Geoinformatica (2006) 10: 261–293
DOI 10.1007/s10707-006-9828-7

Spatial Query Estimation without the Local Uniformity Assumption

Yufei Tao · Christos Faloutsos · Dimitris Papadias

© Springer Science + Business Media, LLC 2006

Abstract Existing estimation approaches for spatial databases often rely on the assumption that data distribution in a small region is uniform, which seldom holds in practice. Moreover, their applicability is limited to specific estimation tasks under certain distance metric. This paper develops the *Power-method*, a comprehensive technique applicable to a *wide range* of query optimization problems under both L_∞ and L_2 metrics. The Power-method eliminates the local uniformity assumption and is, therefore, accurate even for datasets where existing approaches fail. Furthermore, it performs estimation by evaluating only one simple formula with minimal computational overhead. Extensive experiments confirm that the Power-method outperforms previous techniques in terms of accuracy and applicability to various optimization scenarios.

Keywords databases · selectivity · estimation · range queries · nearest neighbor

1 Introduction

A *spatial database* manages a large number of multi-dimensional entities, ranging from simple locations (e.g., the coordinates of restaurants, hotels, etc.) to objects with complex extents (rivers, cities, and so on). It is imperative to geographic

Y. Tao (✉)
Department of Computer Science and Engineering, Chinese University of Hong Kong,
Sha Tin, New Territories, Hong Kong
e-mail: taoyf@cse.cuhk.edu.hk

C. Faloutsos
Department of Computer Science, Carnegie Mellon University,
Forbes Avenue, Pittsburgh, PA, USA

D. Papadias
Department of Computer Science, Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong

information systems (GIS) that require efficient retrieval of objects according to certain predicates on their coordinates. In the past decade, the database community has extensively studied numerous issues arising from efficient implementation of a spatial database, covering data representation/modeling [31], query languages [21], index structures [4], and very importantly, query optimization. In particular, the most important query types involve *range search*, k nearest neighbor (kNN) retrieval, and *spatial join*.

To explain these concepts, consider Figure 1 which illustrates a set of 2D (black) points representing the locations of restaurants. The shaded area corresponds to a square whose centroid lies at the downtown center q (the white dot) and its side length equals 4 miles. A range search retrieves the points in the query square, i.e., the result includes all the restaurants except p_3 and p_7. A kNN query, on the other hand, returns the k objects closest to a query point q. Assuming $k = 3$, the first NN is p_5 (it has the shortest distance to q among all the points in the dataset), while the second and third NNs are p_6 and p_1, respectively. The goal of a spatial join is to report all pairs of restaurants that are (1) within the shaded area, and (2) their mutual distance is no more than 1 mile. In our example, the join produces a single pair $\{p_1, p_2\}$. Note that $\{p_3, p_4\}$ is not a result because (although their distance is less than 1 mile) p_3 falls out of the query region.

Formally, given a query point q and a radius r, a range search (RS) retrieves all points o in a database DB satisfying $dist(o,q) \leq r$, where $dist(o,q)$ denotes the distance between o and q. A k nearest neighbor (kNN) query returns the k closest data points $o_1, o_2, ..., o_k$ such that, given any point $o \in DB - \{o_1,o_2,...,o_k\}$, $dist(o_i,q) \leq dist(o,q)$ for all $1 \leq i \leq k$. Finally, a *regional distance* (*self*-) *join* (RDJ) retrieves all pairs of objects (in some *constraint region*) that are close to each other. It specifies a query point q, a constraint radius r, a distance threshold t, and returns $(o_1,o_2) \in DB \times DB$ such that $o_1 \neq o_2$, $dist(o_1,q) \leq r$, $dist(o_2,q) \leq r$, and $dist(o_1,o_2) \leq t$. The special case where $r = \infty$, corresponds to a *global distance join* (GDJ).

The result of any query depends on the underlying distance metric. The most widely used metrics in spatial applications are the L_∞ and L_2 norms (called Manhattan and Euclidean distance, respectively). Specifically, given two points q, o

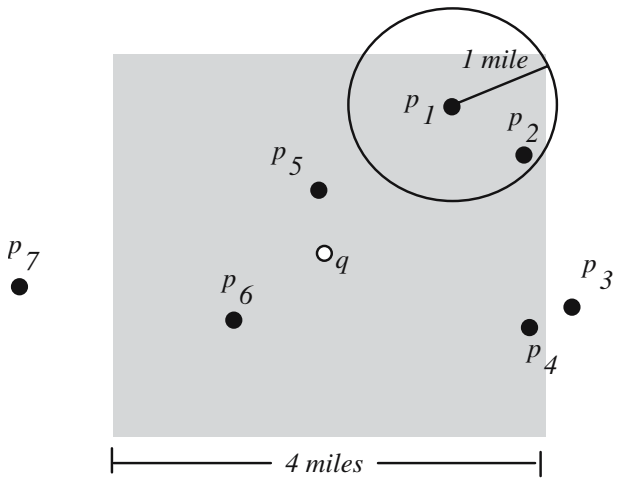


Figure 1 An example spatial database.

whose coordinates on the i-th dimension ($1 \le i \le m$, where m is the dimensionality) are q_{yi} and o_{yi} respectively, their L_∞ and L_2 distances are:

$$L_\infty(q,o) = \max_{i=1\sim m}\left(\left|q_{yi} - o_{yi}\right|\right)$$
$$L_2(q,o) = \left[\sum\nolimits_{i=1\sim m}\left(q_{yi} - o_{yi}\right)^2\right]^{1/2}$$

For instance, if L_∞ is assumed, a search region of an RS is a square centered at q with extent $2r$ on each dimension (e.g., $r = 2$ miles for the shaded area in Figure 1). If L_2 is used, the query region is a circle centered at q with radius r.

1.1 Motivation

The first objective of this paper is to study techniques that accurately predict the selectivity of range and join queries. For an RS, the selectivity is the ratio between the number of retrieved points and the dataset cardinality (e.g., the query in Figure 1 has a selectivity 5/7). For an RDJ, it equals the ratio between the number of qualifying pairs and the size of the cartesian product $DB \times DB$ (i.e., 1/49 for the RDJ in Figure 1, where the denominator is the square of the dataset cardinality 7). The concept of selectivity does not apply to kNN queries where exactly k points are returned. As a second step, we present solutions for estimating the cost (in terms of the number of nodes accessed) of answering each type of queries (including kNN) using a multi-dimensional index.

Selectivity and cost estimation is imperative for the query optimizer to decide an effective execution plan for a complex query that involves numerous (spatial and non-spatial) operators. For example, assume a restaurant relation with three attributes (*name*, *coordinates*, *revenue*) and the query "find all the restaurants that (1) are within 2 miles from the downtown center, and (2) their annual revenues are higher than 1 million dollars". There are at least two different plans for evaluating the query. Specifically, the first one is to find the restaurants qualifying predicate (1) from which we prune away those violating (2). An alternative plan would evaluate predicate (2) first and then filter according to (1). Clearly, the better approach depends on whether condition (1) or (2) is more selective, as well as the overhead of extracting the objects satisfying each predicate.

The above query involves range search, while similar phenomena also exist for RDJ, as in "find all pairs of restaurants that are within 2 miles from the city center, 1 mile from each other, and their revenues differ by no more than 100 thousand dollars. Finally, the cost analysis of kNN queries is also the basis of sophisticated operations including *k closest-pair* (kCP) search [14], where the goal is to report the k pairs of restaurants closest to each other (e.g., in Figure 1, the result for $k = 1$ is pair $\{p_3, p_4\}$). One possible kCP solution is to find the k NNs for each restaurant, and record its distance to every NN, that is, totally $N{\cdot}k$ nearest distances are recorded where N is the number of restaurants. These distances are then sorted in ascending order, and the restaurant pairs producing the k smallest distances are returned as the result. Clearly, the cost of the above algorithm equals the overhead of N kNN queries plus that of sorting $N{\cdot}k$ numbers.

In addition to query optimization, selectivity and cost prediction is also useful for providing quick (expected) statistics about the query result and performance, *prior to*

physically processing the query. Given the expected result size and running time, a user may decide to abort the query if the estimate is considerably larger than her/his expectation. Furthermore, in some scenarios, a user may be interested in only the result size (i.e., a "count" query) as opposed to the details of qualifying objects. In this case, the estimated selectivity can be directly returned as an approximate answer.

Despite the importance of selectivity/cost analysis, however, the existing approaches are inadequate. Specifically, the previous methods fall into two categories. *Local methods* produce a "tailored" estimate depending on the query's concrete location, typically using a histogram [1], [18], which partitions the data space into (disjoint or overlapping) *buckets*. As explained in the next section, histograms have several limitations. First, they assume that the data distribution in each bucket is uniform (i.e., *local uniformity assumption*), which is rarely true for real datasets. Second, estimation for queries under L_2 metric may require expensive evaluation time. Third, their applicability to *k*NN queries is questionable. *Global methods* [5], [16] provide a single estimate corresponding to the average selectivity/cost of all queries, independently of their locations. Such techniques avoid the problems of histograms, but have a serious drawback: since queries at various locations may have different characteristics, their respective selectivity/cost can differ considerably from the average value.

In summary, currently there does not exist a comprehensive approach able to perform all estimation tasks (i.e., selectivity/cost prediction in any distance metric) effectively. Such an approach is highly desirable in practical systems, where it is important to have a single efficient method that applies to all cases, instead of multiple methods that are effective for certain sub-problems but inefficient/inapplicable to others.

1.2 Contributions

This paper develops the Power-Method, a novel estimation technique which combines the advantages of both local and global methods, but avoids their deficiencies. The proposed solutions possess the following attractive features:

- It *eliminates the local uniformity assumption* and, therefore, is accurate for several datasets where existing techniques fail.
- It is the first *comprehensive* technique applicable to *selectivity and cost estimation for all query types* under both L_∞ and L_2 norms (the two most important metrics in spatial databases).
- It supports *all distance metrics* with *small space requirements*.
- It performs any query estimation by evaluating only *one simple formula*; hence, its computational overhead is minimal.
- It provides important information on data characteristics that may assist data mining.

Extensive experimentation proves that the Power-Method achieves accurate estimation (with average relative error below 20%) in circumstances where traditional methods completely fail. The rest of the paper is organized as follows. Section 2 reviews existing query estimation techniques. Section 3 introduces the concept of local power laws and proves the related theorems. Section 4 presents two implementations of the Power-Method, and Section 5 experimentally evaluates their performance. Section 6 concludes the paper with directions for future work.

2 Related work

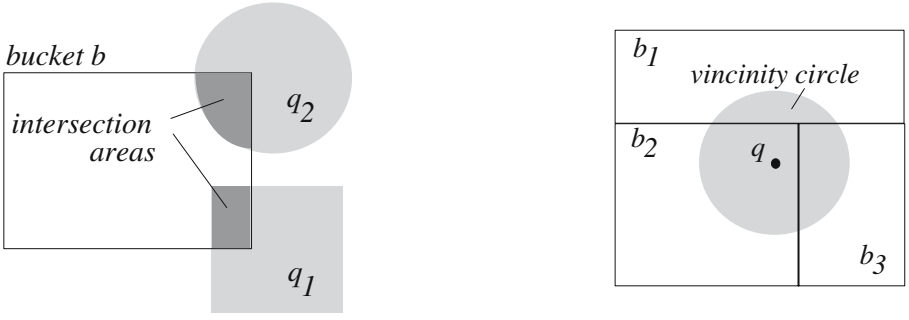
Section 2.1 introduces the existing local estimation approaches and elaborates their deficiencies, while Section 2.2 focuses on the global methods.

2.1 Local estimation methods

Most approaches for multi-dimensional selectivity estimation are based on histograms [1], [18], [12], [19], [24], [29], [34]. A histogram constructs a small number of rectangular buckets, and stores for each bucket b the number of points N_b in its extent (in case of overlapping buckets, a point in the intersection region of multiple buckets is assigned to a unique bucket). The *density* D_b of bucket b is defined as $D_b = N_b/a_b$, where a_b is the area of b. To estimate the selectivity of an RS (range search) query, q, a histogram first computes for each bucket, b, the intersection area a_{intr} (with q); assuming that the data in each bucket are uniform, the number of qualifying objects inside this bucket is then estimated as $D_b \cdot a_{intr}$. The final selectivity is obtained by summing the estimate from each intersecting bucket.

Theodoridis et al. [35], suggest that, under the local uniformity assumption, RDJ (regional distance join) selectivity estimation can be reduced to GDJ (global distance join) on uniform data, for which several cost models are available [2], [3], [35]. The idea is to apply these uniform models inside the query constraint region Q, based on the density of the bucket that covers Q. If Q intersects multiple buckets, their average density is used. The cost estimation of RS [16], [27], [35] and RDJ [35] follows the same reasoning.

Application of histograms has been limited mainly to RS queries under the L_∞ metric, where the intersection area a_{intr} (between the bucket and the query) is a (hyper) rectangle (e.g., query q_1 in Figure 2a). For queries in the other metrics, however, the area computation is usually expensive. In Figure 2a, for instance, the L_2 RS query q_2 intersects bucket b into a rather irregular shape whose area a_{intr} is difficult to compute. To tackle this problem, Berchtold, et al. [6] suggest the *monte-carlo* method, which generates a set of α uniform points in the bucket, counts the number β of them in the query region, and estimates the a_{intr} as $a_b \cdot \beta/\alpha$, where a_b is the bucket's area. A large number of random points (which increases with the dimensionality) is necessary to obtain accurate estimation. Repeating this process for every (partially) intersecting bucket may lead to significant overhead.



