# Tight bounds for 2-dimensional indexing schemes

**Elias Koutsoupias**
University of California, Los Angeles
elias@cs.ucla.edu

**David Scot Taylor**
University of California, Los Angeles
dstaylor@cs.ucla.edu

**Abstract**

We study the trade-off between storage redundancy and access overhead for range queries, using the framework of [6]. We show that the Fibonacci workload of size $n$, which is the regular 2-dimensional grid rotated by the golden ratio, does not admit an indexing scheme with access overhead less than the block size $B$ (the worst possible access overhead), even for storage redundancy as high as $c\log n$, for some constant $c$. We also show that this bound is tight (up to a constant factor) by providing an indexing scheme with storage redundancy $\Theta(\log n)$ and constant access overhead, for any 2-dimensional workload. We extend the lower bound to random point sets and show that if the maximum storage redundancy is less than $c$loglog n, the access overhead is $B$. Finally, we explore the relation between indexability and fractal (Hausdorff) dimension of point sets.

## 1 Introduction

In this paper we continue the work of [6] towards a theory of indexability —that is, towards a better understanding of the storage redundancy/access overhead tradeoff in the indexing problem for complex workloads, in a secondary memory device with a fixed block size $B$. Since data items are stored and retrieved in blocks of $B$, a query may retrieve many more blocks than the size of the answer set divided by $B$ —in the worst case, $B$ times more. This is the access overhead of the indexing scheme. To decrease the access overhead, we may choose to store data redundantly, increasing storage costs —hence the tradeoff. B-trees owe their widespread success to the fact that they are the perfect solution to this problem —unfortunately, only for 1-dimensional workloads. The deterioration in the performance of B-trees on multidimensional workloads led to intensive research on secondary memory data structures that perform well in higher dimensions [5, 12, 16, 19]. In fact, in [7], a generalized indexing system (GiST) was proposed and implemented. A conclusion of this work was the need of a mathematical methodology that would evaluate rigorously the power and limitations of indexing techniques. Such a framework was proposed in [6] where the notion of an indexing scheme was introduced, as well as a notion of "data storage complexity," as opposed to the computational complexity that had been studied so far. The efficiency of an indexing scheme is captured by two salient parameters: *storage redundancy* (how many times each item in the data set is stored), and *access overhead* (how many times more blocks than necessary does a query retrieve).

To introduce the notion of an indexing scheme we first need a few definitions: A workload $W = (D, I, \mathcal{Q})$ consists of a finite subset $I \subseteq D$ ($I$ is called the instance and $D$ the domain) together with a set $\mathcal{Q} \subseteq 2^I$ of queries. For example, the workloads that we consider here are the 2-dimensional range queries, for which $I$ is a finite set of points of the 2-dimensional Euclidean space ($D = R^2$), and $\mathcal{Q}$ consists of all intersections of $I$ with rectilinear rectangles.

Intuitively, an indexing scheme for a workload $W = (D, I, \mathcal{Q})$ is a way to store (possibly in multiple copies) the elements of the instance $I$ in the secondary memory in blocks of size $B$. The objective is to be able to answer (cover) each query in $\mathcal{Q}$ by retrieving few blocks (see [6] for a more extensive discussion). The storage redundancy $r$ of an indexing scheme measures the number of copies of the elements of $I$ that are stored in the secondary memory (there are two kinds, the maximum redundancy and the average redundancy). The access overhead $a$ of an indexing scheme measures how many times more blocks we have to retrieve to answer a query $Q \in \mathcal{Q}$ compared with the ideal $|Q|/B$. Predictably, there is a tradeoff between the redundancy and the access overhead of an indexing scheme: the higher the redundancy, the lower the access overhead should be.

Using this framework, [6] study indexing schemes for range queries. In particular, they study workloads of 2-dimensional range queries of the instance $I$ that consists of the $k \times k$ grid points. They show that an indexing scheme that achieves access overhead $a$ must have storage redundancy $\Omega(\log B/(a^2 \log a))$. They conjecture that other 2-dimensional workloads that are not restricted to the $k \times k$ grid may have worst tradeoffs. Here, we confirm their conjecture, and in fact in the strongest possible way: We show that if we rotate the points of the above workload —as it happens, by the *golden ratio* $\phi = (-1 + \sqrt{5})/2$— then any indexing scheme for range queries must have storage redundancy logarithmic in the number of points if it is to achieve access overhead less than the block size.

