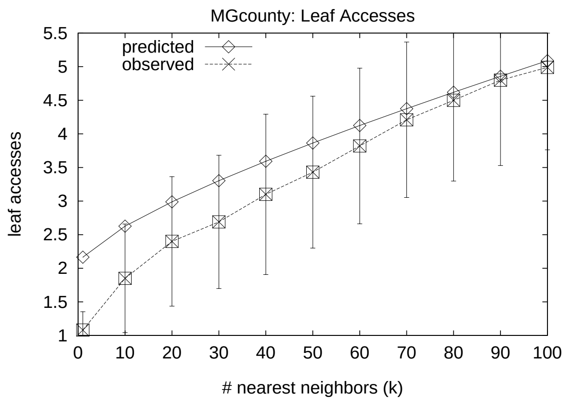
Figure 5(b) shows the leaf accesses as a function of the intrinsic dimensionality for the `manifold` data set with N=100K and k=50, in both graphical and tabular form. We increased $D_0 (= D_2)$ from 1 to 6 while keeping the embedding dimension fixed at $E = 10$. The I/O performance appears to depend on D (not E), worsening as D increases. Note that the uniformity and independence assumptions are insensitive to the increase in D.

6 Conclusions

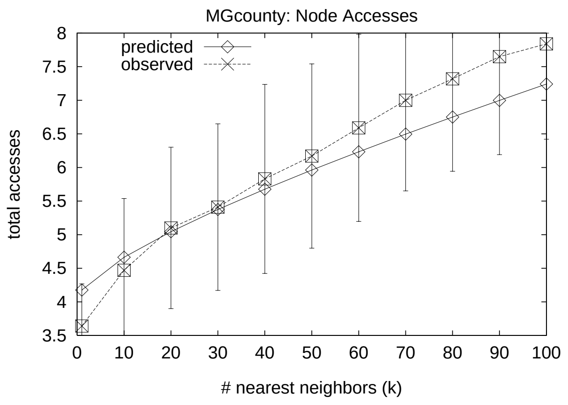
We focused on the performance of R-trees for k-nearest neighbor queries. Contrary to previous approaches that make the uniformity or independence assumption (or both), we only assumed that real data sets are self-similar, and that they are characterized by their fractal (more specifically,

[1] available at `http://gist.cs.berkeley.edu`

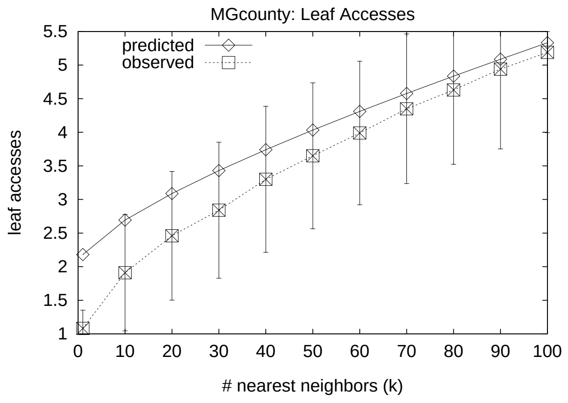
[2] The results from the `LBcty` data set were similar and are not reported here for brevity.



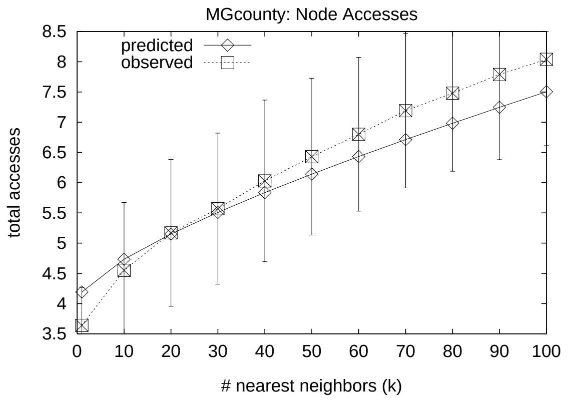
(a) L_∞ norm, leaf accesses



(b) L_∞ norm, total accesses



(c) L_2 norm, leaf accesses



(d) L_2 norm, total accesses

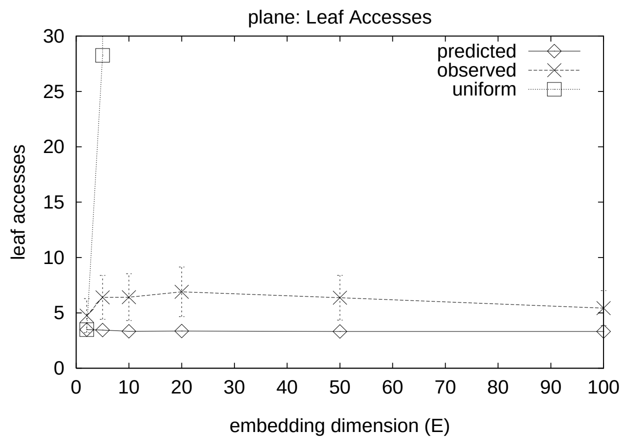
Figure 4. Observed versus predicted performance for `MGcty` data set, varying k, L_∞ norm: (a) leaf accesses and (b) total page accesses; L_2 norm: (c) leaf accesses and (d) total page accesses.