(a) Irregular intersection area (b) Difficulty for *k*NN

Figure 2 Deficiencies of histograms.

We are not aware of any local estimation method for *k*NN query cost. The main difficulty of applying histograms lies in the prediction of d_k (the distance from the query point to its *k*-th NN), which is the first step of the cost analysis. The value of d_k should be such that the vicinity circle centering at q with radius d_k covers expectedly k points (assuming uniformity in buckets). Finding such a circle requires non-trivial repetitive tuning (increasing-decreasing the radius, depending on how many points fall in the current "trial" circle). This is especially complicated if the circle intersects multiple buckets (the circle in Figure 2b intersects 3 buckets) and produces irregular intersection regions. To avoid this problem, Berchtold et al. [6] and Bohm [10] apply their (uniform) *k*NN cost model, to non-uniform distributions *anyway*, and show that (surprisingly) sufficient accuracy *may* be achieved. In Section 3.3 we reveal the conditions that must be satisfied for this conjecture to hold in practice.

Other multi-dimensional estimation techniques include *sampling* [13], [25], [28], [37], *kernel estimation* [9], *single value decomposition* [29], *compressed histograms* [20], [22], [23], [33], *maximal independence* [15], *Euler formula* [32], etc. These methods, however, target specific cases (mostly RS selectivity under the L_∞ norm), and their extensions to other problems are unclear.

2.2 Global estimation methods

All the existing global estimation methods (for non-uniform data) require the *intrinsic dimension* of the underlying dataset. In order to illustrate the concept of intrinsic dimension, consider the well-known Sierpinski triangle example. Figure 3a illustrates the first three steps in constructing the Sierpinski triangle, which is derived from an equilateral triangle ABC, by excluding its middle (triangle $A'B'C'$) and recursively repeating (endlessly) this procedure for each of the resulting triangles. Subsequent patterns contain "holes" in any scale; moreover, each smaller triangle is a miniature replica of the whole pattern. Figure 3b demonstrates a Sierpinski dataset with 100 k points. Clearly, the dataset is not 1D since its distribution is not "linear", as in Figure 3c (where points fall on the diagonal line of the data space). On the other hand, it is not 2D either (even though it is "embedded" in a 2D space), as is true for the uniform dataset in Figure 3d. Actually the intrinsic dimension of Sierpinski is 1.58 (i.e., between 1D and 2D).

A common method to measure the intrinsic dimension, is to divide the data space into $(1/\varepsilon)^2$ regular cells with length ε (assuming a unit space). In Figure 3b, $\varepsilon = 0.25$ and the data space is partitioned into 16 cells. For each cell i ($1 \le i \le (1/\varepsilon)^2$), we count the number p_i of points in it, and denote the square sum of p_i for all cells as $S_2(\varepsilon) = \sum_i (p_i)^2$. Then, the intrinsic dimension is defined as:

Definition 2.1: For a self-similar dataset, the *intrinsic dimension* d is:

$$d = \partial S_2(\varepsilon)/\partial \varepsilon$$

for the range of ε where $\partial S_2(\varepsilon)/\partial \varepsilon$ is a constant.

Definition 2.1 extends to higher dimensions in a straightforward manner. The intrinsic dimension offers a convenient way to predict the average selectivity/cost of *biased queries* (i.e., the query distribution follows the data distribution), based on the following power law.

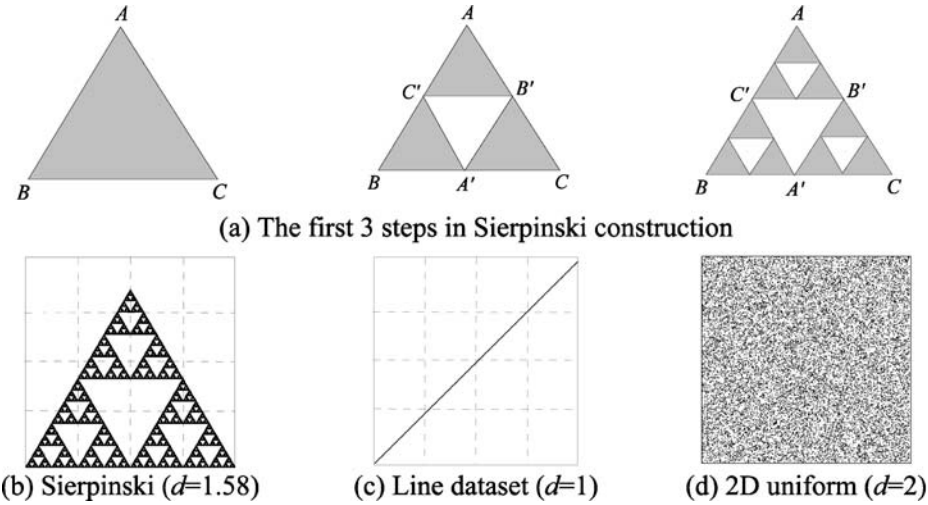


Figure 3 Datasets' intrinsic dimensionalities.

Theorem 2.1 [5] **(Global power law):** Given a dataset with intrinsic dimension d, the *average selectivity* of L_∞ RS biased queries with radius r equals $N \cdot (2r)^d$, where N is the cardinality of the dataset.

Belussi and Faloutsos [5] present formulae that reduce the average selectivity of RS with L_2-norm to the L_∞ case, and thus enable the application of the above power law. Belussi and Faloutsos [5] and Faloutsos et al. [17] analyze selectivity estimation of GDJ using power laws, but their analysis does not address RDJ. Regarding average query cost, Faloutsos and Kamel [16] study RS queries, while kNN retrieval is discussed in Pagel et al. 2000 [26] and Bohm 2000 [10]. Although global power laws can accurately capture the structure of "perfect" fractal datasets, as shown in the experiments they fail when the dataset properties vary throughout the data space. The local power law concept presented next avoids these pitfalls.

3 The local power law

Section 3.1 explains the problems associated with the local uniformity assumption in histograms. Then, Section 3.2 describes the *local power law* (LPLaw), which overcomes these problems, and, unlike the global power law, does not assume the same properties throughout the data space. Applying the LPLaw, Section 3.3 solves the selectivity estimation problem, while Section 3.4 and 3.5 analyze query cost for L_∞ and L_2 metrics, respectively. Our discussion focuses on biased queries, due to their practical importance [5], [10], [26], [37]. Unbiased queries are discussed in Section 4.3. Table 1 summarizes the symbols that will be used frequently.

3.1 Density trap

Histograms are based on the hypothesis that the data distribution in sufficiently small regions of space is piece-wise uniform. Thus, they compute and store the

Table 1 Frequently used symbols.

Symbol	Description
$nb_p(r)$	Number of points within distance r from p
c_p	Local constant of point p
n_p	Local exponent of point p
N	Dataset cardinality
ρ	Radius of the LPLaw neighborhood
r	Query radius (for RS, RDJ)
k	Number of NNs to be retrieved (for kNN)
t	Distance threshold (for RDJ)
f	Average node fanout of the R-tree
l	Side length of a leaf MBR of the R-tree
Sel(q)	Query selectivity
Cost(q)	Query cost

density of such regions. A main contribution of this work is to show that this, apparently reasonable, hypothesis is *wrong* for the vast majority of real data, as well as the vast majority of perfect, Euclidean-geometry datasets. Consider a set of N = 1,000 2D points evenly distributed along the major diagonal of a unit data space, and a query point q at the center of the space (Figure 4). What is the density in the vicinity of q, e.g., a square region centered at q? The answer is surprising: *undefined*!

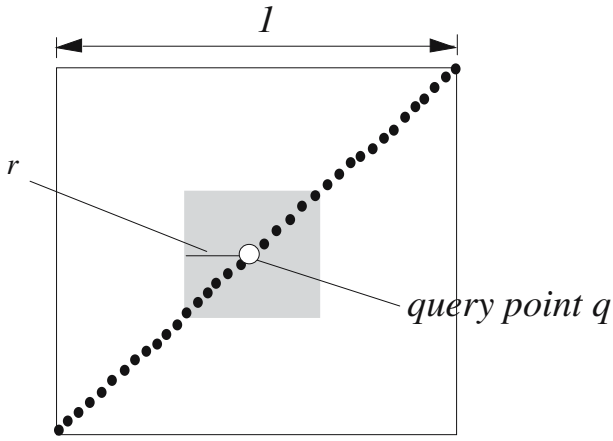
The density changes *dramatically* with the radius r of the vicinity, diverging to infinity when r goes to zero. This is so counter-intuitive that we give an arithmetic example. If the radius is r = 0.5 (i.e., the shaded square covers the entire data space), then the density equals 1,000 (the number of points in the square divided by its area). When the radius shrinks to r = 0.05 (i.e., the square has a side length 0.1), then 100 points are expected to fall in the square, and the density is $100/0.1^2$ = 10,000! For a general value of r, the side length of the square is $2r$, and hence, the shaded area is expected to cover $2r$·1,000 points, leading to a density of $2{,}000r/(2r)^2 = 500/r$. Evidently, the density goes to zero with growing radius r.

Notice that the above situation is generated not by a mathematical oddity like the Hilbert curve or the Sierpinski triangle, but by a line, a perfectly behaving Euclidean object! Moreover, many real objects behave like lines or collections of lines (highway networks, rivers). Thus, the problem is real, and cannot be downplayed as a mathematical oddity—it *does* appear in practice! The next question then is: if the local density around a point q is not well defined, what is the invariant that could help us describe somehow the neighborhood of q? The following section provides the answer.

3.2 Assumptions, definitions, and properties

The resolution to the density trap comes from the concept of intrinsic dimensionality. The data points around the vicinity of point q in Figure 4 form a linear manifold—asking for their density is as unusual as asking for the area of a line. The

Figure 4 Density trap.



invariant we hinted before is encapsulated in the concept of *Local Power Law* (LPLaw):

Assumption 3.1 (Local power law): Let p be a data point in a dataset. For certain range of r, the number $nb_p(r)$ of points with distances no more than r from p can be represented as:

$$nb_p(r) = c_p \cdot r^{n_p} \qquad (3-1)$$

where c_p and n_p are constants termed the *local constant* and *exponent*, respectively.

For convenience we refer to c_p and n_p collectively as the *coefficients* of the LPLaw for point p. Each LPLaw models the distribution around a *particular* data point. The LPLaw can handle the "density trap" problem of Figure 4 with $c_p = 2N$ and $n_p = 1$. Furthermore, it captures the global power law as a special case where the coefficients of all data points are similar, as for example, the "perfect" datasets of Figure 3.

In general, however, the LPLaw of various points may differ in two aspects. *First, two points can have different local exponents, implying that the data correlation "characteristics" in their respective vicinity are different.* This is obvious in Figure 5a that mixes four distributions with different intrinsic dimensions (the upper-left

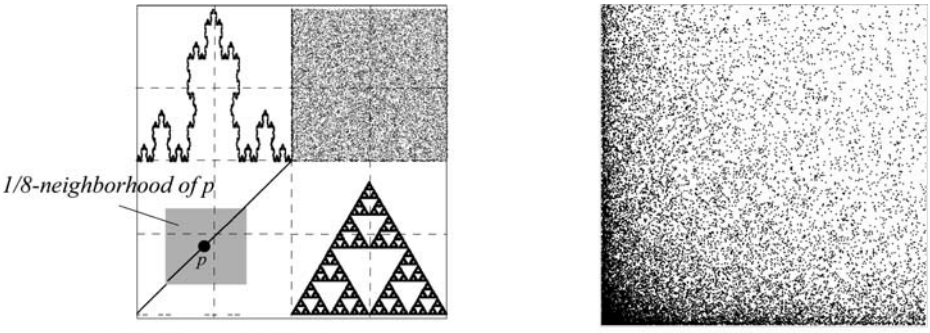


Figure 5 Non-fractal examples.

"Koch snowflake" distribution in Figure 5a has intrinsic dimension 1.34). The local exponent of each point is determined by the intrinsic dimension of the region it lies in. *Second, two points may have similar local exponents but different constants, implying that the data "densities" in their respective vicinity are different.* Figure 5b illustrates the 2D independent Zipf distribution, where all points have local exponents 2 (revealing the independence between the two dimensions), while those in denser areas have higher constants. Notice that *the LPLaw includes the local uniformity as a special case where* $n_p = 2$.

Given a point p in an m-dimensional dataset, we define the L_∞ *ρ-neighborhood* of p as the m-dimensional box centering at q with length 2ρ on each axis. Then, the LPLaw coefficients of p can be measured as follows.

Lemma 3.1 (LPLaw under L_∞): Given a data point p such that the data distribution in its L_∞ ρ-neighborhood is self-similar with intrinsic dimension d_p, then the LPLaw of p under the L_∞ metric is:

$$nb_{p\infty}(r) = \left(N_{p\infty}/\rho^{d_p}\right) r^{d_p} \qquad (3-2)$$

where $N_{p\infty}$ denotes the number of points in the L_∞ ρ-neighborhood of p.

The above lemma is a "local" version of Theorem 2.1 inside a ρ-neighborhood (instead of the entire data space). Note that ρ can be any value in the range of r where the LPLaw holds. As an example, the LPLaw of p in Figure 5a can be measured using a 1/8-neighborhood (the shaded rectangle in the figure), while its LPLaw $nb_p(r) = c_p \cdot r$ (with exponent 1) holds for the range [0, 1/4] of r.

