A Lower Bound Theorem for Indexing Schemes and its Application to Multidimensional Range Queries *

Vasilis Samoladas
The University of Texas at Austin
vsam@cs.utexas.edu

Daniel P. Miranker
The University of Texas at Austin
miranker@cs.utexas.edu

Abstract

Indexing schemes were proposed by Hellerstein, Koutsoupias and Papadimitriou [7] to model data indexing on external memory. Using indexing schemes, the complexity of indexing is quantified by two parameters: storage redundancy and access overhead. There is a tradeoff between these two parameters, in the sense that for some problems it is not possible for both of these to be low.

In this paper we derive a lower-bounds theorem for arbitrary indexing schemes. We apply our theorem to the particular problem of d-dimensional range queries. We first resolve the open problem of [7] for a tight lower bound for 2-dimensional range queries and extend our lower bound to d-dimensional range queries. We then show, how, the construction in our lower-bounds proof may be exploited to derive indexing schemes for d-dimensional range queries, whose asymptotic complexity matches our lower bounds.

1 Introduction

In the last decade, the relational database paradigm has been extended in numerous ways. Here we are concerned with the introduction of new data models and query languages and the implications on indexing methods. We selectively mention geographical information systems, abstract data types and object data models, constraint databases, temporal databases, and on-line analytical processing. In these new contexts, the typical indexing methods of relational databases, B-trees and hashing, are generally considered inadequate [18]. Thus, there is a renewed need to develop a deeper understanding of data structures and algorithms to speed operations on external memory.

Database extensibility necessarily includes extensible development of index mechanisms. Engineering approaches comprising parameterized components and/or libraries of composable components are demonstrating successes [8, 3]. In related work, at least one effort is underway to generate such a system from formal specifications [2]. This effort, though it started in the database domain, has focused on main-memory data structures and admits, relative to databases, simple cost models. The database problem is more challenging, because it involves a tradeoff between two antagonistic aspects; storage redundancy, and access overhead.

In this paper, we adopt the model of *indexing schemes*, proposed in [7]. Indexing schemes are *not* data structures, as they do not model the problem of searching for the data. They only model the (spatial) locality issues that arise. It is assumed that for any given query we can determine, without additional I/O, the particular disk pages that we have to access in order to answer it. Keeping this abstraction in mind, consider the problem of indexing N elements on a hard disk, where each disk page can hold B elements. Ideally, we would like our index to consume $\frac{N}{B}$ disk pages, and be able to answer any query Q by reading only $\lceil \frac{|Q|}{B} \rceil$ disk pages where $|Q|$ is the number of objects returned by the query. In reality, for some types of queries this is not possible. Assume that some particular index occupies $r\frac{N}{B}$ disk blocks, and can answer queries by reading at most $A\lceil \frac{|Q|}{B} \rceil$ disk pages. We identify r as the storage redundancy, and A as the access overhead for this index.

Our main theorem, the Redundancy Theorem, provides lower bounds for the storage redundancy r , given a desired access overhead A and the block size B, *for any indexing problem*. Our approach is combinatorial in nature. The results are reminiscent of the AT^2-complexity results in VLSI theory [11]. The main argument in AT^2-complexity, consists of two parts: a *geometric separation* theorem, and a concept of *communication complexity*, or "information content" of a digital circuit, modeled as a boolean function. In the domain of indexing schemes, geometric separation corresponds to properties of disk pages, viewed as sets of fixed size. Communication complexity characterizes the particular type of indexing *workload* we study (multi-dimensional range queries, set inclusion queries etc.).

Our results identify an analog of a geometric separation theorem. To date we have only partially succeeded in characterizing workloads in a general manner, with respect to their intrinsic complexity. Thus, our Redundancy Theorem will sometimes only yield trivial lower bounds. However, it allows us to derive a number of interesting results. First, we use it to prove the conjecture of Hellerstein, Koutsoupias and Papadimitriou for a worst-case trade-off of $r = \Omega(\log B / \log A)$ for 2-dimensional range queries. Then, we extend this result to the d-dimensional case, where we

*This research is partially funded by DARPA, contract number: F30602-96-2-0226 and the Applied Research Laboratories, University of Texas, Internal Research and Development Program.