Before [6], there was substantial work on the indexing problem [9, 14, 15, 18, 21]. Most of this work involves upper bounds, and is therefore mainly concerned with the analysis of the searching aspect of the problem. There is however a

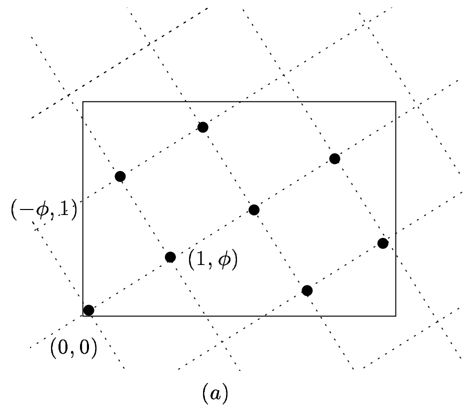

Figure 1: The Fibonacci lattice and $F_8$.

notable result on lower bounds that appears in the last section of [18] and resembles our lower bound. It is shown that there is a 2-dimensional point set such that storage redundancy $\Omega(\log n/\log\log n)$ is necessary if *additive* (as opposed to multiplicative in [6]) access overhead is to remain polynomial in $\log n/\log B$.

## Our results

We provide a tight bound for the tradeoff between storage redundancy and access overhead of 2-dimensional workloads. We show that there are "natural" sets of points on the plane (the *Fibonacci workloads*) such that any indexing scheme must have storage redundancy logarithmic ($r > c\log n$) in the total number of points $n$ to achieve access overhead less than the worst possible bound $B$. Thus, any indexing scheme with storage redundancy less than $c\log n$ has in the worst case the same performance with an indexing scheme with storage redundancy 1. Surprisingly, there are indexing schemes with slightly higher redundancy $r = 4\log(n/B)$ that achieve constant access overhead! Thus the tradeoff between redundancy and access overhead for a Fibonacci workload exhibits a "threshold" behavior.

In Section 2, we formally introduce the Fibonacci workloads and prove our lower bound on the redundancy/access tradeoff. The proof is based on some important properties of the Fibonacci workload, mainly on its discrepancy properties: Any rectilinear rectangle with area $A$ contains approximately $A$ points of the workload, independently of its shape. This property of the Fibonacci workload has been used before in many contexts, for example in [4]. Using this property, we show that a Fibonacci workload of $n$ points has $\Theta(n\log n)$ queries of size $B$. We also show that each pair of points cannot be in many queries of size $B$. The lower bound then follows from these two properties. In Section 3, we give the upper bound that shows that the redundancy/access tradeoff of the Fibonacci workload is the worst possible among all 2-dimensional workloads (up to a constant factor). We also present a simple extension of the lower bound to random 2-dimensional workloads (with required redundancy loglog n).

Mandelbrot [13] provides strong evidence that many natural phenomena exhibit self-similarity. Adapting this approach, Faloutsos and Kamel proposed in [3] that many database workloads have also some self-similarity; thus, for these workloads their fractal (Hausdorff) dimension exists and may be an important parameter. In Section 4, we discuss possible relations between the fractal dimension of a point set and its indexability. We argue that indexability is "orthogonal" to fractal dimension in the sense that the fractal dimension provides absolutely no information about the indexing properties of a point set (with respect to range queries). Although our argument seems in conflict with evidence presented in [1], there is no actual incompatibility because [1] investigates square queries. Finally, in Section 5 we extend the main result of [6] to $d > 2$ dimensions and we close the paper with some open problems.

## 2 Fibonacci workloads

In this paper, we consider only workloads of range queries on a Euclidean space. These workloads are simply determined by their set of points and for this reason we will use identical names for workloads and their sets of points. Consider a workload $C$ of a Euclidean space. Notice that the indexing properties of $C$ do not change if we stretch it out and, in fact, the same is true if we stretch out any stripe of the Euclidean space — a stripe is a set of points whose projection on some dimension $x_i$ belongs to a given interval. We can say that the resulting workload is equivalent to the original one. This motivates the following general definition:

**Definition 1** *Two workloads $C$ and $C'$ are equivalent iff there is a one-to-one mapping $f$ between points of $C$ and $C'$ such that for any subset $Q$ of workload $C$, $Q$ is a query of workload $C$ iff $f(Q)$ is a query of workload $C'$.*