Similarly, the LPLaw of L_2-norm can be measured in the L_2 *ρ-neighborhood* of a point p (i.e., a sphere centering at p with radius ρ) while the following lemma provides a faster way to derive an L_2 LPLaw from its L_∞ counterpart.

Lemma 3.2 (LPLaw under L_2): Given an m-dimensional data point p with L_∞ LPLaw $nb_{p\infty}(r) = c_{p\infty} \cdot r^{n_p}$, its L_2 LPLaw is:

$$nb_{p2}(r) = c_{p\infty} \left\{ \frac{VolSphere_2(1)}{VolSphere_\infty(1)} \right\}^{n_p/m} r^{n_p} \qquad (3-3)$$

where $nb_{p2}(r)$ is the number of points within L_2 distance r from p, and $VolSphere_2(1)$ is the volume of an m-dimensional sphere with radius 1.

It is clear that the LPLaw of a point p under different distance metrics have the same local exponent n_p, and their local constants can be derived from each other using n_p. The intuitive explanation is that the ρ-neighborhood of a point under various metrics only affects how many points fall in the neighborhood (related to the neighborhood volume), but does not influence the way data are correlated (which is a data characteristic captured by the exponent).

The LPLaw is satisfied in many real datasets. To demonstrate this, we select two real distributions (Figure 6): (1) *SC* dataset, which contains 36 k points representing the coast line of Scandinavia, and (2) the *UK* dataset, which contains 40 k points representing the boundary of the United Kingdom. Figure 7a plots $nb_{p\infty}(r)$ (i.e., the number of points within distance r to a point p in the L_∞ metric) as a function of r (in log-log scale) for the two points p_1, p_2 in Figure 6a. Similarly, Figure 7b

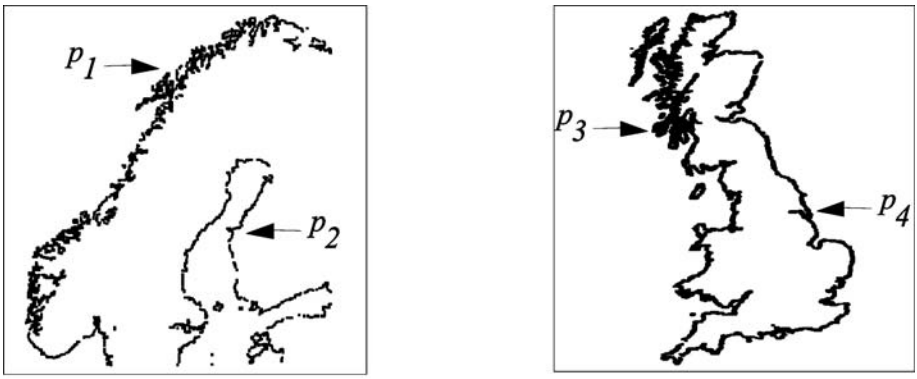


Figure 6 Real distributions.

illustrates the same information for the two points p_3, p_4 in Figure 6b. It is clear that $nb_{p\infty}(r)$ approximates a power law in all cases, and has various exponents (i.e., slopes of the fitting lines in Figure 7) for different points.

Figure 8a and b demonstrates the local exponent (constant) distribution for *SC*. The values are measured using L_∞ 0.05-neighborhoods (i.e., a square with length 0.1 on each axis). Figure 8c and d illustrate the corresponding distributions for *UK*. Constants and exponents differ substantially in the data space (suggesting that a global law for the whole dataset would introduce inaccuracy), but are highly location-dependent (confirming the intuition behind LPLaw).

3.3 Selectivity estimation using the LPLaw

Next we apply the LPLaw to estimate the selectivity of RS and RDJ queries.

Theorem 3.1 (RS selectivity): Given an RS query point q with radius r, the selectivity of q is:

$$\mathrm{Sel}(q) = c_q \cdot r^{n_q}/N \qquad (3-4)$$

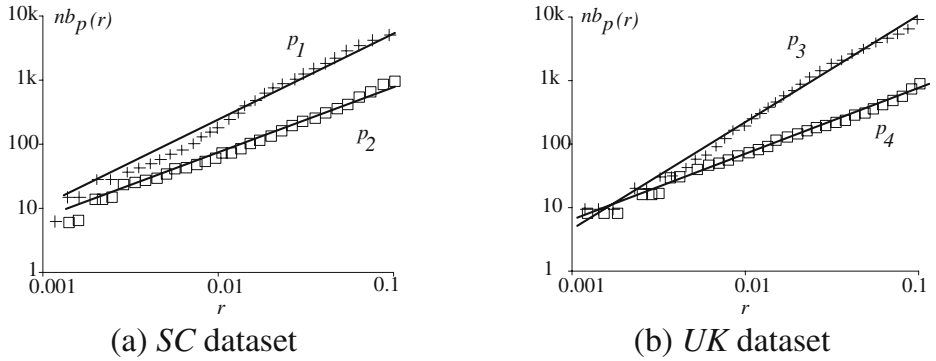
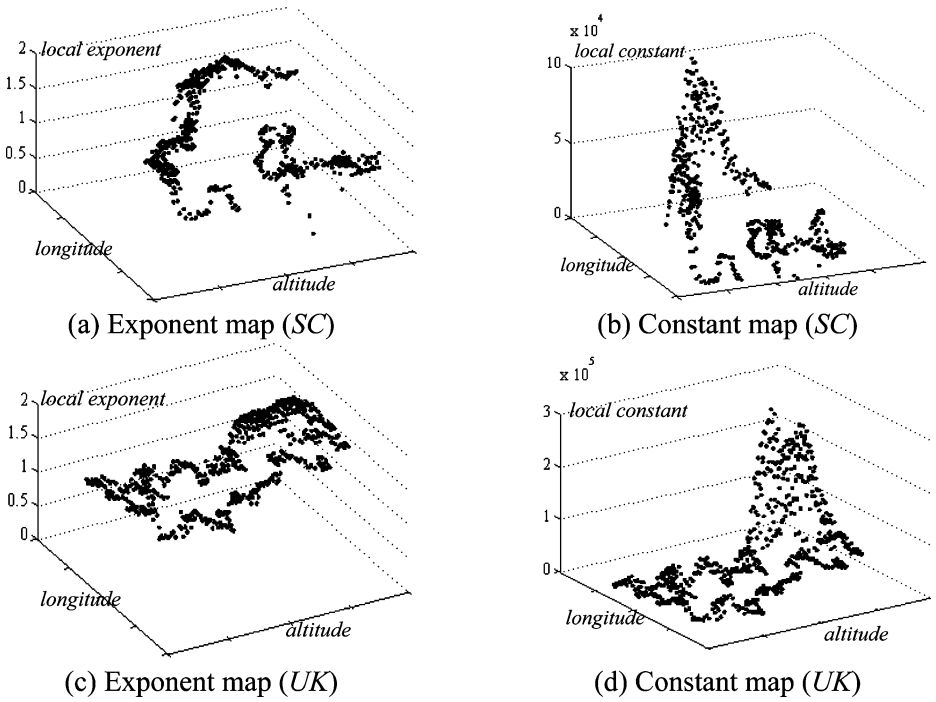


Figure 7 LPLaw in real data.



(a) Exponent map (*SC*) (b) Constant map (*SC*)

(c) Exponent map (*UK*) (d) Constant map (*UK*)

Figure 8 LPLaw coefficient distributions.

where N is the cardinality of the dataset, c_q the local constant (under the corresponding distance metric) at q, and n_q the local exponent.

Proof: Straightforward since the number $nb_q(r)$ of points retrieved by q satisfies the LPLaw $nb_q(r) = c_q \cdot r^{n_q}$. Hence the selectivity equals $c_q \cdot r^{nq}/N$. □

The following analysis is based on an assumption that the LPLaw is approximately the same for all the data points with distance no more than r from the query q. As demonstrated in the previous section, nearby points have similar LPLaws, and therefore, the assumption is reasonable for relatively small r.

Theorem 3.2 (RDJ selectivity): Let q be an RDJ query with (1) constraint radius r, and (2) distance threshold t. The selectivity of q is:

$$\mathrm{Sel}(q) = c_q^2 (t \cdot r)^{n_q} / (2N^2) \qquad (3-5)$$

where N is the cardinality of the dataset, c_q the local constant (under the corresponding distance metric) inside the constraint region, and n_q the local exponent.

Proof: Let N_r be the number of points in the constraint region (i.e., within distance r to q). According to the LPLaw (Definition 3.1):

$$N_r = c_q \cdot r^{n_q} \qquad (3-6)$$

Let us consider any point p among these N_r points. If we denote λ as the number of other data points within distance t from p (i.e., these data produce qualifying pairs with p for the join), λ satisfies:

$$\lambda = c_q \cdot t^{n_q} \tag{3-7}$$

In other words, p contributes λ pairs to the join result, and hence, the total contribution of all N_r points involves $N_r \cdot \lambda$ pairs. However, each pair is counted twice (as the contributions of the two participating points, respectively). Therefore, the actual number of qualifying pairs equals $N_r \cdot \lambda/2$, which leads to the equation in the theorem. □

3.4 Cost prediction using the LPLaw

After solving selectivity estimation, we proceed with the query cost analysis. Although the following formulae can be applied to any data partitioning index such as the *X*- [7], *A*-trees [30], etc, we assume the *R**-tree [4] because it is the most popular spatial structure and is the only one that has been implemented in commercial systems. We measure the query cost in terms of the number of *R*-tree leaf accesses because (1) this number dominates the total cost, and (2) in practice non-leaf nodes are usually found in the buffer directly. Unlike the selectivity analysis, the derivation for the query cost results in different formulae for various distance metrics. In the sequel, we first prove the theorems for L_∞ before extending the results to L_2 in Section 3.5. Our discussion is based on the following lemma:

Lemma 3.3 (*R*-tree node extent): Let l be the length of a leaf MBR on each dimension; then: $l = 2(f/c_\infty)^{1/n}$, where c_∞ (n) is the L_∞ local constant (exponent) at the centroid of the MBR, and f is the average node fanout (i.e., number of entries in a node).

Proof: Since a leaf node contains f entries, by the LPLaw we have $f = c_\infty \cdot (l/2)^n$. Solving l from this equation completes the proof. □

Theorem 3.3 (RS cost): Given an RS query point q with radius r, the cost of q is:

$$\text{Cost}(q) = \frac{c_{q\infty}}{f}\left[\left(\frac{f}{c_{q\infty}}\right)^{1/n_q} + r\right]^{n_q} \tag{3-8}$$

where $c_{q\infty}$ is the local constant (under metric L_∞) at q, n_q the local exponent, and f the average node fanout.

Proof: The crucial observation is that *the centriods of leaf MBRs are distributed in the same way as the underlying dataset*. The centroid distribution is essentially a sparse version of the data distribution. Equivalently, if the original dataset in a region follows an LPLaw $N_r = c_{q\infty} \cdot r^{n_q}$, the centroid distribution can be described by

$$N'_r = \left(c_{q\infty}/f\right) \cdot r^{n_q} \tag{3-9}$$

The law has the same local exponent as the dataset LPLaw, but its local constant is f times smaller. Denote l as the average side length of the leaf MBRs around q. Observe that such an MBR intersects the query region (a square centering at q with radius r) if and only if its centroid is within L_∞ distance $l/2 + r$ from q (e.g., in Figure 9, the centroid of the leaf MBR falls in the dashed square centering at q with a radius $l/2 + r$). Therefore, the number of intersecting leaf MBRs can be obtained by replacing r with $l/2 + r$ in Equation 3-9. The value of l is given in Lemma 3.3, and then Equation 3-8 results. □

Theorem 3.4 (*k*NN cost): The cost of a kNN query q is:

$$\text{Cost}(q) = \frac{1}{f}\left[f^{1/n_q} + k^{1/n_q}\right]^{n_q} \tag{3-10}$$

where n_q is the local exponent at q, and f the average node fanout.

Proof: Let d_k be the distance between the query point and its k-th NN. Then, by Definition 3.1, we have $k = c_{q\infty} \cdot (d_k)^n$, where $c_{q\infty}$ is the local constant at location q; hence $d_k = \left(k/c_{q\infty}\right)^{1/n}$. As proven in Berchtold et al. 1997 [6], the cost of a kNN query equals that of an RS query with radius d_k. As a result, the kNN cost can be represented as in Theorem 3.3, setting $r = d_k$. The formula in the theorem results from necessary simplification (which removes $c_{q\infty}$). □

An important observation from the above theorem is that, the cost of a kNN query q is not affected by the local constant, but depends only on the local exponent. Thus, the conjecture of [6], [10], that the kNN cost is the same for *all* datasets (i.e., independently of the data density in the vicinity of q), *only holds for datasets with the same local exponent* (i.e., 2 if uniform models are applied).