PODS 98 Seattle WA USA
Copyright ACM 1998 0-89791-996-3 98 6 ,$5.00

show a lower bound of $r = \Omega\left(\left(\frac{\log B}{\log A}\right)^{d-1}\right)$. Also, we demonstrate that our technique for deriving lower bounds can often be "reversed" to derive optimal indexing schemes, and for the case of d-dimensional range queries we show $r = O\left(\left(\frac{\log B}{\log A}\right)^{d-1}\right)$. Finally, we discuss some open issues, and relate our results to recent developments in the area of indexing.

2 Related Work

Our work continues on the work of Hellerstein, Koutsoupias and Papadimitriou, presented in [7]. That work was in turn motivated by the work of Hellerstein, Naughton and Pfeffer on the Generalized Search Tree [8] (also known as GiST). GiST is an extensible indexing structure, organized as a balanced search tree. In their discussion of indexing issues, the authors stated the need for a "theory of indexability", a formal framework that would "... describe whether or not trying to index a given data set is practical for a given set of queries." Previous work on indexing data structures concentrated on the study of specialized problems. The first problem-independent insight on external data structures was offered in [7]. Continuing on this work, Koutsoupias and Taylor [10] investigate the indexability of 2-dimensional data sets, and derive asymptotically tight indexing schemes for these sets. In particular, they identify the Fibonacci workload as a worst-case workload for 2-dimensional indexing.

The research into external data structures has largely been experimental. Theoretical work on the B-tree and its variants, as well as on external hashing, concentrated mainly on probabilistic analysis of performance, under various distributions of the indexed data. For these problems, the worst-case asymptotic performance has been known for a long time.

In the area of multidimensional indexing, data structures are classified into two categories: those that partition the data set, such as R-trees and their variants, and those that partition the search space. In both categories, most of the proposed algorithms are based on heuristics, and have relatively bad worst-case asymptotic performance. See [6] for an overview.

This situation has been changing in the past few years, mostly due to the work of Kanellakis, and that of Vitter, and their collaborators. We attribute the renewed interest in the fundamental results of [9], where it was shown that indexing in new database paradigms, such as constraint databases, and databases with class hierarchies, can be reduced to special cases of multidimensional range searching. In subsequent publications, [14, 13, 16, 17] there are presented asymptotically efficient dynamic algorithms for 2-sided, 3-sided and interval management queries. An optimal solution for the interval management problem has recently been found [1]. We also mention the work of [12], who use cost metrics similar to ours, to characterize the locality in external graph searching.

Finally, we should mention the recent results of Faloutsos and Kamel [5], on characterizing the performance of multidimensional range queries on R-trees, using the fractal dimension of real-world datasets. Their work provides only intuitive and experimental evidence, in contrast to our theoretical approach, but we feel that there is a strong correlation between their empirical observations and our theoretical models, and we will discuss this issue further in this paper.

3 Definitions and Notation

We now define indexing schemes, departing slightly from the notation of [7], and present some additional notation of our own.

3.1 Indexing Schemes

Definition 3.1 *A workload W is a tuple $W = (I, \mathcal{Q})$, where I is a non-empty finite set, and $\mathcal{Q}$ is a set of subsets of I.*

For a workload $W = (I, \mathcal{Q})$, the elements of I are called *objects*, and I is the *object set*. Also, the elements of $\mathcal{Q}$ are called *queries*, and $\mathcal{Q}$ is the *query set*. We also define N to stand for $|I|$, the cardinality of I, and q to stand for $|\mathcal{Q}|$.

In the terminology of combinatorics, W is a simple hypergraph, where I is the vertex set, and $\mathcal{Q}$ is the edge set. We choose not to use this terminology, but instead we adopt terminology that is more natural for databases.