For simplicity, we will concentrate on 2-dimensional range queries, since, in most cases, there is an obvious generalization to higher dimensions. Fix some 2-dimensional workload $C$ of size $n$. Assume for the moment that no two points are in the same horizontal or vertical line. We can then move the points of $C$ so that the points have integer coordinates $0, 1, \ldots, n-1$ and the resulting point set $C'$ is equivalent to $C$. The point set $C'$ spans the grid-lines with coordinates $0, 1, \ldots, n-1$ and it has one point on each horizontal and vertical grid-line. We will call such a point set a *permutation.* It should be clear that not all finite point sets are equivalent to a permutation, because there may be more than one point on a horizontal (or vertical) line. In the other extreme, lies the point set studied in [6] where every horizontal and vertical line has $k = \sqrt{n}$ points. It is not difficult to see that, with respect to indexing, the worst workloads are permutations. More precisely, given a 2-dimensional workload $W$, we can slightly perturb its points so that no two points are in a horizontal or vertical line. The resulting workload is a permutation $W'$ with no better redundancy/access overhead tradeoff. The reason is that any query of $W$ is also a query of $W'$; the inverse is not in general true. Here, we will only consider workloads that are permutations.

A natural question to ask is what point set and in particular what permutation has the worst tradeoff between redundancy and access overhead. We will introduce now a family of permutations, the *Fibonacci* workloads, and we will show that they are indeed the worst workloads (up to a constant factor). The *Fibonacci lattice*, depicted in Figure 1.*a*, is the 2-dimensional lattice with base $\{(1, \phi), (-\phi, 1)\}$, where $\phi = (-1 + \sqrt{5})/2$ is the golden ratio. This lattice has some amazing properties and we will employ some of them here. It plays an important role in the discrepancy theory [8, 11, 4] and in the theory of Diophantine approximation [8]. Our workloads must be finite, so they contain some small part of the Fibonacci lattice. In particular, we will consider workloads whose points are the Fibonacci lattice points of some appropriate rectangles and have sizes $n = f_k$, where $f_k$ is the $k$-th Fibonacci number. The Fibonacci workload of $n$ points, which we will denote by $F_n$, is the equivalent permutation (see Figure 1). More precisely, for $n = f_k$, the Fibonacci workload $F_n$ of order $n$ is the following permutation:

$$F_n = \{(k, k f_{k-1} \, mod \, n) : k = 0, 1, \ldots, n-1\}.$$

The properties of Fibonacci lattice that we will employ in our proofs can be found in [4, 2]. They are summarized in the following proposition.

**Proposition 1** *For the Fibonacci workload $F_n$ of $n$ points*

*a. Every rectangle with area $c_1 n$ contains at least one point of $F_n$, where $c_1 \approx 1.9$.*

*b. Every rectangle with area $c_2 n$ contains at most two points of $F_n$, where $c_2 \approx 0.45$.*

Notice that the proposition implies that a rectangle with area $tn$ contains between $\lfloor t/c_1 \rfloor$ and $\lceil t/c_2 \rceil$ points of $F_n$. For large $t$ however the number of points of $F_n$ within a rectangle with area $tn$ is very close to $t$ [4, 2]. In fact, *any* lattice with base $\{(1, \alpha), (-\alpha, 1)\}$, where $\alpha$ is an (irrational) quadratic number, has similar properties. The constants in Proposition 1 depend on the continued fraction expansion of $\alpha = [a_1, a_2, \ldots]$ and in particular on the maximum $a_i$. As a result, any irrational number with bounded $a_i$'s satisfies Proposition 1 with different constants $c_1$ and $c_2$; the quadratic numbers have bounded $a_i$'s simply because their continued fraction expansion is periodic. All our results about Fibonacci workloads hold also for all these lattices (of course, with different constants). The connection between Proposition 1 and the continued fraction expansion is related to the remarkable *three-distance theorem* (see [10, 20]). We will expand on this in the final version of the paper, but we add now that the Fibonacci lattice has the best constants $c_1$ and $c_2$, because the maximum $a_i$ is the smallest possible ($\phi = [0, 1, 1, 1, \ldots]$).

Before we present our main theorem, we digress to discuss the $O$ notation in this paper. To keep expressions simple, we treat the block size $B$ as a constant, so, for example, the expression $O(n)$ may conceal some multiplicative factor that depends on $B$. Our aim is to show the dependency of the indexing tradeoff on the workload size, not the block size. Our main theorem shows that indexing has an inherent $\log n$ tradeoff.

**Theorem 1** *For each block size $B > 4$, there exists a positive constant $c$ such that no indexing scheme for the Fibonacci workload $F_n$ that has average redundancy $r \leq c \log n$ has access overhead $a < B$.*

The proof of the theorem is a direct application of the following two lemmata. The first lemma estimates the total number of queries, while the second one provides an estimate of queries that contain a pair of points.