Theorem 3.5 (RDJ cost): Let q be an RDJ with (1) constraint radius r, and (2) distance threshold t. The query cost is:

$$\text{Cost}(q) = \frac{c_{q\infty}^{\,2}}{f^2}\left[\left(\frac{f}{c_{q\infty}}\right)^{1/n_q} + r\right]^{n_q}\left[2(f/c_\infty)^{1/n_q} + t\right]^{n_q} + \frac{c_{q\infty}}{f}\left[\left(\frac{f}{c_{q\infty}}\right)^{1/n_q} + r\right]^{n_q} \tag{3-11}$$

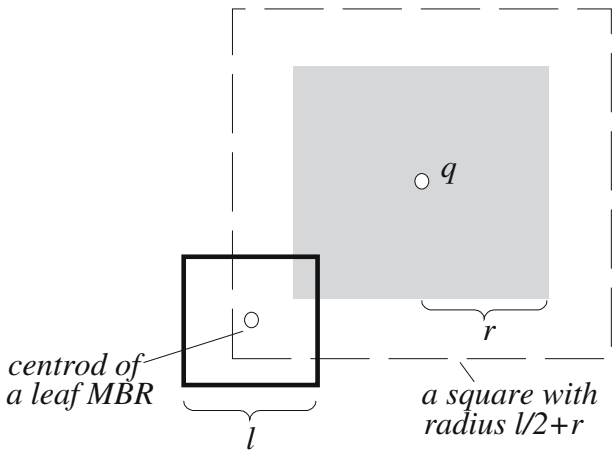


Figure 9 A query region, an intersecting leaf MBR, and the expanded region.

where $c_{q\infty}$ and n_q are the local constant (under metric L_∞) and exponent at q, respectively.

Proof: Let C be the number of leaf nodes intersecting the constraint region. By Theorem 3.3 we have: $C = (c_{q\infty}/f)\left[(f/c_{q\infty})^{1/n_q} + r\right]^{n_q}$. We use S to represent the set of these nodes. The R-join algorithm [11] processes the join by accessing (1) each node in S once, and (2) every pair of nodes in S whose centroids are within distance $l + t$, where l is the extent of a leaf MBR given in Lemma 3.3. The cost of (1) is exactly C (i.e., the size of S).

It suffices to analyze the number of pairs of type (2). As discussed in the proof of Theorem 3.3, the centroids of the nodes in S follow the LPLaw of Equation 3-9. Hence, each node in S produces pairs of (2) with $(c_{q\infty}/f)\cdot(l+t)^{n_q}$ elements in S. The total number of pairs is therefore $C\cdot(c_{q\infty}/f)\cdot(l+t)^{n_q}/2$ (as with the proof for Theorem 3.2, the "/2" is needed to avoid redundant counting), and R-join performs $C\cdot(c_{q\infty}/f)\cdot(l+t)^{n_q}$ accesses of type 2 (two accesses for each pair). Substituting C and l into the formula leads to Theorem 3.5. □

3.5 Extension to L_2 norm

In this section, we extend the analysis of the previous section to L_2 metric. Theorems 3.1 and 3.2 (on query selectivity) are directly applicable to L_2 norm, and hence, in the sequel we focus on query cost.

The major obstacle is the fact that a leaf MBR differs in shape from an L_2 circle. As shown in Figure 9, in deriving RS cost, we utilize the observation that a leaf MBR intersects the search region if and only if its centroid falls in the dashed rectangle (i.e., the expanded region). For L_2 norm, unfortunately, the expanded region does not have a regular form. This is explained in Figure 10, where the shaded area represents an RS, and the solid curve illustrates half of the boundary of the expanded region. Imagine that we move a square with side length $l/2$ (i.e., half of a leaf MBR's extent) along the perimeter of the query circle, keeping the square tangent with the circle (Figure 10 shows four positions of the square as dashed rectangles). Then, the expanded region corresponds to the area swept by a square.

Fortunately, since our focus is on spatial applications where the dimensionality m is low (usually 2 or 3), we can accurately approximate the expanded region with an

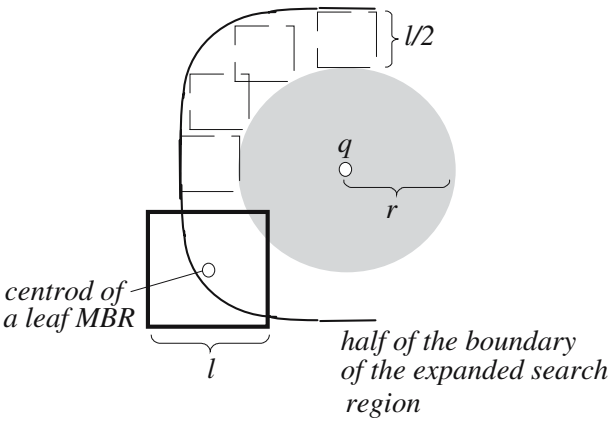


Figure 10 An SR query, an intersecting leaf MBR, and the expanded region.

m-dimensional sphere centering at q having radius $l/2 + r$. The approximation significantly simplifies the cost model derivation, and results in accurate prediction as shown in the experiments. Based on this assumption, the cost of RS and RDJ under L_2 norm can be obtained using directly Theorems 3.3 and 3.5, except that the local constant c_∞ should be replaced with c_2. The kNN cost can also be captured by Theorem 3.4, i.e., kNN retrieval under L_∞ and L_2 norms has exactly the same overhead. We omit the proofs since they are identical to those in Section 3.4.

4 Implementation of the power method

We have shown that all the estimation tasks can be performed by evaluating a single equation based on the LPLaw at the query location. Motivated by the fact that, points close in space usually have similar local constants and exponents (see Figure 8), we can pre-compute the LPLaw for a set of representative points, and perform the estimation using the LPLaw of a point close to q. Section 4.1 and 4.2 introduce two implementations of the Power-Method, the P-Patch and P-Anchor, based on this idea. Then, Section 4.3 discusses unbiased queries, where the query's location is not in the data-populated region.

4.1 The P-Patch method

P-Patch first partitions the (unit) m-dimensional data space into a grid with w^m regular cells (i.e., each cell has length $1/w$ on each axis), where w is a constant (e.g., $w = 64$ for $m = 2$). A cell is *active*, if it covers some data points, or *inactive* otherwise. For each active cell, we compute the LPLaw coefficients in the L_∞ ρ-neighborhood of the centroid as illustrated in Lemma 3.1. This can be done using a standard algorithm [16] for computing the intrinsic dimensionality, which has the same performance complexity as external sort. Inactive cells are ignored.

Instead of storing the LPLaw coefficients for all the active cells, P-Patch groups adjacent cells with similar coefficients into a *patch*, and stores the average coefficients in the patch. Given a biased (RS/kNN/RDJ) query q, we find the patch that contains q, and perform the estimation using the LPLaw of the patch. The total number B of patches is a parameter subject to the available memory size. We consider only *disjoint rectangular patches*, namely, (1) the extents of the cells covered by a patch must form a rectangle, and (2) no two patches cover the same cell. However, we do not require a partition of the data space, i.e., the union of all patches does not need to cover the entire space. In case that q falls in an inactive cell, the LPLaw of the nearest active cell is used. The problem of patch construction is formulated as follows.

Problem 4.1: Given S cells such that each cell i ($1 \le i \le S$) stores the (1) local constant c_i (under the L_∞ metric), (2) exponent n_i, and (3) number N_i of points it covers (if $N_i = 0$, n_i and c_i are undefined), the goal is to create B disjoint rectangular patches that minimize the average estimation error.

Before presenting our solution, we first analyze the "quality" of a patch b. Assume, without loss of generality, that b covers the first β active cells. Specifically, b is associated with (1) the average local constant $\overline{c} = (1/\beta)\sum_{i=1\sim\beta}(c_i)$, (2) the average exponent $\overline{n} = (1/\beta)\sum_{i=1\sim\beta}(n_i)$. Consider a L_∞ RS query q with radius r

that falls in cell i ($1 \le i \le \beta$). Using the LPLaw coefficients in cell i, its estimated selectivity is $\mathrm{Sel}_i(r) = c_i \cdot r^{n_i}$. On the other hand, if the coefficients of the patch are used, the estimation will be $\overline{\mathrm{Sel}}_i(r) = \overline{c} \cdot r^{\overline{n}}$. To reduce the error caused by averaging, we aim at minimizing the relative error $\mathrm{err}_i(r) = \left|\overline{\mathrm{Sel}}_i(r) - \mathrm{Sel}_i(r)\right| / \mathrm{Sel}_i(r)$. Towards this, we assume an *optimization range* $[\gamma_-, \gamma_+]$ of r that is "interesting" to the query optimizer. If r falls out of the interesting range, the query optimizer immediately decides to use an index to process the query *(i.e.,* $r < \gamma_-$), or not to (i.e., $r > \gamma_+$). Thus, the average error $\overline{err}_i$ incurred by all queries (with different radii) located in cell i is:

$$\overline{err}_i = \frac{1}{\gamma_+ - \gamma_-} \int_{\gamma_-}^{\gamma_+} err_i(r)\, dr = \frac{1}{\gamma_+ - \gamma_-} \int_{\gamma_-}^{\gamma_+} \left| \frac{\overline{c}}{c_i} r^{\overline{n} - n_i} - 1 \right| dr \qquad (4-1)$$

Then, the total error *pchterr* of patch b is the sum of the average error of all active cells (in b), weighted by the number of data points they contain, or formally:

$$pchterr = \sum_{\substack{\text{all active cells} \\ i \text{ in the bucket}}} (N_i \cdot \overline{err}_i) \qquad (4-2)$$

where N_i is the number of points in cell i, and $\overline{err}_i$ is given in Equation 4-1. The quality of P-Patch is measured as the sum (denoted as *totalerr*) of *pchterr* of all patches. To minimize *totalerr*, we adopt a greedy algorithm shown in Figure 11. Initially the histogram contains a single patch, whose extent is the MBR of all active cells. Then, at each iteration, we split a patch at the boundary of a cell (covered by the patch) along a certain dimension (i.e., *binary partitioning*). The splitting patch, dimension and position are the ones leading to the largest decrease in *totalerr*. The algorithm terminates after B patches have been obtained.

In the above discussion the patch quality is measured using the estimation error of RS selectivity. Similarly, we could choose to minimize the error for other query types. Our experiments, however, indicate that the P-Patch in all cases has similar performance. Hence, we adopt RS selectivity as the optimization goal due to its simplest equations.

algorithm Build_P-Patch (B)
/*this algorithm constructs a P-Patch with B patches */
1. divide the data space into regular cells; for each active cell, compute the number of points it covers, and its LPLaw coefficients
2. b_1.MBR=the MBR that covers all active cells (i.e., the initial patch); $b_1.\Delta E$=maximum *totalerr* decrease among all possible split positions
3. *Pcht*={b_1} /*the set of patches already obtained*/
4. while *Pcht* contains less than B patches
5. let b_s be the patch with the highest $b_s.\Delta E$
6. if $b_s.\Delta E$=0 then
return *Pcht* /* no more patch is necessary*/
7. split b_s into b' and b'' at the best split position
8. compute $b'.\Delta E$ and $b''.\Delta E$
9. *Pcht*=*Pcht* $-$ b_s; *Pcht*=*Pcht* $\cup$ {b',b''}
10. return *Pcht*
end Build_P-Patch

Figure 11 Algorithm for building P-Patch.

4.2 The P-Anchor method

P-Anchor selects a set A of *anchor points* from the database using any sampling technique [25], [28], [37], and, for each anchor point, it computes the LPLaw coefficients in its L_∞ ρ-neighborhood (using Lemma 3.1). Given a biased query q, P-Anchor first finds the point p in A nearest to q, and then performs the estimation using the LPLaw of p.

Each anchor point $p \in A$ implicitly defines a patch, namely, the region where points are closer to p than any other point in A. This is the *Voronoi cell* of p in the Voronoi Diagram imposed by the anchor points. Patches defined by Voronoi cells offer some flexibility compared to their rectangular counterparts (as in P-Patch). First, a Voronoi cell can be a polygon with variable (up to $|A|-1$, the number of points in A) edges. This property is especially interesting in higher dimensional spaces, where the Voronoi cell becomes a complex polyhedral, while all rectangular patches have exactly $2m$ faces (m is the dimensionality). Second, Voronoi cells are immediately defined after the anchor points are obtained, without going through any splitting process.

It is worth mentioning that, P-Anchor differs considerably from a sampling method in the following ways. First, P-Anchor can perform all the estimation tasks, while sampling has been limited to selectivity estimation. Second, even for selectivity prediction, P-Anchor evaluates the LPLaw of a single anchor point, while sampling examines each point against the query conditions. Third, P-Anchor is efficient independently of the data distribution, while it is well-known problem that sampling is inaccurate for skewed data [13]. Fourth, P-Anchor achieves satisfactory accuracy using a very small number (100 in our experiments) of anchors, while sampling requires a much larger fraction of the dataset [13].

4.3 Estimation for unbiased queries

P-Patch and P-Anchor are optimized for biased queries. Query optimization for unbiased queries is less important because, an RS/RDJ query in a non-populated region usually returns only a small number of objects, and the query optimizer can safely assume a very low selectivity in selecting the execution plan. Detecting whether a query q is biased is easy: for P-Patch (P-Anchor), we check if the query region intersects (covers) any patch (anchor point). Only if the answer is positive, the optimizer performs query estimation, using the LPLaw associated with the patch (anchor point) closest to q.