Definition 3.2 *An indexing scheme $\mathcal{S}$ for block size B, B an integer greater than 1, is a pair $\mathcal{S} = (W, \mathcal{B})$, where $W = (I, \mathcal{Q})$ is a workload, and $\mathcal{B}$ is a set of B-subsets of I, such that $\mathcal{B}$ covers I.*

We refer to the elements of $\mathcal{B}$ as blocks, and to $\mathcal{B}$ as the set of blocks. We refer to B as the block size, and K stands for $|\mathcal{B}|$. Again, notice that an indexing scheme is a simple, B-regular hypergraph with vertex set I.

We use some standard notation throughout this paper. We will use lower-case letters from the end of the alphabet, x, y, z to represent objects, letter Q, possibly with subscripts, to denote queries, and letter b, possibly with subscripts, to denote blocks. Also, we typically use U to represent sets of blocks.

3.2 Performance measures

We now define the two performance measures that we use, departing slightly from the notation of [7]. In the following definitions, let $\mathcal{S} = (W, \mathcal{B})$ be an indexing scheme of block size B on workload $W = (I, \mathcal{Q})$.

3.2.1 Redundancy

Definition 3.3 *The redundancy $r(x)$ of object x is defined as the number of blocks that contain x:*

$$r(x) = |\{b \in \mathcal{B} : x \in b\}|$$

The redundancy r of $\mathcal{S}$ is then defined as the average of $r(x)$ over all objects:

$$r = \frac{1}{N} \sum_{x \in I} r(x)$$

It is easy to see that $K = \frac{rN}{B}$. We also define the maximum redundancy $\hat{r}$ in $\mathcal{S}$, as $\hat{r} = \max_{x \in I} r(x)$

3.2.2 Access Overhead

Definition 3.4 *A set of blocks U covers a query Q, iff $Q \subseteq \bigcup U$.*

Definition 3.5 *A cover set C_Q for query Q is a minimum-size set of blocks that covers Q.*

Notice that a query may have multiple cover sets.

Definition 3.6 *The access overhead $A(Q)$ of query Q is defined as*

$$A(Q) = \frac{|C_Q|}{\lceil \frac{|Q|}{B} \rceil}$$

where C_Q is a cover set for Q.

Notice that $1 \leq A(Q) \leq B$.

Informally, $A(Q)$ models the normalized cost of the query Q, in terms of block accesses. For a given query Q, $\lceil |Q|/B \rceil$ is the minimum number of blocks required. $A(Q)$ is the multiplicative overhead associated with Q for a particular indexing scheme.

We now define the access overhead A of indexing scheme S, to be the maximum of $A(Q)$ over all queries.

Definition 3.7 *The access overhead A for indexing scheme S is*

$$A = \max_{Q \in \mathcal{Q}} A(Q)$$

3.3 Some Trivial Bounds and Trade-offs

We assume that the number of objects N is always much greater than the block size B, although B can grow arbitrarily large.

For some indexing scheme S, the minimum possible redundancy is 1, when $\mathcal{B}$ is a partition of I, and the maximum is $\frac{B}{N}\binom{N}{B}$, when $\mathcal{B} = \binom{I}{B}$. For S having maximum redundancy, A is exactly 1, which is minimum. Also, for $r = 1$ it is easy to devise a problem where $A = B$, which is maximum (e.g. $\mathcal{Q} = \binom{I}{B}$).

We will not comment further on the proposed framework, since it is thoroughly discussed in [7].

4 The Redundancy Theorem

We now turn our attention to an analysis of the above model, that will lead us to the Redundancy Theorem. We first state and prove a set-theoretic result that is of central importance to our work. Note that this theorem is not specific to indexing schemes, but is really a theorem in extremal set theory. The reader is warned that the notation does not correspond to indexing schemes.