**Lemma 1** *For every $B > 4$, the total number of range queries of size $B$ of the Fibonacci workload $F_n$ is $\Omega(n \log n)$.*

**Lemma 2** *For every $B$, the number of range queries of size $B$ that contain a given pair of points of the Fibonacci workload $F_n$ is constant (independent of $n$).*

We postpone the proof of the lemmata to first show how Theorem 1 can be derived from them.

**Proof of Theorem 1.** We consider only queries of size $B$. The theorem follows from the fact that Lemma 1 asserts that the number of these queries is large, $d_1 n \log n$ —for some positive constant $d_1$— while Lemma 2 guarantees that each pair in a block belongs to few queries, i.e. to at most $d_2$ queries, for some constant $d_2$.

More precisely, a range query of size $B$ that has access overhead less than $B$ contains at least a pair that belongs to the same block. The total number of these queries is at most $r \frac{n}{B} \binom{B}{2} d_2$ (there are $r \frac{n}{B}$ blocks, each one containing at most $\binom{B}{2}$ pairs, and each pair belongs to at most $d_2$ queries). On the other hand, the total number of queries of size $B$ is at least $d_1 n \log n$. So, as long as $r \frac{n}{B} \binom{B}{2} d_2 < d_1 n \log n$, there is at least one query that has access overhead exactly $B$. Equivalently, if $r < c \log n$, for $c \approx d_1/(d_2 B)$, then some query has access overhead $a = B$. □

It is worth mentioning that Theorem 1 cannot be derived from Lemma 1 based solely on the number of queries. In fact, it is not hard to come up with 2-dimensional point sets that have $n^2$ queries of size $B$ but constant redundancy and access overhead (all points are arranged into two slanted parallel lines). We now proceed with the proofs of Lemmata 1 and 2.

Theorem 1 guarantees that there are queries that have the worst possible access overhead $B$. In fact, from the proof we can extract a much stronger statement. It follows from the proof that only a fraction of $(r \frac{n}{B} \binom{B}{2} d_2)/(d_1 n \log n)$ of queries of size $B$ can have access overhead less that $B$. Thus, if $r = o(\log n)$ then the vast majority of queries of size $B$ has access overhead $B$. Equivalently:

**Corollary 1** *For any indexing scheme for the Fibonacci workload $F_n$ with $B > 4$ and storage redundancy $r = o(\log n)$, the probability that a random query of size $B$ has access overhead $a = B$ is $1 - o(1)$.*

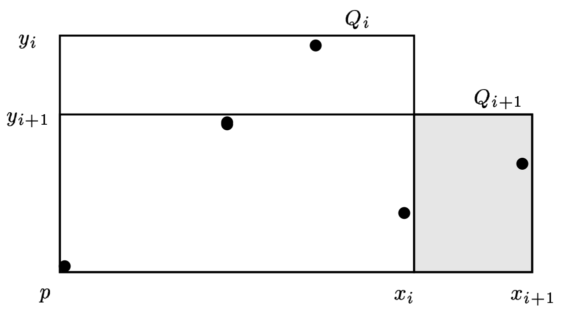

Figure 2: Successive queries.

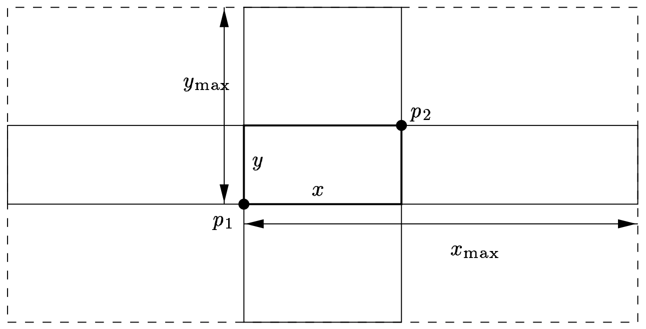

Figure 3: Queries that contain two points $p_1$ and $p_2$.

**Proof of Lemma 1.** Fix a point $p$ of $F_n$. We will consider the queries of size $B$ for which $p$ is the lowest, leftmost point. Such a query is defined by its rightmost point and its highest point that are not necessarily distinct. Let $Q_1, Q_2, \ldots, Q_k$ be these queries and let $x_i$ and $y_i$ denote the width and height of $Q_i$. We assume that the queries are ordered by their width, i.e., $x_{i+1} > x_i$. Our aim is to find a relation between $x_i$ and $x_{i+1}$.