On the other hand, a kNN query that is far away from data is most likely to be *meaningless*. According to [8],[1] this happens if the distance from q to its k-th NN differs from that between q and its $(k + k')$-th NN by less than ε percent, where k' and ε are user-defined constants. In this case, the k NNs are not necessarily "better" than the next k' NNs. Such queries should be avoided, or the user should at least be warned about the significance of the result. Consider Figure 12 where the dataset contains the black points, and the white dot represents a single NN query q ($k = 1$). Note that q is unbiased because it is distant even from its nearest data point p, which

[1] It is worth mentioning that Beyer et al. 1999 [8] focus on high-dimensional data, where NN search is meaningless even for $k = 1$. Here, using their definition of "meaningless", we show that in 2D space a kNN query may have the same problem if the query point falls in a non-populated area.

is crossed by curve C (i.e., part of a circle centering at q whose radius equals $dist(q,p)$). Curve C' is an arc on another circle also centering at q but with radius $dist(q,p')$, where p' is the 5th NN of q. Notice that the radius of C is very close to that of C'. If $\varepsilon = 10$ and $k' = 5$, the NN result is meaningless because $dist(q,p')$ exceeds $dist(q,p)$ by less than 10%—the importance of p is limited because a user who finds p useful would most likely be equally interested in p'. Detecting meaningless queries can be achieved by computing the distance from the query to its nearest patch (anchor point). If the distance is larger than certain threshold, then we judge the query to be meaningless.

5 Experiments

This section compares experimentally the accuracy and computational overhead of the Power-Method with *minskew* (the best histogram for low dimensionalities proposed in [1]), and the *global* power law that provides an average estimate using the global intrinsic dimension. We use a PIII 1 Ghz CPU, and six real datasets: *SC*, *UK*, *CA*, *LB*, *NA*, *Household*[2]. As mentioned in Section 3.2, *SC* (*UK*) contains 36 k (40 k) points along the boundary of Scandinavia (*UK*). *CA*, *LB*, *NA* include 62 k, 53 k, and 569 k points representing locations in California, the Long Beach County, and north America, respectively (Figure 13 illustrates the distribution of the three datasets). *Household* is three dimensional, and each point captures the expenditure of an American family on insurance, property tax, and electricity, represented as percentages of its annual income[3].

Unless otherwise stated, for 2D datasets, we partition the space into a grid of 64×64 cells on which the patches of P-Patch are constructed. The corresponding grid for the 3D dataset *Household* has $16 \times 16 \times 16$ cells. 0.05-neighborhoods under L_∞ metric (see Lemma 3.1) are used for constructing all P-Patch and P-Anchor. Every method (except *global* which needs only two values: the dataset cardinality and intrinsic dimension) is allowed the same amount of memory that is enough to store 100 patches or anchors. Figure 14a and b demonstrate the patch extents of P-Patch for *SC* and *UK*, respectively, and Figure 14c and d show the anchor points. The patches and anchors for the other datasets follow the data distribution in the same manner.

The bucket construction algorithm of *minskew* first partitions the data space into 64×64 regular cells, and obtains, for each cell, the number (*frequency*) of data points covered. Then, the cells are grouped into 140 buckets using a greedy algorithm that aims at minimizing a certain function. The number of buckets (140) is larger than the number of patches and anchor points, since each bucket stores a single value (frequency) instead of two (local coefficients). The bucket structure is rather sensitive to the dataset and the "optimization function" deployed; in many cases the algorithm terminates without producing enough buckets (the problem is also mentioned in [37]). Similar tuning problems exist for most multi-dimensional histograms such as *genhist* Gunopulos et al. [18]. To alleviate the problem, we tested

[2] *SC* and *UK* are available at http://www.cs.cityu.edu.hk/ds.html. *CA*, *LB*, and *NA* can be downloaded at http://www.census.gov/geo/www/tiger/, and Household at http://www.ipums.org/.

[3] We use Household to examine the effectiveness of our method for three dimensions because we are not aware of any published real 3D geographic data.

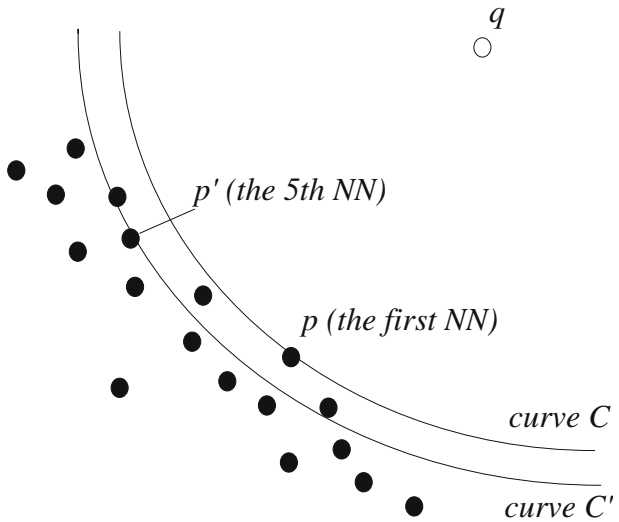


Figure 12 A meaningless NN query (k = 1).

multiple optimization functions. The best overall performance was obtained by minimizing $\sum_{i=1\sim B}(v_i/n_i)$, (where n_i is the average frequency of cells in bucket b_i and v_i the variance of n_i) and we follow this approach in our implementation.

A query workload consists of 500 biased queries with the same parameters. The workload estimation error is defined as: $(\sum_i |act_i - est_i|)/(\sum_i act_i)$, where act_i (est_i) is the actual (estimated) selectivity/cost of the i-th query in the workload. For *minskew* and metric L_2, if the query region (of RS/RDJ) partially intersects a bucket, the intersection area is computed using the *monte-carlo* method with 1,000 random points, which (based on our experiments using different numbers) leads to a reasonable tradeoff between estimation time and accuracy.

5.1 Selectivity estimation

The first experiment examines the performance of alternative methods for estimating RS selectivity. Figure 15a shows the error for L_∞ norm as a function of

Figure 13 Visualizations of *CA*, *LB*, and *NA*.

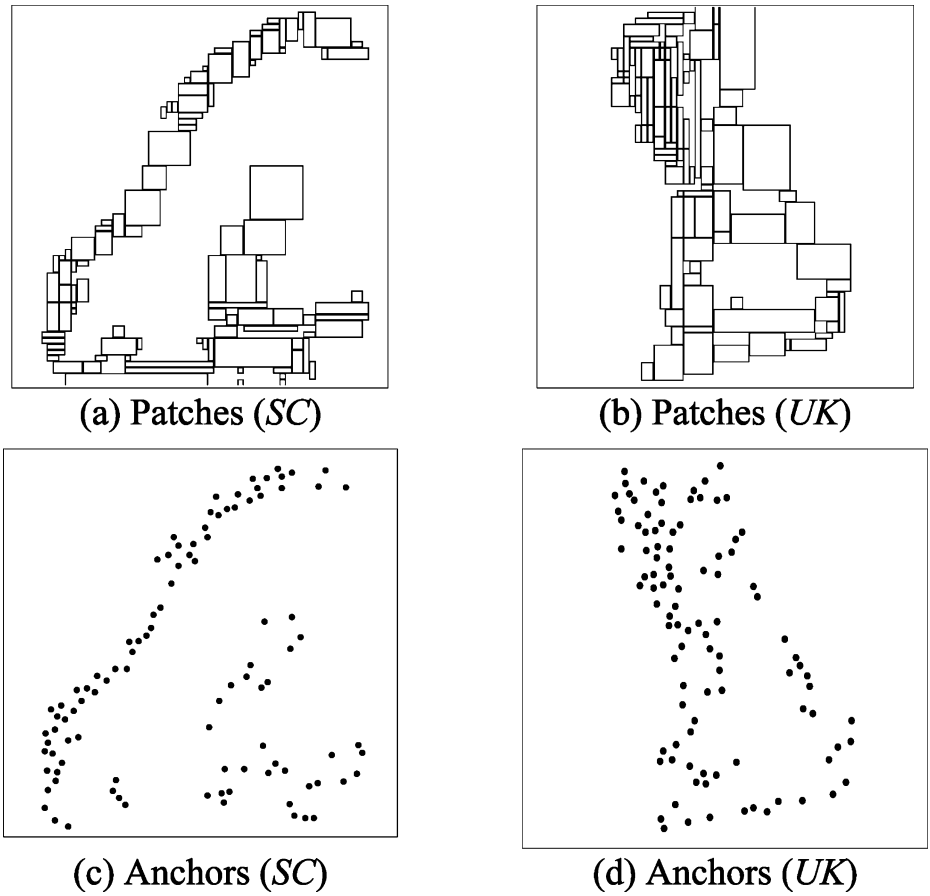


Figure 14 Anchor points (100 anchors).

the query radius r (ranging from 0.01 to 0.1) using the *SC* dataset. The Power-Method significantly outperforms the previous techniques. Both P-Patch and P-Anchor have the minimum error at $r = 0.05$, because the LPLaw is obtained using 0.05-neighborhoods. As expected, *global* yields considerable error due to the selectivity variation of individual queries. *Minskew* is even less accurate for small queries due to the violation of the local uniformity assumption. Its precision gradually improves for larger queries (which is consistent with previous studies [1]), because a large query covers more buckets completely, from which precise partial estimation can be obtained.

Figure 15b, c, d, e, and f demonstrate the error for the other datasets, and Figure 16 presents the results for L_2 metric, revealing similar observations. Every method has the best precision for *NA* (which has the largest size among all datasets) because, in general, statistical analysis is more accurate for larger datasets. Note that performance of each method has little difference under L_∞ and L_2 norms, which is consistent with the findings in Belussi and Faloutsos 1995.

To evaluate the error of predicting RDJ selectivity, we fix the query constraint radius r to 0.05, and vary the distance threshold t from 0.001 to 0.01. Figure 17 shows

the results for *SC*, *CA*, *NA* under L_∞ norm, and the other datasets under L_2 norm. The other dataset/norm combinations are omitted due to their similarity (similar to the comparison of Figures 15 and 16). There is no prediction by *global* because, as mentioned in Section 2.2, it is not applicable to RDJ [17] (it focuses only on global distance joins). P-Patch again outperforms P-Anchor and both are almost *an order of magnitude* more accurate than *minskew*. Since selectivity estimation is meaningless for *k*NN queries, in the sequel we proceed with cost (i.e., number of disk accesses) estimation.

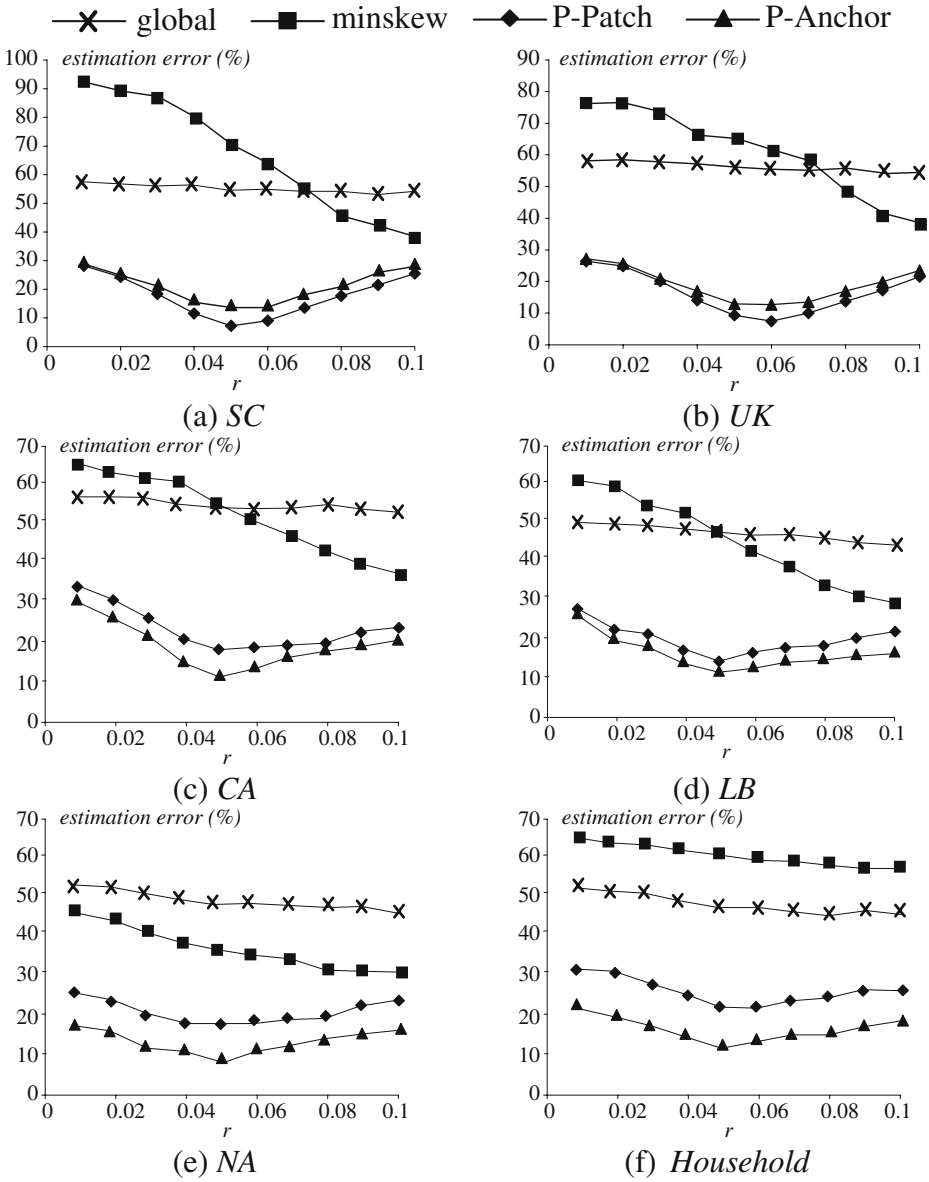


Figure 15 Error vs. query radius r (RS selectivity, L_∞).