Theorem 4.1 *Let $S_1, S_2, \ldots, S_a$ ($a \geq 1$) be non-empty finite sets, $Q = S_1 \cup S_2 \cup \ldots \cup S_a$ be their union, and $L \leq |Q|$ be a positive integer. Let k denote the maximum integer such that there exist k pair-wise disjoint sets $P_1, P_2, \ldots, P_k$, so that for all i, $1 \leq i \leq k$,*

1. *$|P_i| = L$, and*
2. *$P_i \subseteq S_j$ for some j, $1 \leq j \leq a$.*

or $k = 0$ if no such sets exist. Then,

$$k \geq \left\lceil \frac{|Q| - a(L-1)}{L} \right\rceil \qquad (1)$$

Proof: We proceed by induction on a. For $a = 1$, the proof is trivial. Assume that the theorem is true for some a. Let $S_1, S_2, \ldots, S_a, S_{a+1}$ be non-empty, finite sets. Let Q denote their union, and Q' denote the union of the first a of them, $Q' = S_1 \cup S_2 \cup \ldots \cup S_a$. Finally, let k be defined as in the theorem, for all $a+1$ sets.

We apply the induction hypothesis on $S_1, S_2, \ldots, S_a$. Let $k' = \left\lceil \frac{|Q'| - a(L-1)}{L} \right\rceil$. If $k' > 0$, there exist (at least) k' disjoint sets $P_1, \ldots, P_{k'}$, with $|P_i| = L$, each contained entirely in some set S_j, for $j \leq a$. Let P stand for their union, $P = P_1 \cup \ldots \cup P_{k'}$. If $k' = 0$, let $P = \emptyset$.

Now, take $U = S_{a+1} \cap P$, and $V = S_{a+1} - U$. Write $|V| = Ln + l$, for $n \geq 0$ and $0 \leq l < L$. We conclude that

$$k \geq k' + n$$

which implies

$$Lk \geq Lk' + Ln$$

Replacing Lk', we have

$$Lk \geq |Q'| - a(L-1) + Ln$$

But $|Q| = |Q'| + |S_{a+1}| - |Q' \cap S_{a+1}|$. Replacing, we get

$$Lk \geq |Q| - |S_{a+1}| + |Q' \cap S_{a+1}| - a(L-1) + Ln$$

Replacing $|S_{a+1}|$ by $|U| + |V|$,

$$Lk \geq |Q| - |U| - |V| + |Q' \cap S_{a+1}| - a(L-1) + Ln$$

Replacing Ln by $|V| - l$ we get

$$\begin{aligned} Lk &\geq |Q| - |U| - |V| + |Q' \cap S_{a+1}| - a(L-1) \\ &\quad + |V| - l \\ &= |Q| - |U| + |Q' \cap S_{a+1}| - a(L-1) - l \end{aligned}$$

Finally, note that $|U| \leq |Q' \cap S_{a+1}|$, which gives us

$$Lk \geq |Q| - a(L-1) - l \geq |Q| - (a+1)(L-1)$$

from which the theorem follows.□

Although not necessarily obvious from the above inductive proof, there is a simple way of constructing sets $P_1, \ldots, P_k$ given sets $S_1, \ldots, S_a$. We proceed in a steps, processing set S_j in step j. In step j, we use elements of S_j to create as many disjoint sets P_i as possible, taking care not to use again any elements of S_j used by previous steps. At every step, we "ignore" at most $L-1$ elements of S_j, where "ignore" means that we do not use these elements to construct P_i sets. At the end, we will have "ignored" a total of at most $a(L-1)$ elements of Q, thus the result of the theorem.

The inequality in the theorem is tight. For example, the equality applies in the case where sets S_j are disjoint, and $|S_j| \bmod L = L - 1$

We now apply the above theorem to the domain of indexing schemes. First we define a new concept, *flakes*.