Notice first that $Q_{i+1}$ results from $Q_i$ if we remove the highest point of $Q_i$ and add a point to its right (see Figure 2). Notice also that the point set $Q_{i+1} - Q_i$ is a rectangle that contains exactly one point of $F_n$. Therefore, by Proposition 1.a, the area $(x_{i+1} - x_i)y_{i+1}$ of this rectangle is at most $c_1 n$. Because query $Q_{i+1}$ contains $B$ points, its area $x_{i+1}y_{i+1}$ is, by Proposition 1.b, at least $c_2 Bn$. From these, we get the following relation between $x_i$ and $x_{i+1}$:

$$\frac{x_{i+1} - x_i}{x_{i+1}} \leq \frac{c_1}{c_2 B}.$$

Solving this recurrence, we get $x_i \leq (\frac{c_2 B}{c_2 B - c_1})^i x_1$. In particular,

$$x_k \leq (\frac{c_2 B}{c_2 B - c_1})^{k-1} x_1. \tag{1}$$

Assume now that the point $p$ is in the lower left quadrant. We will show that the number $k$ of queries that contain $p$ as the lowest, leftmost query is large. For this, it suffices to show that $x_k/x_1$ is large. We first find a (constant) upper bound of $x_1$. By Proposition 1.a, we have $x_1 y_1 \leq c_1 Bn$ and since $y_1$ is at least $n/2$, we get $x_1 \leq 2c_1 B$. Similarly, we get that $y_k \leq 2c_1 B$ and, by Proposition 1.b, $x_k y_k \geq c_2 Bn$. Thus, $x_k \geq c_2 n/(2c_1)$ is linear in $n$. Putting everything together, we get that $x_k/x_1$ is linear in $n$:

$$x_k/x_1 \geq c_2 n/(4c_1^2 B). \tag{2}$$

By (1) and (2), we get that there is a constant $c \approx \log(\frac{c_2 B}{c_2 B - c_1})$ such that $k \geq c \log n$. In conclusion, every point $p$ in the lower left quadrant is the lowest, leftmost point of at least $c \log n$ queries of size $B$. By symmetry every point is an "extreme" (lowest/highest and leftmost/rightmost) point of $c \log n$ queries. Since a query cannot have more than 2 "extreme" points we conclude that there are at least $\frac{1}{2} cn \log n = \Omega(n \log n)$ queries of size $B$. □

It is worth mentioning that the bound of Lemma 1 is tight (up to a constant factor). We now prove Lemma 2.

**Proof of Lemma 2.** Let $p_1$ and $p_2$ be points of $F_n$. We will show that there is only a constant number (independent of $n$) of queries of size $B$ that contain both $p_1$ and $p_2$. The main idea is that only a constant number of points can be in a query of size $B$ with both $p_1$ and $p_2$.

Let $x$ and $y$ be the horizontal and vertical distance of $p_1$ and $p_2$. Also, let $x_{\max}$ be the width of the widest query $Q$ that contains both $p_1$ and $p_2$ (see Figure 3). Since $Q$ has height at least $y$ and width $x_{\max}$, an application of Proposition 1.a gives $x_{\max} y \leq c_1 Bn$. Similarly we define $y_{\max}$ and we get $xy_{\max} \leq c_1 Bn$. Multiplying we get that $xyx_{\max}y_{\max} \leq c_1^2 B^2 n^2$. On the other hand, by Proposition 1.b, we have $xy \geq c_2 n$. Thus, $x_{\max}y_{\max} \leq c_1^2 B^2 n/c_2$.

So, every point that is in a query with both $p_1$ and $p_2$ lies in a rectangle of dimensions $(2x_{\max}) \times (2y_{\max})$ that extends around $p_1$ and $p_2$. But, by Proposition 1.b, a rectangle of dimensions $(2x_{\max}) \times (2y_{\max})$ contains at most $4x_{\max}y_{\max}/(c_2 n) \leq 4c_1^2 B^2/c_2^2$ points. Thus, the number

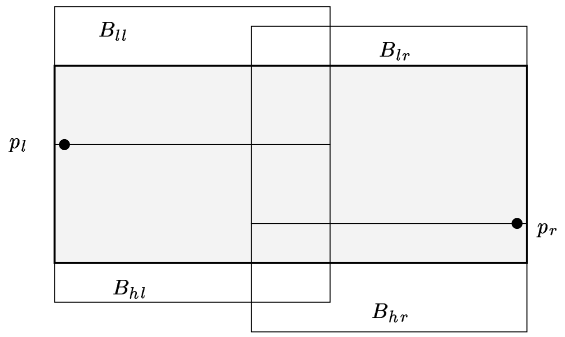

Figure 4: Covering a query with 4 blocks.

of points that can be in a query of size $B$ with both $p_1$ and $p_2$ is constant; therefore, the number of queries that contain these points is also constant. In fact, the number of queries is at most $(4c_1^2 B^2/c_2^2)^3$ since a query of fixed size $B$ is defined by three points (its leftmost, lowest, and highest point).