5.2 Query cost estimation

Concentrating on RS, Figure 18 plots the error rate as a function of the query radius. The relative behavior of alternative methods (and the corresponding explanation) is similar to that in Figures 15 and 16 (we omit some dataset/metric combinations for the same reason in Figure 17).

Next we study the accuracy of *k*NN cost prediction. As mentioned in Section 2.1, there is no previous work that uses histograms to solve this problem. Thus, we replace *minskew* with the cost model proposed in Berchtold et al. 1997 [6], which

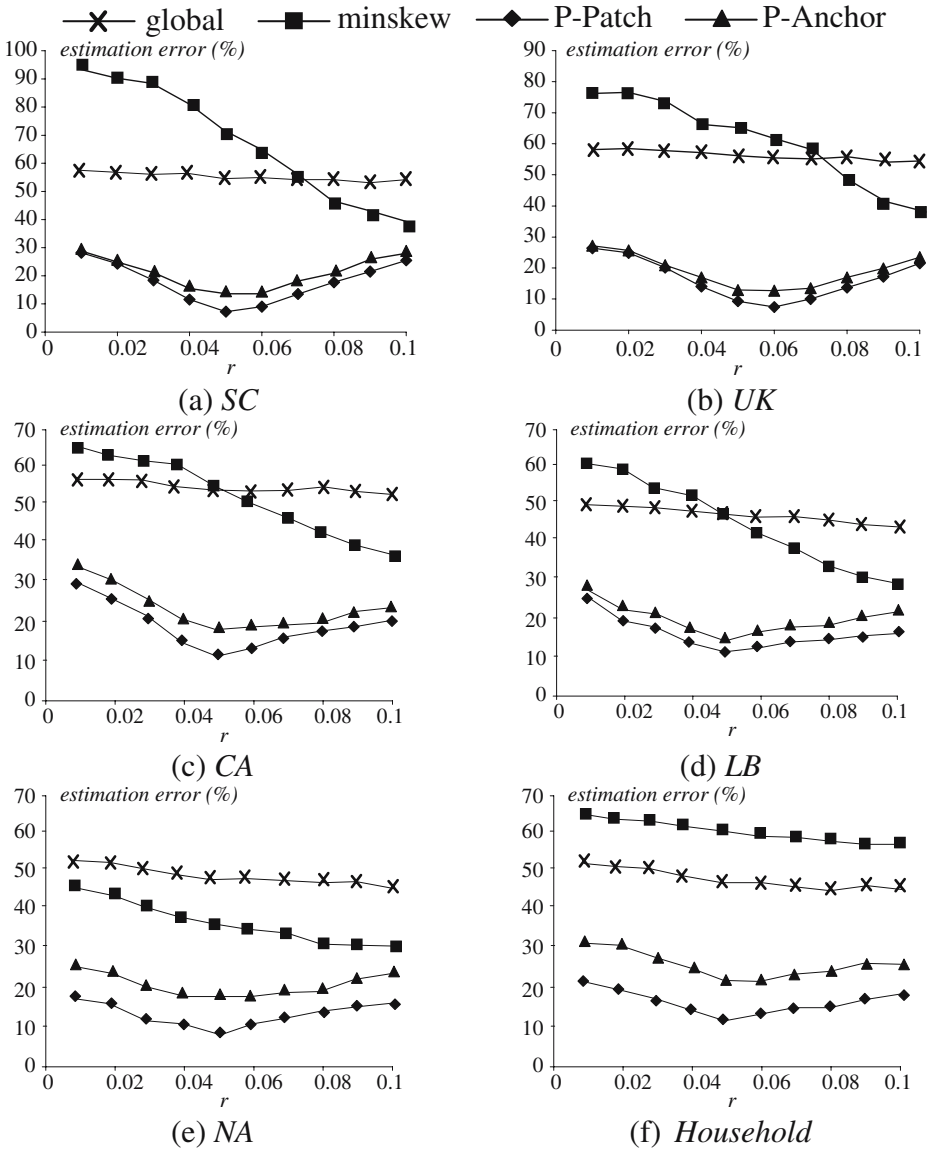


Figure 16 Error vs. query radius r (RS selectivity, L_2).

assumes local uniformity around the query's location. Figure 19 compares this model with the Power method and *global*, varying k from 1 to 100. The local uniformity assumption leads to substantial error, confirming the necessity of capturing local data correlation. *Global* has reasonable performance, because, according to Theorem 4.3, the cost of kNN queries is only affected by the local exponent at the query point. As a result, compared to other estimation problems, the variation of local constants does not introduce additional error.

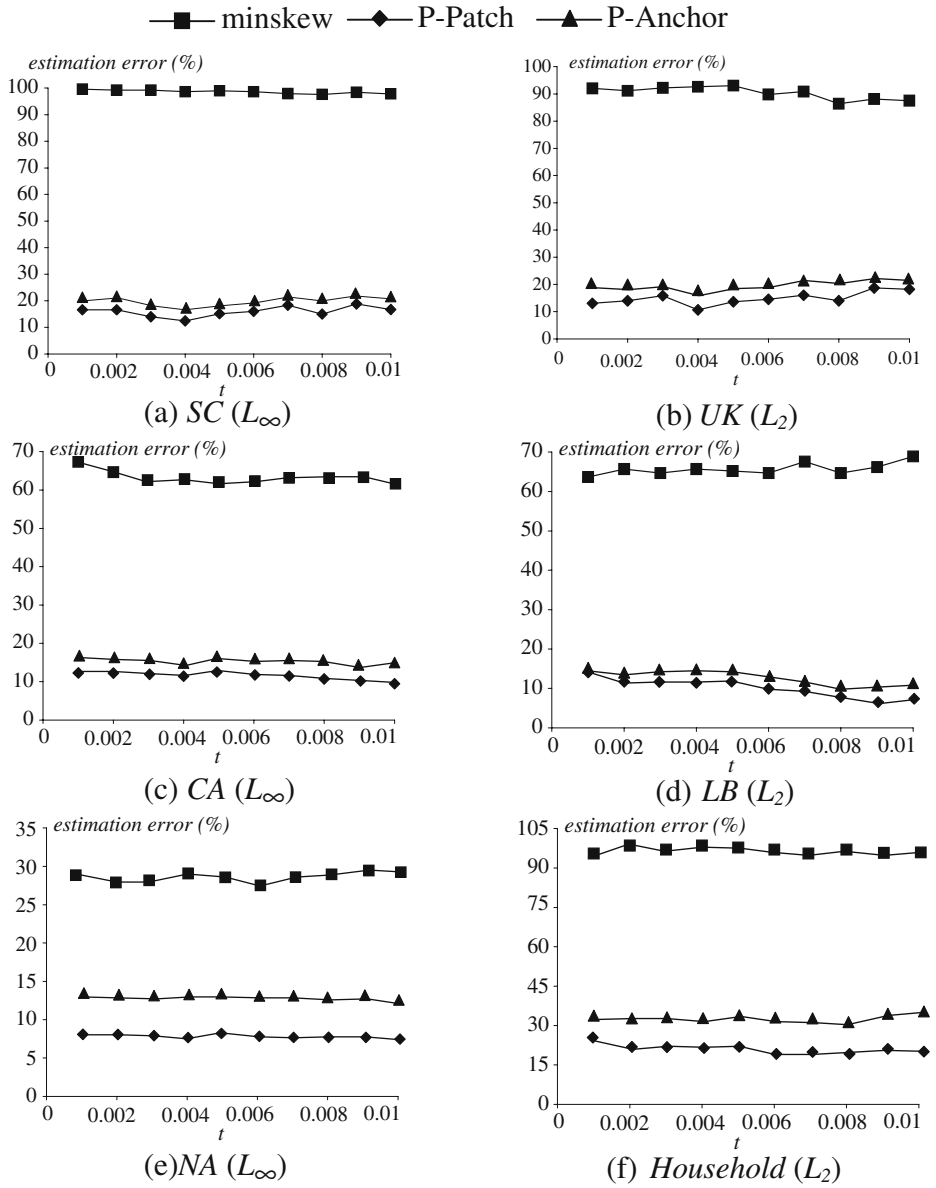


Figure 17 Error vs. distance threshold t (RDJ selectivity).

Interestingly, the two variations of the Power-Method have almost identical performance, which implies that P-Anchor captures the local exponent distribution well. In other words, its additional error (compared to P-Patch) in RS/CSJ estimation, is mainly due to the difference between the local constants of the query point and its nearest anchor point. The fact that the accuracy of the Power-Method improves with k can be explained as follows. For small k, the distance from the query to the k-th NN is

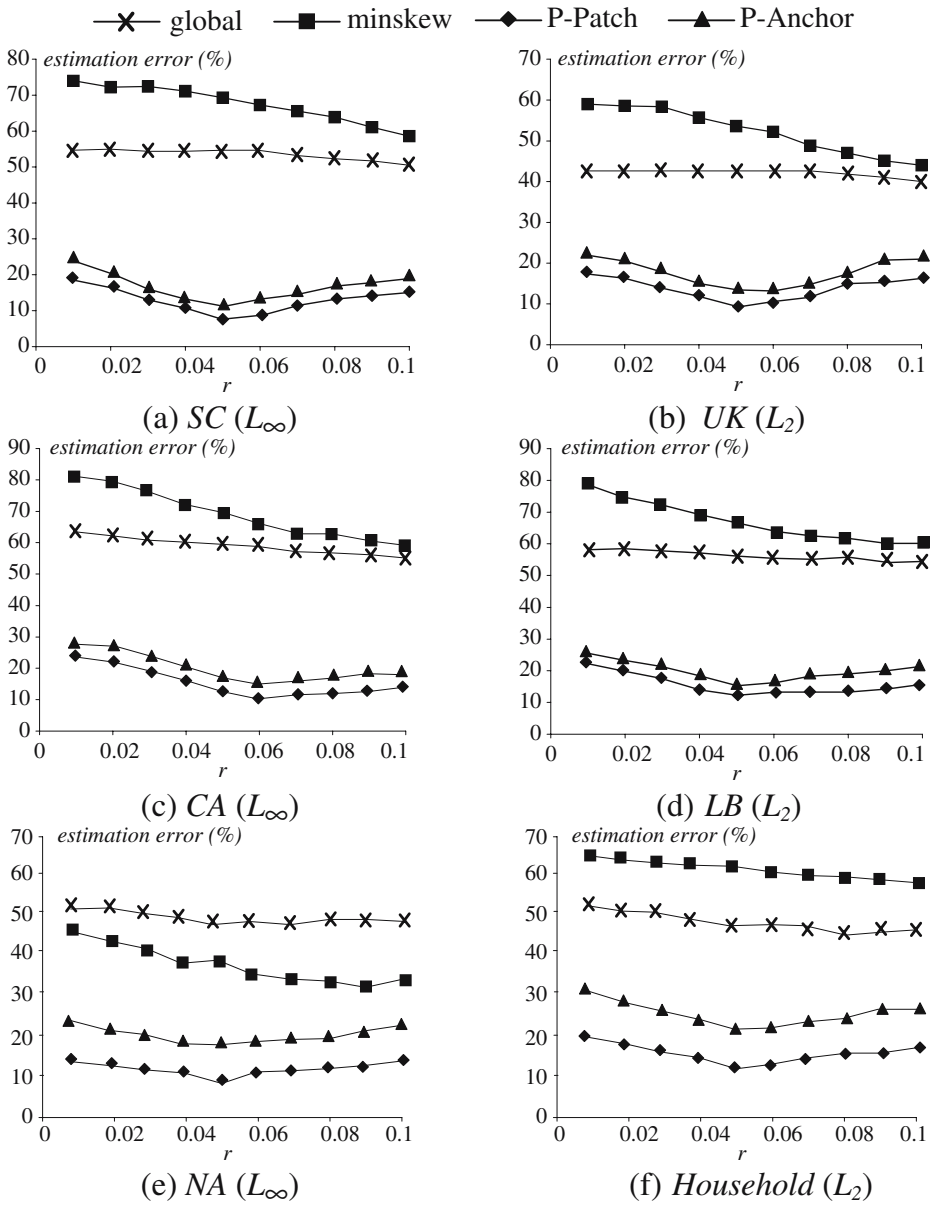


Figure 18 Error vs. query radius r (RS cost).

very short, and deviates from the scope of the LPLaw. As this distance increases with k, it is better modeled by the LPLaw, leading to more accurate prediction.

Figure 20 compares the precision of various methods for estimating RDJ cost (fixing r to 0.05), where the distance threshold t ranges from 0.001 to 0.01. The behavior of each solution is analogous to Figure 17, and the superiority of the Power-Method is obvious (*global* is again inapplicable).