E	predicted	observed	unif/ind
2	2.53	4.72 ± 1.81	3.49
10	2.53	6.42 ± 2.11	847.26
20	2.53	7.76 ± 4.12	all
50	2.53	6.15 ± 2.82	all
100	2.53	5.64 ± 2.32	all

Table 3. Leaf accesses for increasing E with $D_0 (= D_2)$ fixed at 1.58, `Sierpinski` data set (N = 100K and $k = 50$).

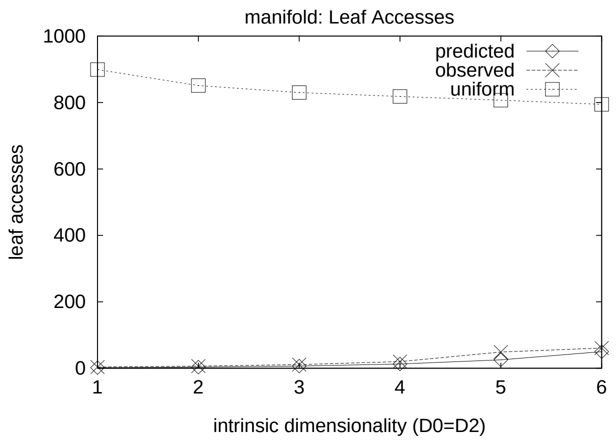
Hausdorff D_0 and Correlation D_2) dimensions. Using these two fractal dimensions, we derived closed formulas for the problem at hand. The practical contribution of this work is the derivation of accurate formulas to predict the number of I/Os in an R-tree for nearest neighbor queries, which could be used for spatial query optimization. These formulas have the following advantages over previous cost models:

- They accurately predict I/O performance, to within *one standard deviation* of that observed. At the same time, they are closed-form and easy to use, requiring only *five* parameters: N, k, C, D_0 and D_2;
- They apply to real data sets because they do not make any implicit assumptions of uniformity and independence;
- They are valid for k-nearest neighbors, for any arbitrary k, and as such are more general than the older formulas (*e.g.*, [29] and [7]) that only estimate for the case of $k = 1$;



E	predicted	observed	unif/ind
2	3.49	4.75 ± 1.54	3.49
5	3.45	6.40 ± 1.98	28.26
10	3.34	6.42 ± 2.11	847.26
20	3.36	6.90 ± 2.24	all
50	3.32	6.37 ± 2.02	all
100	3.32	5.43 ± 1.58	all

(a) `plane` data set, increasing E



D	predicted	observed	unif/ind
1	1.79	3.55 ± 1.70	899.31
2	3.36	6.56 ± 2.71	851.1
3	6.53	11.04 ± 3.08	829.89
4	12.85	20.21 ± 7.33	818.32
5	25.30	48.90 ± 18.69	806.89
6	49.77	60.94 ± 21.67	794.35

(b) `manifold` data set, increasing $D_0 (= D_2)$

Figure 5. Leaf accesses for increasing (a) E with $D_0 (= D_2)$ fixed at 2, `plane` data set; and (b) $D_0 (= D_2)$ with E fixed at 10, `manifold` data set. In both data sets, N = 100K and $k = 50$.

- They are exceptionally accurate in high dimensions when compared to other cost models, which are over-pessimistic; for example, whereas our formulas gave an estimate for 100-dimensional data that is correct to within one or two I/Os, other cost models will 'blow up' under these circumstances.

From a theoretical point of view, our formulas are interesting because they show that data sets with low fractal dimension do *not* suffer from the dimensionality curse when $D_0 \ll E$, in contrast to the recent pessimistic analyses. Indeed, nearest neighbor search does not necessarily exhibit the curse of dimensionality when the embedding dimension is high. Our experiments produced results not sensitive to increasing embedding dimension, for values of E ranging from 2 to as high as 100. Furthermore, as intuitively expected, we observed that the I/O performance degraded with fixed embedding dimension as the intrinsic dimensionality increased.

Future work could involve analyzing nearest neighbor queries on data sets of non-point objects.

Acknowledgements

We would like to thank Gı́sli Hjaltason for providing an R*-tree implementation with incremental nearest neighbor code.

References

[1] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. *Fourth Int. Conf. on Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Oct. 13-15 1993. also available through anonymous ftp, from olympos.cs.umd.edu: ftp/pub/TechReports/fodo.ps.

[2] 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 23-25 1990.

[3] A. Belussi and C. Faloutsos. Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In *Proc. 21st Intl. Conf. on VLDB*, pages 299–310, Zurich, 1995.

[4] Alberto Belussi and Christos Faloutsos. Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In *Proc. of VLDB*, pages 299–310, Zurich, Switzerland, September 1995.