Definition 4.2 *Let $S = (W, \mathcal{B})$ be an indexing scheme on workload $W = (I, \mathcal{Q})$. A flake is any set of objects $F \subseteq I$ such that for some query Q and some block b, $F \subseteq Q \cap b$.*

We now have the following lemma on flakes:

Lemma 4.1 (Flaking Lemma) *Let S be an indexing scheme, A be the access overhead, and $2 < \varepsilon < \frac{B}{A}$ be a real number. Then, any query Q with $|Q| \geq B$ will contain at least $(\varepsilon - 2)A\frac{|Q|}{B}$ pair-wise disjoint flakes of size $\lfloor \frac{B}{\varepsilon A} \rfloor$.*

Proof: Choose a cover set for Q, say $C_Q = \{b_1, \ldots, b_a\}$, of size a. Let $S_1, \ldots, S_a$ be defined as $S_i = Q \cap b_i$ for $1 \leq i \leq a$.

We have $a = A(Q)\lceil\frac{|Q|}{B}\rceil \leq A\lceil\frac{|Q|}{B}\rceil$. From Theorem 4.1 we know that the number k of flakes of size $\lfloor\frac{B}{\epsilon A}\rfloor$ is at least

$$\begin{aligned}
k &\geq \left\lceil \frac{|Q| - a(\lfloor\frac{B}{\epsilon A}\rfloor - 1)}{\lfloor\frac{B}{\epsilon A}\rfloor} \right\rceil \\
&\geq \frac{|Q| - a(\lfloor\frac{B}{\epsilon A}\rfloor - 1)}{\lfloor\frac{B}{\epsilon A}\rfloor} \\
&\geq \frac{|Q| - A\lceil\frac{|Q|}{B}\rceil(\lfloor\frac{B}{\epsilon A}\rfloor - 1)}{\lfloor\frac{B}{\epsilon A}\rfloor} \\
&\geq \frac{|Q| - A(\frac{|Q|}{B}+1)(\lfloor\frac{B}{\epsilon A}\rfloor - 1)}{\lfloor\frac{B}{\epsilon A}\rfloor} \\
&\geq \frac{|Q| - A(\frac{|Q|}{B}+1)(\frac{B}{\epsilon A} - 1)}{\frac{B}{\epsilon A}} \\
&\geq \frac{\epsilon A|Q|}{B} - A\left(\frac{|Q|}{B}+1\right) \\
&\geq \frac{\epsilon A|Q|}{B} - 2A\frac{|Q|}{B} \\
&= A(\epsilon-2)\frac{|Q|}{B}
\end{aligned}$$

The last inequality follows from $|Q| \geq B$. □

Before we proceed to prove our main theorem, we need a technical tool from extremal set theory, known as Johnson's lemma:

Lemma 4.2 (Johnson's lemma) *Let A be a finite set, and $S_1, S_2, \ldots, S_k$ be subsets of A, each of size at least $\alpha|A|$, such that the intersection of any two of them is of size at most $\beta|A|$. If $\beta < \frac{a^2}{2-a}$, the number of subsets k is at most α/β.*

Proof: See [7] for a proof. □

We are now ready to state and prove our main theorem.

Theorem 4.3 (Redundancy Theorem) *Let S be an indexing scheme, and $Q_1, Q_2, \ldots, Q_M$ be queries, such that for every i, $1 \leq i \leq M$:*

1. $|Q_i| \geq B$, *and*
2. $|Q_i \cap Q_j| \leq \frac{B}{2(\epsilon A)^2}$ *for all* $j \neq i$, $1 \leq j \leq M$.

Then, the redundancy is bound by

$$r \geq \frac{\epsilon-2}{2\epsilon}\frac{1}{N}\sum_{i=1}^{M}|Q_i|$$

where $2 < \epsilon < \frac{B}{A}$ is any real number such that $\frac{B}{\epsilon A}$ is integer.

Proof: We begin by discussing the role of parameter ϵ. This parameter exists in the analysis for assuring that $\frac{B}{\epsilon A}$ is integer. For the reader's convenience, it may be assumed that ϵ lies between 3 and 6 (provided that $\frac{B}{A} > 6$). There is no technical importance to parameter ϵ, and our analysis would be better without it, but we were unable to remove it without seriously complicating the rest of our proof.