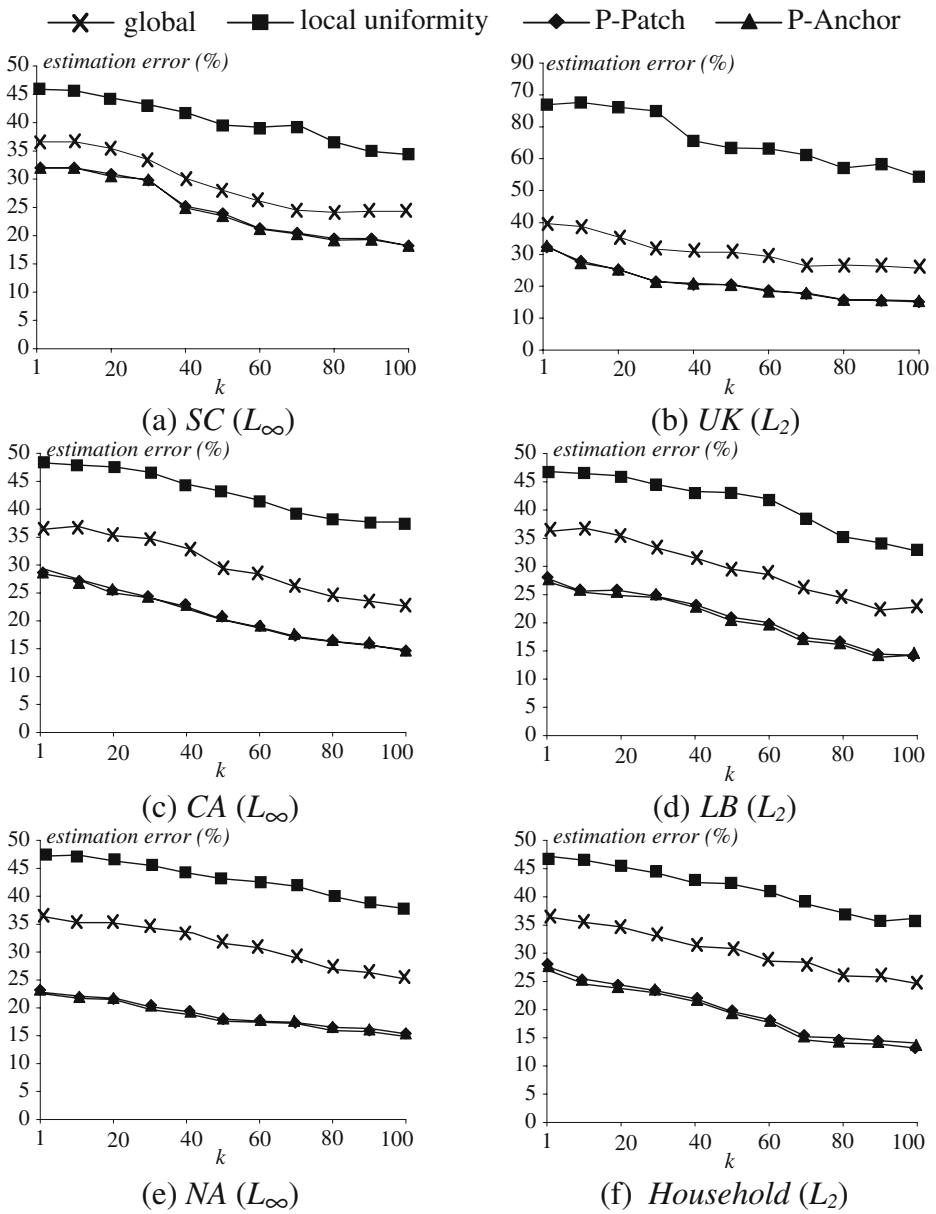


Figure 19 Error vs. k (kNN cost).

5.3 Sensitivity to parameters

In the previous sections, the memory consumption is fixed to the amount required by P-Patch (P-Anchor) to maintain 100 patches (anchors). In the sequel, we examine the influence of this parameter on accuracy. For this purpose, we repeat the above experiments by varying the number of patches (anchors) used by P-Patch (P-Anchor) from 50 to 150, and allowing the same size of memory for *minskew*

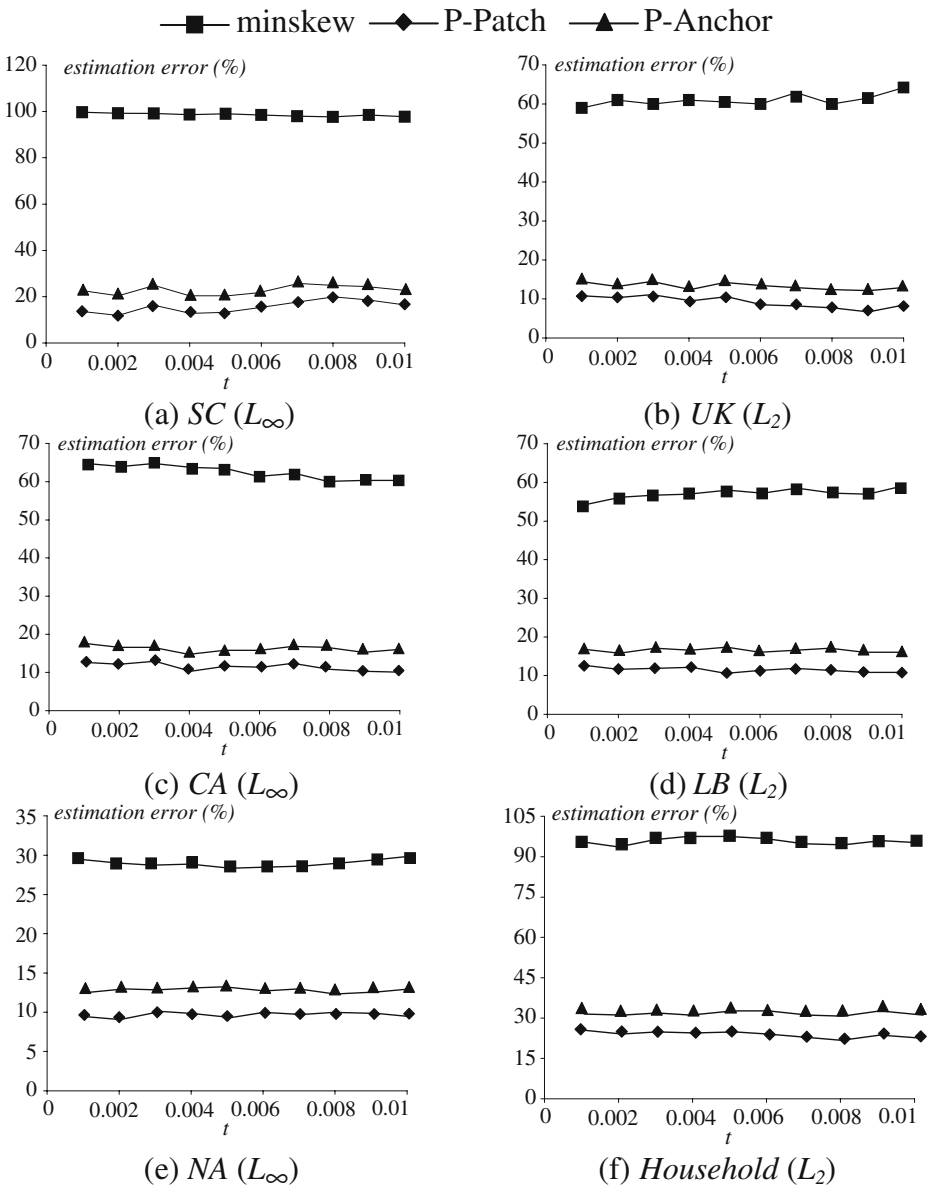


Figure 20 Error vs. dist threshold t (RDJ cost, $r = 0.05$).

(*global* is not included because it is not affected by the volume of available space). In the sequel, we present the diagrams for only *NA* and L_∞ metric (where *minskew* has the best accuracy), since the phenomena for the other datasets and L_2 norm are analogous. Figure 21 demonstrates the results for queries with median parameters (r = 0.05 for RS and RDJ, k = 50 for NN search, and t = 0.005 for RDJ). Evidently,

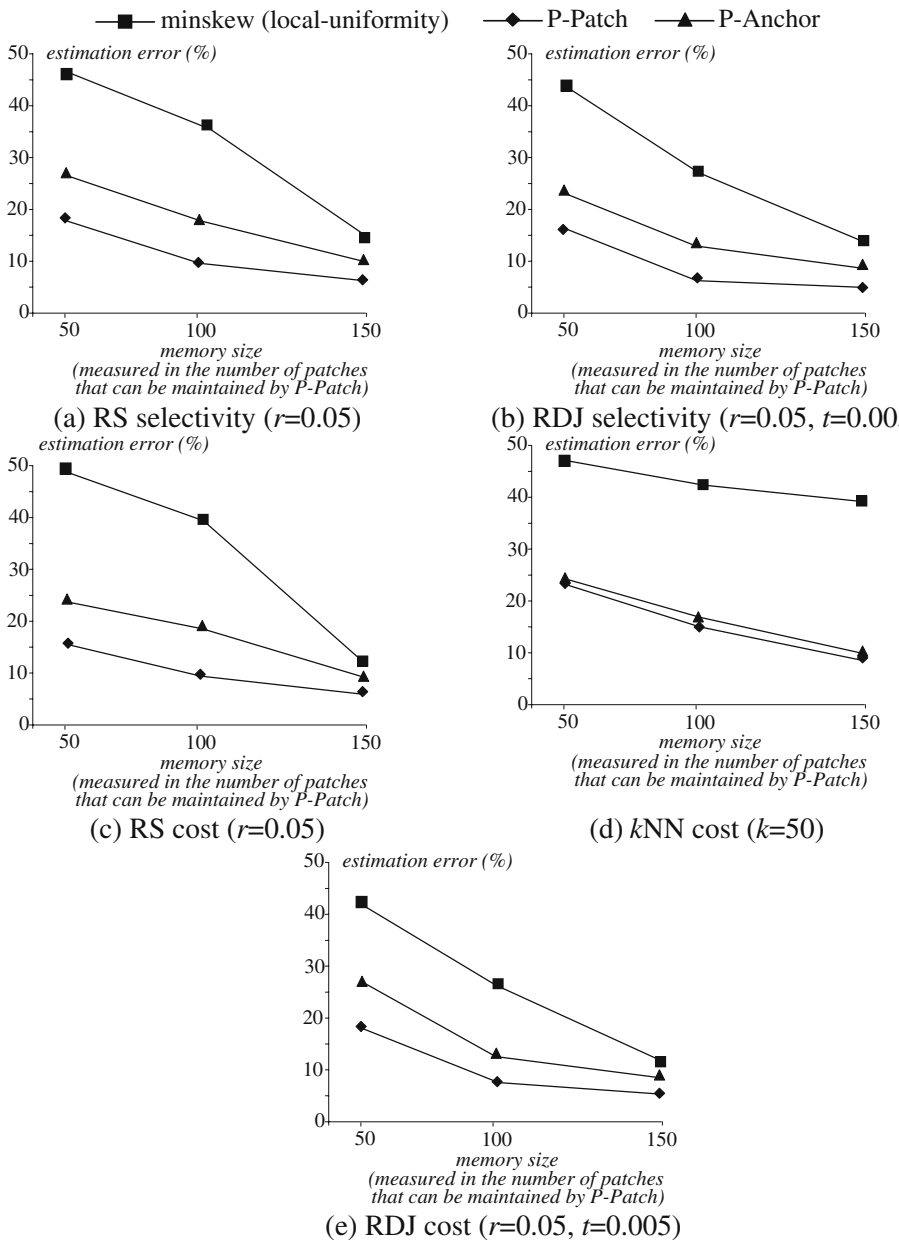


Figure 21 Error vs. space consumption (*NA*, L_∞).

all methods improve when more memory is permitted, while the proposed solutions still significantly outperform *minskew*.

For 2D (3D) datasets, the patches P-Patch are computed with an underlying grid containing 64×64 ($16 \times 16 \times 16$) cells so far. The next experiment examines the impact of the grid resolution. Fixing the number of buckets to 100, Figure 22a plots the error of P-Patch in estimating RS selectivity on *NA* (representative of the 2D datasets), and Figure 22b shows the results of the same experiment on *Household*, setting r to 0.05 in both cases. In the 2D case, when the resolution grows from 32 to 64, the error rate drops considerably, while the improvement is small as the resolution continues to increase. Similar observations exist in Figure 22b, where the stabilization happens at resolution 16. To explain the phenomena, recall that P-Patch has the best precision if the LPLaw coefficients at query point q are identical to those stored in the patch containing q. If the resolution is excessively low, the stored coefficients are averaged over a large number of points, thus leading to large error. On the other hand, when the resolution is already high enough, further increasing the resolution does not significantly decrease the coefficient variance in a cell, and hence, the error rate remains relatively the same. We do not include the results of the experiments with respect to the other estimation tasks, datasets, and L_2 norm because they demonstrate identical behavior.

5.4 Overhead of pre-processing and estimation

Apart from the *global* approach, the other methods require pre-computing necessary structures prior to performing estimation. Table 2 evaluates the pre-processing time for all solutions and methods, where the memory size is equivalent to that of 100 patches for P-Patch (grid resolution set to 64). *Minskew* is the fastest to build, which, however, does not justify its poor estimation accuracy. P-Anchor (due to its simple building algorithm) requires much less pre-processing time than P-Patch.