Using a proof very similar to the proof of Lemma 1, we can obtain a tighter result: the number of queries that contain a given pair of points is at most $c_3 B^3 \log B$, for some constant $c_3$. $\square$

The constant $c$ in Theorem 1 that results from the above proofs is very small for practical values of $B$ and $n$. Our aim was to provide simple proofs for Theorem 1, instead of trying to squeeze out better constants.

## 3 Upper Bounds and Random Point Sets

Theorem 1 asserts that we need logarithmic redundancy ($r > c \log n$) to achieve access overhead less than $B$ for the Fibonacci workload $F_n$. It is natural to ask whether this bound can be improved or whether there is a worse workload than $F_n$. The answer is provided by the following theorem where we show that with a little more ($r = 4\log(n/B)$) we can achieve constant access overhead, $a = 4$, for *all* 2-dimensional workloads of range queries. It follows that $F_n$ has the worst (up to a constant factor) tradeoff between redundancy and access overhead.

**Theorem 2** *For every block size $B$, any set of $n$ points on the 2-dimensional plane has an indexing scheme with average redundancy $r = 4\log(n/B)$ and access overhead $a = 4$.*

**Proof.** We will exhibit an indexing scheme of at most $4\log(n/B)$ rectangular blocks that has redundancy 4. For each point $p$ of the workload, consider the blocks for which $p$ is the lowest, leftmost point. The indexing scheme contains $\log(n/B)$ of these blocks of width $B, 2B, \ldots, 2^k B, \ldots, n$ (some of these blocks may not be full if there are not enough points). In general, the indexing scheme contains $4\log(n/B)$ blocks for which $p$ is the lowest/highest, leftmost/rightmost point and have width $B, 2B, \ldots, 2^k B, \ldots, n$. In total, the indexing scheme has at most $4n\log(n/B)$ blocks and average redundancy $r = 4\log(n/B)$.

We will show how to cover a query $Q$ of size $B$ with 4 blocks (larger queries can be broken into rectangles of size $B$ each and covered in a similar manner). Let $p_l$ and $p_r$ be the leftmost and rightmost points of $Q$ and let $x$ denote the width of $Q$. Let $w = 2^k B$ be such that $x/2 \leq w < x$. It is not hard to verify that $Q$ is covered by 4 blocks $B_{ll}, B_{hl}, B_{lr}, B_{hr}$ of width $w$ such that $p_l$ is the lowest (resp. highest), leftmost point of $B_{ll}$ (resp. $B_{hl}$) and $p_r$ is the lowest (resp. highest), rightmost point of $B_{lr}$ (resp. $B_{hr}$) (see Figure 4). $\square$

Theorems 1 and 2 characterize the tradeoff between redundancy and access overhead in the worst case (Fibonacci workload). But it is not clear how this applies to point sets of particular applications. Probably, the workloads of practical problems may have a better tradeoff, but very little is known about the properties of "practical" point sets. A first approach is to determine the tradeoff of random (uniformly distributed) point sets. The next theorem shows that almost all 2-dimensional sets of $n$ points require maximum redundancy at least $\Omega(\mathrm{loglog}\ \mathrm{n})$ in order to achieve constant access overhead. We do this by showing that a random (uniform) 2-dimensional point set has bad tradeoff.

**Theorem 3** *For every $\epsilon > 0$, there is a positive constant $c'$ such that a set of $n$ random (uniformly distributed) points on the plane has no indexing scheme with maximum redundancy $r \leq c'\mathrm{loglog}\ \mathrm{n}$ and access overhead $a < B$, with probability at least $1 - \epsilon$.*

**Proof.** The idea is very simple: Every large set of random points contains a small subset that looks like a Fibonacci lattice. More precisely, a random permutation of $k$ points is the Fibonacci permutation with probability $1/k!$. Consider now a set of $n$ random points. We can partition it into $n/k$ stripes of $k$ points. The probability that a stripe looks like a Fibonacci lattice is $1/k!$ and therefore the probability that no stripe is like a Fibonacci lattice is at most $(1 - 1/k!)^{n/k}$. It follows that if we choose $k$ such that $(1 - 1/k!)^{n/k} = \epsilon$ then the probability that no stripe looks like a Fibonacci lattice of $k$ points is at most $\epsilon$. Therefore, if the maximum redundancy $r$ is at most $c \log k$, where $c$ is the constant of Theorem 1, there is a query in some stripe that has access overhead $a = B$. The equality $(1 - 1/k!)^{n/k} = \epsilon$ is satisfied by some $k = \Theta(\log n/\mathrm{loglog}\ \mathrm{n})$; for such $k$, we have $\log k = \mathrm{loglog}\ \mathrm{n} + \mathrm{O}(1)$. In conclusion, there is a constant $c' \approx c$ such that any indexing scheme with maximum redundancy $r \leq c'\mathrm{loglog}\ \mathrm{n}$ has access overhead $a = B$, with probability $\epsilon$. $\square$