[5] Stefan Berchtold, Christian Boehm, Bernhard Braunmueller, Daniel A. Keim, and Hans-Peter Kriegel. Fast similarity search in multimedia databases. In *SIGMOD Conference*, pages 1–12, 1997.

[6] Stefan Berchtold, Christian Boehm, and Hans-Peter Kriegel. The pyramid-tree: Breaking the curse of dimensionality. In *SIGMOD Conference*, pages 142–153, 1998.

[7] Stefan Berchtold, Christian Bohm, Daniel A. Keim, and Hans-Peter Kriegel. A cost model for nearest neighbor search in high-dimensional data space. In *PODS '97*, pages 78–86, Tucson, AZ, May 1997.

[8] Stefan Berchtold, Bernhard Ertl, Daniel A. Keim, Hans-Peter Kriegel, and Thomas Seidl. Fast nearest neighbor search in high-dimensional spaces. In *ICDE '98*, pages 209–218, Orlando, Florida, February 1998.

[9] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An index structure for high-dimensional data. *VLDB*, pages 28–39, 1996.

[10] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *ICDT '99*, pages 217–235, Jerusalem, Israel, January 1999.

[11] Allan Borodin, Rafail Ostrovsky, and Yuval Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. In *ACM STOC '99*, pages 312–321, Atlanta, GA, May 1999.

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

[13] Paolo Ciaccia, Marco Patella, and Pavel Zezula. A cost model for similarity queries in metric spaces. In *PODS '98*, pages 59–68, Seattle, WA, June 1998.

[14] C. Faloutsos and I. Kamel. Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension. In *Proc. ACM 13t Int. Symposium on Principles of Database Systems*, pages 4–13, Mineapolis, Minnesota, 1994.

[15] Christos Faloutsos and Ibrahim Kamel. Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension. In *Proc. ACM SIGACT-SIGMOD-SIGART PODS*, pages 4–13, Minneapolis, MN, May 24-26 1994. Also available as CS-TR-3198, UMIACS-TR-93-130.

[16] Volker Gaede and Oliver Gunther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, June 1998.

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

[18] Joseph Hellerstein, Elias Koutsoupias, and Christos Papadimitriou. On the analysis of indexing schemes. In *PODS '97*, pages 249–256, Tucson, Arizona, May 1997.

[19] Joseph M. Hellerstein, Jeffrey F. Naughton, and Avi Pfeffer. Generalized search trees for database systems. In *Proc. of VLDB Conf.*, pages 562–573, Zurich, Switzerland, Sept. 11-15 1995.

[20] G. R. Hjaltason and H. Samet. Ranking in spatial databases. In *Advances in Spatial Databases - 4th Symposium (SSD '95)*, pages 83–95, 1995.

[21] Ibrahim Kamel and Christos Faloutsos. Hilbert R-tree: An improved R-tree using fractals. In *Proc. of VLDB Conference,*, pages 500–509, Santiago, Chile, Sept. 12-15 1994.

[22] Norio Katayama and Shin'ichi Satoh. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *SIGMOD '97*, pages 369–380, Tucson, AZ, May 1997.

[23] Vikrant Kobla, David Doermann, King-Ip Lin, and Christos Faloutsos. Compressed domain video indexing techniques using dct and motion vector information in mpeg video. In *SPIE Proceedings Vol. 2916*, Boston, MA, November 1996.

[24] Flip Korn, Nikolaos Sidiropoulos, Christos Faloutsos, Eliot Siegel, and Zenon Protopapas. Fast nearest-neighbor search in medical image databases. *Conf. on Very Large Data Bases (VLDB)*, September 1996. Also available as Univ. of Maryland tech. report: CS-TR-3613, ISR-TR-96-13.

[25] King-Ip Lin, H.V. Jagadish, and Christos Faloutsos. The TV-tree - an index structure for high-dimensional data. *VLDB Journal*, 3:517–542, October 1994.

[26] F. Le Lionnais. *Le Nombres Remarquables*. Hermann, Paris, 1983.

[27] B. Pagel, H. Six, H. Toben, and P. Widmayer. Towards an analysis of range query performance. In *Proc. of ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 214–221, Washington, D.C., May 1993.

[28] B.-U. Pagel and H.-W. Six. Are window queries representative for arbitrary range queries? In *Proc. ACM 15th Int. Symp. on Principles of Database Systems*, pages 150–160, Montréal, Canada, 1996.

[29] Apostolos Papadopoulos and Yannis Manolopoulos. Performance of nearest neighbor queries in R-trees. *6th Int. Conf. on Database Theory (ICDT '97)*, Jan. 8-10 1997.

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

[31] 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.

[32] Dennis Shasha and Tsong-Li Wang. New techniques for best-match retrieval. *ACM TOIS*, 8(2):140–158, April 1990.

[33] Yannis Theodoridis and Timos Sellis. A model for the prediction of R-tree performance. In *Proc. of ACM PODS*, Montreal, Canada, 1996.

[34] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB '98*, pages 194–205, New York, NY, August 1998.