As discussed in Section 2.1, histograms (including *minskew*) may incur significant estimation overhead in L_2 metric. To verify this, Table 3 compares the prediction time of P-Patch, P-Anchor, and *minskew* for L_2 queries. The overhead of the Power-method is constant because only one patch or anchor is accessed in any case,

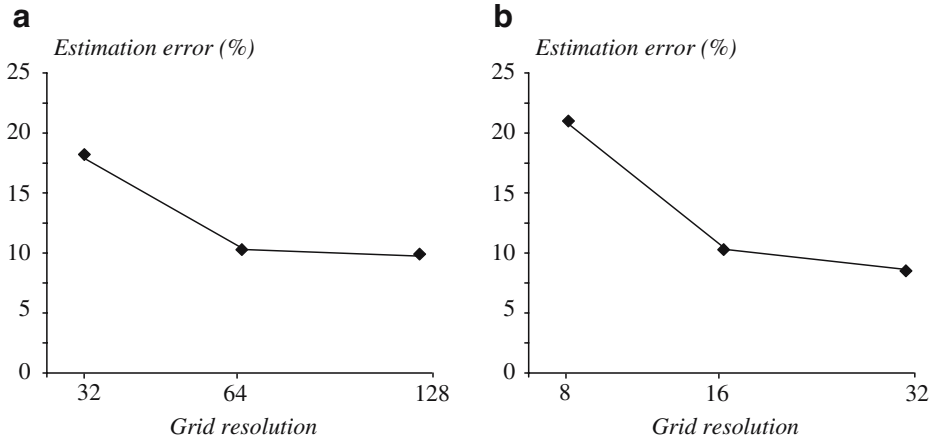


Figure 22 P-Patch Error vs. the resolution of the initial grid (L_∞, $r = 0.05$).

Table 2 Construction time (s).

Dataset	P-Patch	P-Anchor	Minskew
SC	32	18	15
UK	33	19	15
CA	70	38	32
LB	65	32	29
NA	129	57	49
Household	83	46	40

independent of the total number of patches or anchors. On the other hand, *minskew* is significantly more expensive, and its overhead increases with the query radius when more buckets partially intersect the query, and the expensive *monte-carlo* method must be performed more frequently. Note that the results presented here apply to all datasets and estimating both selectivity and cost.

As a conclusion of the experimental evaluation, the Power-Method is accurate and efficient for all estimation problems. P-Patch is more robust, but requires longer time to build, while P-Anchor is simple to maintain. Therefore, P-Patch could be used for relatively static databases and P-Anchor for intensive updates, especially if the data distribution or correlation changes frequently.

6 Conclusions and future work

The uniformity assumption is extremely easy to believe and analyze. Although it took many years before it was discredited, it is still lingering, hiding inside the "local uniformity assumption", which is the basis underneath most multi-dimensional histograms. In this paper, not only we spot the "density trap" problem of the local uniformity assumption, but we also show how to resolve it, using a novel perspective, which leads to the concept of Local Power Law. The advantages of LPLaw are:

- It accurately models real datasets.
- It works perfectly for Euclidean ones (smooth lines, smooth surfaces etc), including (unrealistic) uniform datasets.
- It leads to single-formula estimation methods for selectivities of all the popular query types, for both and L_2 distance metrics.
- It also leads to single-formula estimation methods for query I/O costs—something that no other published method can achieve.

We propose two methods based on the LPLaw, the P-patch and the P-anchor, which are simple to implement, and fast to initialize and run. Extensive experiments on several real datasets, confirm the effectiveness, efficiency and flexibility of our techniques.

Table 3 Estimation time for L_2 RS/RDJ queries (ms).

Query radius	P-Patch	P-Anchor	Minskew
0.01	0.5	0.5	3
0.03	0.5	0.5	5
0.05	0.5	0.5	9
0.07	0.5	0.5	15
0.09	0.5	0.5	25

An interesting direction for future work is to devise algorithms that utilize multiple patches (anchor points) based on *query partitioning*. For example, in RS selectivity, since the Power-Method is most accurate at $r = \rho$ (i.e., the radius of the neighborhood where the LPLaw is measured), we can divide a large query (regularly) into multiple parts with radii ρ. A separate estimate can be obtained for each partition and the sum of all parts constitutes the final answer. Furthermore, although in this paper we focused on a single point dataset, we plan to apply the Power-Method to joins between multiple sources, and datasets with rectangles or (poly) lines.

Query optimization and data mining are related, both looking for methods to concisely describe a dataset. The parameters of LPLaw do exactly that: they use the local coefficients to describe the vicinity of a point. Thus, these coefficients are suitable for data mining and pattern recognition tasks. For example, in Figure 8, the north-west patches (or anchors) in the 'Scandinavia' dataset have higher values, which, in retrospect, correspond to the Norwegian fjords. This is actually a rule ('northwest Norway has LPLaw exponents in the range 1.3–1.5), which can be used to detect outliers and extrapolate hidden/corrupted values: suppose that the very north part of Norway is not available—what can we say about it? Clearly, we can speculate that the points we are missing will have high LPLaw exponents. We believe that the above examples are just the beginning. Exploiting LPLaws for discovering patterns in real datasets (rules, outliers, clusters) is a very promising area of research.

Acknowledgments Yufei Tao was supported by RGC Grant CityU 1163/04E from the HKSAR government. Christos Faloutsos was supported by the National Science Foundation under Grants No. IIS-0209107, INT-0318547, SENSOR-0329549, EF-0331657, and IIS-0326322. Christos Faloutsos was also supported by the Pennsylvania Infrastructure Technology Alliance, a partnership of Carnegie Mellon, Lehigh University, the Commonwealth of Pennsylvania's Department of Community and Economic Development, Intel, and NTT. Dimitris Papadias was supported by RGC Grant HKUST 6178/04E from the HKSAR government.

References

1. S. Acharya, V. Poosala, and S. Ramaswamy. "Selectivity Estimation in Spatial Databases," In Proceedings of *ACM SIGMOD Conference*, 13–24, 1999.
2. N. An, Z. Yang, and A. Sivasubramaniam. "Selectivity Estimation for Spatial Joins," In Proceedings of *ICDE Conference*, 368–375, 2001.
3. W. Aref and H. Samet. "A Cost Model for Query Optimization Using R-Trees," In Proceedings of *ACM GIS Conference*, 1–8, 1994.
4. N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," In Proceedings of *ACM SIGMOD Conference*, 322–331, 1990.
5. A. Belussi and C. Faloutsos. "Estimating the Selectivity of Spatial Queries Using the Correlation's Fractal Dimension," In Proceedings of *VLDB Conference*, 299–310, 1995.
6. S. Berchtold, C. Bohm, D. Keim, and H. Kriegel. "A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space," In Proceedings of *ACM PODS Conference*, 78–86, 1997.
7. S. Berchtold, D. Keim, and H. Kriegel. "The X-tree: An Index Structure for High-Dimensional Data," In Proceedings of *VLDB Conference*, 28–39, 1996.
8. K. Beyer, J. Goldstein, and R. Ramakrishnan. "When Is "Nearest Neighbor" Meaningful?" In Proceedings of *ICDT Conference*, 217–235, 1999.
9. B. Blohsfeld, D. Korus, and B. Seeger. "A Comparison of Selectivity Estimators for Range Queries on Metric Attributes," In Proceedings of *ACM SIGMOD Conference*, 239–250, 1999.

10. C. Bohm. "A cost model for query processing in high dimensional data spaces," *ACM TODS*, Vol. 25(2):129–178, 2000.
11. T. Brinkhoff, H. Kriegel, and B. Seeger. "Efficient Processing of Spatial Joins Using R-trees," In Proceedings of *ACM SIGMOD Conference*, 237–246, 1993.
12. N. Bruno, L. Gravano, and S. Chaudhuri. "STHoles: A Workload Aware Multidimensional Histogram," In Proceedings of *ACM SIGMOD Conference*, 211–222, 2001.
13. S. Chaudhuri, G. Das, M. Datar, R. Motwani, and V. Narasayya. "Overcoming Limitations of Sampling for Aggregation Queries," In Proceedings of *IEEE ICDE Conference*, 534–542, 2001.
14. A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos. "Closest Pair Queries in Spatial Databases," In Proceedings of *ACM SIGMOD Conference*, 189–200, 2000.
15. A. Deshpande, M. Garofalakis, and R. Rastogi. "Independence Is Good: Dependency-Based Histogram Synopses for High-Dimensional Data," In Proceedings of *ACM SIGMOD Conference*, 199–210, 2001.
16. C. Faloutsos and I. Kamel. "Beyond Uniformity and Independence, Analysis of R-trees Using the Concept of Fractal Dimension," In Proceedings of *ACM PODS Conference*, 4–13, 1994.
17. C. Faloutsos, B. Seeger, A. Traina, and C. Traina. "Spatial Join Selectivity Using Power Laws," In Proceedings of *ACM SIGMOD Conference*, 177–188, 2000.
18. D. Gunopulos, G. Kollios, V. Tsotras, and C. Domeniconi. "Approximate Multi-Dimensional Aggregate Range Queries over Real Attributes," In Proceedings of *ACM SIGMOD Conference*, 463–474, 2000.
19. J. Jin, N. An, and A. Sivasubramaniam. "Analyzing Range Queries on Spatial Data," In Proceedings of *IEEE ICDE Conference*, 525–534, 2000.
20. J. Lee, D. Kim, and C. Chung. "Multidimensional Selectivity Estimation Using Compressed Histogram Information," In Proceedings of *ACM SIGMOD Conference*, 205–214, 1999.
21. H. Lin and B. Huang. "Sql/sda: a query language for supporting spatial data analysis and its web-based implementation," *IEEE TKDE*, Vol. 13(4):671–682, 2001.
22. Y. Mattias, J. Vitter, and M. Wang. "Wavelet-Based Histograms for Selectivity Estimation," In Proceedings of *ACM SIGMOD Conference*, 448–459, 1998.
23. Y. Mattias, J. Vitter, and M. Wang. "Dynamic Maintenance of Wavelet-Based Histograms," In Proceedings of *VLDB Conference*, 101–110, 2000.
24. M. Muralikrishna and D. DeWitt. "Equi-Depth Histograms for Estimating Selectivity Factors for Multi-Dimensional Queries," In Proceedings of *ACM SIGMOD Conference*, 28–36, 1998.
25. F. Olken and D. Rotem. "Random Sampling from Database Files: A Survey," In Proceedings of *IEEE SSDBM Conference*, 92–111, 1990.
26. B. Pagel, F. Korn, and C. Faloutsos. "Deflating the Dimensionality Curse using Multiple Fractal Dimensions," In Proceedings of *IEEE ICDE Conference*, 589–598, 2000.
27. B. Pagel, H. Six, H. Toben, and P. Widmayer. "Towards an Analysis of Range Query Performance in Spatial Data Structures," In Proceedings of *ACM PODS Conference*, 214–221, 1993.
28. C. Palmer and C. Faloutsos. "Density Biased Sampling: An Improved Method for Data Mining and Clustering," In Proceedings of *ACM SIGMOD Conference*, 82–92, 2000.
29. Y. Poosala and Y. Ioannidis. "Selectivity Estimation without the Attribute Value Independence Assumption," In Proceedings of *VLDB Conference*, 486–495, 1997.
30. Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima. "The A-tree: An Index Structure for High-Dimensional Spaces Using Relative Approximation," In Proceedings of *VLDB Conference*, 516–526, 2000.
31. S. Shekhar, M. Coyle, D. Liu, B. Goyal, and S. Sarkar. "Data models in geographic information systems," *Communication of the ACM*, Vol. 40(4), 1997.
32. C. Sun, D. Agrawal, and A. El Abbadi. "Exploring Spatial Datasets with Histograms," In Proceedings of *IEEE ICDE Conference*, 93–102, 2002.
33. N. Thaper, S. Guha, P. Indyk, and N. Koudas. "Dynamic Multidimensional Histograms," In Proceedings of *ACM SIGMOD Conference*, 428–439, 2002.
34. Y. Theodoridis and T. Sellis. "A Model for the Prediction of R-tree Performance," In Proceedings of *ACM PODS*, 161–171, 1996.
35. Y. Theodoridis, E. Stefanakis, and T. Sellis. "Cost Models for Join Queries in Spatial Databases," In Proceedings of *IEEE ICDE Conference*, 476–483, 1998.
36. TIGER, http://www.census.gov/geo/www/tiger/.
37. Y. Wu, D. Agrawal, and A. El Abbadi. "Applying the Golden Rule of Sampling for Query Estimation," In Proceedings of *ACM SIGMOD Conference*, 449–460, 2001.

Yufei Tao is an assistant professor at the Department of Computer Science and Engineering, the Chinese University of Hong Kong (CUHK). Before joining CUHK, he was a visiting scientist at the Carnegie Mellon University during 2002–2003 and an assistant professor at the City University of Hong Kong during 2003–2006. He is the winner of the Hong Kong Young Scientist Award 2002. His research interests include temporal databases, spatial databases, data privacy and security.

Christos Faloutsos is a Professor at Carnegie Mellon University. He has received the Presidential Young Investigator Award by the National Science Foundation (1989), seven best paper awards, and several teaching awards. He has served as a member of the executive committee of SIGKDD; he has published over 140 refereed articles, one monograph, and holds five patents. His research interests include data mining for streams and networks, fractals, indexing for multimedia and bio-informatics data bases, and performance.

Dimitris Papadias is a professor at the Computer Science Department, Hong Kong University of Science and Technology. Before joining HKUST in 1997, he worked and studied at the German National Research Center for Information Technology (GMD), the National Center for Geographic Information and Analysis (NCGIA, Maine), the University of California at San Diego, the Technical University of Vienna, the National Technical University of Athens, Queen's University (Canada), and University of Patras (Greece). He has published extensively and been involved in the program committees of all major Database Conferences, including SIGMOD, VLDB and ICDE. He is an associate editor of TKDE, VLDB Journal, and in the advisory editorial board of Information Systems.