The lower bound of Theorem 3 is very weak. Although we can slightly strengthen it, it is significantly lower than the upper bound provided by Theorem 2; it also addresses only maximum redundancy but not average redundancy. We

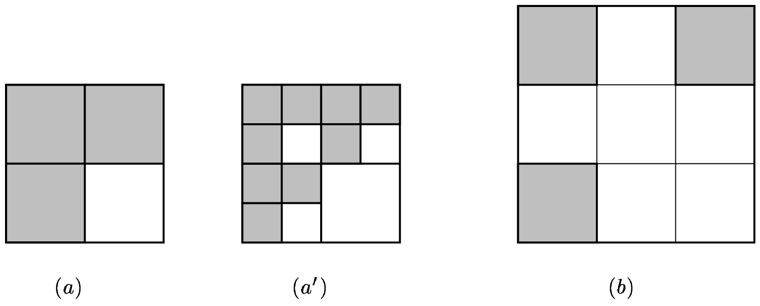

Figure 5: Self-similar point sets (fractals).

leave it as an important open problem to determine the exact tradeoff for sets of random points.

## 4 Indexing of fractals

As it was mentioned above, very little is known about "practical" point sets. An interesting suggestion in [3] is that many practical point sets are self-similar and their fractal dimension seems to be an important parameter. It is natural to ask whether there is any relation between fractal dimension and hardness of indexing. For example, it seems plausible that a point set with low fractal dimension will require small redundancy. This is suggested by some results in [1, 3]. Unfortunately, this is not the case. In fact,

> *there is no relation between the fractal dimension of a point set and its indexability.*

The reason is that for each set $C$ of the $d$-dimensional Euclidean space of fractal dimension $D$, there is an equivalent point set $C'$ that has fractal dimension $D' \approx 0$. To see this, notice that if we *thin out* (take a whole horizontal stripe and stretch it) a set of points, we get an equivalent set of points with respect to range queries. However, the new point set may have different fractal dimension, because the fractal dimension depends on the underlying Euclidean metric. Given a set of points $C$, we can thin it out so that the resulting set of points has dimension as close to 0 as we want. Figure 5 illustrates the point by considering fractals that are produced as follows: Take a square, partition it into $k \times k$ subsquares, throw away some of the subsquares, and repeat the process on the remaining $m$ subsquares ad infinitum (see Figure 5.$a, a'$). In the limit, the remaining points have fractal dimension $\log m / \log k$ [13]. Figure 5.$a, a'$ depicts a fractal of dimension $\log 3/\log 2 \approx 1.58$ and Figure 5.$b$ depicts a fractal of dimension $\log 3/\log 3 = 1$. However, it is obvious that the two point sets are equivalent with respect to range queries. By the way, the latter fractal also illustrates the fact that the fractal dimension is unrelated to indexing since it has dimension 1 but it is not as easy to index it as a straight line (also of dimension 1).

In fact, we claim that no kind of "dimension" can capture the indexability properties of a point set. The reason is that any reasonable definition of dimension must have the property that the dimension of a point set is invariant under rotations. However, our main result (Theorem 1) shows that the indexability of a point set can change dramatically if we rotate it: The redundancy/access overhead tradeoff can vary from constant (e.g., the regular grid studied in [6]) to logarithmic in the size of the point set (e.g., the Fibonacci workload).

In conclusion, if indeed practical workloads have good indexing properties, it is not because they have a small fractal dimension. It would be interesting to identify properties — necessary and sufficient conditions— of practical workloads that allow good indexing tradeoff. One necessary property is suggested by Theorem 1: Some pairs or points (and in general constant-size sets of points) must be contained in many queries of size $B$.

## 5 High-Dimensional Trade-offs

Finally, we state an extension of the main result of [6] to many dimensions. Independently, [17] came up with more comprehensive results of this sort. The proof is similar to that in [6] and is omitted from this abstract.

**Theorem 4** *For the $d$-dimensional workload with point set $\{1, \ldots, n^{1/d}\}^d$, the access overhead $a$ and the redundancy $r$ must satisfy $r \geq c_d \frac{log^{d-1} B}{2a^d \log(2a^d)}$, for some constant $c_d$.*

Extending also the upper bound of [6], we can show that there is an indexing scheme with access overhead $a$ and redundancy $r = (2d \frac{\log B}{\log a})^{d-1}$.

## 6 Conclusion and open problems

We provided tight bounds for the worst-case indexability of 2-dimensional workloads. Our results apply to higher dimensional workloads, but it is still an open problem to determine the exact worst-case redundancy/access overhead tradeoff for $d$-dimensional range queries. We conjecture that redundancy $\Theta(\log^{d-1} n)$ is required to achieve access overhead less than $B$. Unfortunately, there is no simple generalization of the Fibonacci lattice in higher dimensions. Furthermore, no simple point set is known to have good discrepancy properties in higher dimensions (see however [11]).

We also leave it as a major open problem to determine the tradeoff of random point sets. We have made some progress on this problem, but our results are too preliminary to be included in this abstract. Finally, there is some room of improvement on the upper bound of Theorem 2 to show that logarithmic *maximum* (as opposed to average) redundancy is sufficient to achieve constant access overhead for all 2-dimensional workloads.

## References

[1] A. Belussi and C. Faloutsos. Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension. In *Proc. 21st International Conference on Very Large Data Bases*, pp:299–310, Zurich, September 1995.

[2] B. Chor, C. E. Leiserson, R. L. Rivest, and J. B. Shearer. An Application of Number Theory to the Organization of Raster-Graphics Memory. *Journal of the ACM* 33(1):86–104, January 1986.

[3] C. Faloutsos and I. Kamel. Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension. In *Proc. 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp:4–13, Minneapolis, May 1994.

[4] A. Fiat and A. Shamir. How to Find a Battleship. Networks 19:361–371, 1989.

[5] A. Guttman. R-Trees: A Dynamic Index Structure For Spatial Searching. In *Proc. ACM-SIGMOD International Conference on Management of Data*, pp:47–57, Boston, June 1984.

[6] J. M. Hellerstein, E. Koutsoupias, and C. H. Papadimitriou. On the analysis of indexing schemes. *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Tucson, Arizona, 12–15 May 1997.

[7] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. Generalized Search Trees for Database Systems. In *Proc. 21st International Conference on Very Large Data Bases*, Zurich, September 1995.

[8] G. H. Hardy and E. M. Wright. *An introduction to the Theory of Numbers.* Third ed., Oxford University Press, 1956.

[9] P. C. Kanellakis, S. Ramaswamy, D. E. Vengroff, and J. S. Vitter. Indexing for Data Models with Constraints and Classes. In *Proc. 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp:233–243, Washington, D.C., May 1993. (Recent version available from the www.)

[10] D. E. Knuth. *The Art of Computer Programming; Volume III: Searching and Sorting.* Addison Wesley, 1973.

[11] N. Linial, M. Luby, M. Saks, and D. Zuckerman. Efficient Construction of a Small Hitting Set for Combinatorial Rectangles in High Dimension. *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, pp:258–267, San Diego, California, May 1993.

[12] D. B. Lomet and B. Salzberg. The hB-Tree: A Multiattribute Indexing Method. *ACM Transactions on Database Systems*, 15(4), December 1990.

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

[14] S. Ramaswamy and P. C. Kanellakis. OODB Indexing by Class Division. In *Proc. 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp:233–243, 1993.

[15] S. Ramaswamy and S. Subramanian. Path Caching: A Technique for Optimal External Searching. In *Proc. 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Minneapolis, 1994.

[16] J. T. Robinson. The k-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes. In *Proc. ACM-SIGMOD International Conference on Management of Data*, pp:10–18, Ann Arbor, April/May 1981.

[17] V. Samoladas and D. P. Miranker. A Lower Bound Theorem for Indexing Schemes and its Application to Multidimensional Range Queries. This conference.

[18] S. Subramanian and S. Ramaswamy. The $p$-range Tree: A Data Structure for Range Searching in Secondary Memory. *Proc. 6th SODA*, 1995.

[19] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: A Dynamic Index For Multi-Dimensional Objects. In *Proc. 13th International Conference on Very Large Data Bases*, pp:507–518, Brighton, September 1987.

[20] V. Turán Sós. *Acta Math. Acad. Sci. Hung.* 8:461–571, 1957.

[21] D. E. Vengroff and J. S. Vitter. Efficient 3-d Searching in External Memory. *Proc. 28th STOC*, pp:191–201, 1996.