We will prove the lower bound in two steps. First, we compute the *minimum* number of flakes associated with queries $Q_1, Q_2, \ldots Q_M$. Let this number be f_1. Then we will compute the maximum number of flakes associated with each block. Let this number be f_2. Clearly, there will be at least f_1/f_2 blocks in B.

Step 1 Consider any query Q_i. By the partitioning lemma, this query is associated with at least $(\epsilon-2)A\frac{|Q_i|}{B}$ *distinct* flakes of size $\frac{B}{\epsilon A}$. Let F be such a flake. F cannot be associated with some other query Q_j, $j \neq i$, because if it were, then it would be a subset of Q_j as well as of Q_i, and thus $|Q_i \cap Q_j| \geq \frac{B}{\epsilon A} > \frac{B}{2(\epsilon A)^2}$. We conclude that

$$f_1 = \sum_{i=1}^{M}(\epsilon-2)A\frac{|Q_i|}{B} = \frac{(\epsilon-2)A}{B}\sum_{i=1}^{M}|Q_i|$$

Step 2: Consider any block b, and let $F_1, F_2, \ldots, F_k$ be the flakes associated with this block. Since all these flakes are subsets of b, we upper bound the number of flakes k, using Johnson's lemma. Each flake F_i is of size $\frac{1}{\epsilon A}B$. Also, for two distinct flakes F_i and F_j, $i \neq j$, $|F_i \cap F_j| \leq \frac{1}{2(\epsilon A)^2}B$, by the following argument: If the flakes are associated with the same query, then they are disjoint. If the flakes are associated with different queries, then their intersection is bounded by the intersection of these queries. Thus, Johnson's lemma is applicable with $\alpha = \frac{1}{\epsilon A}$, and $\beta = \frac{1}{2(\epsilon A)^2}$. It can easily be checked that $\beta < a^2/(2-a)$. Thus, we conclude that

$$f_2 = \frac{\alpha}{\beta} = 2\epsilon A$$

The proof is complete, by the inequality $K = \frac{rN}{B} \geq \frac{f_1}{f_2}$ which simplifies to

$$r \geq \frac{\epsilon-2}{2\epsilon}\frac{1}{N}\sum_{i=1}^{M}|Q_i|$$

□.

5 Lower bounds for d-dimensional range queries

In this section we apply the Redundancy Theorem to d-dimensional range queries. First, we examine the case for 2-dimensional range queries, and then we generalize to d dimensions.

For any $d \geq 1$, we define the d-dimensional range query workload, $\mathcal{R}_n^d$, whose object set is $I = [1:n]^d$ and whose query set is

$$\mathcal{Q} = \{[a_1 : b_1] \times \ldots \times [a_d : b_d] | 1 \leq a_i \leq b_i \leq n\}$$

For this workload, $N = n^d$.

5.1 2-d range queries

In order to apply the Redundancy Theorem, we must identify queries $Q_1, Q_2, \ldots, Q_M$, each of size at least B, and with pairwise intersections at most $B/2(\epsilon A)^2$. We consider only queries of size $c^j \times \frac{B}{c^j}$, for $j = 0, 1, \ldots, \log_c B$. For each aspect ratio, we will partition the $n \times n$ space, obtaining a total of $M = \frac{n^2}{B}(1 + \log_c B)$ queries of size B each. Before we apply the theorem, we compute parameter c.

Let j and j' be integers $0 \leq j < j' \leq \log_c B$, and Q_j and $Q_{j'}$ be queries of dimensions $c^j \times \frac{B}{c^j}$ and $c^{j'} \times \frac{B}{c^{j'}}$ respectively. Fig.1 depicts the setup. It is easy to see that for any j and j', $|Q_j \cap Q_{j'}| \leq \frac{B}{c^{j'-j}} \leq \frac{B}{c}$. Thus, we take $c = 2(\epsilon A)^2$.

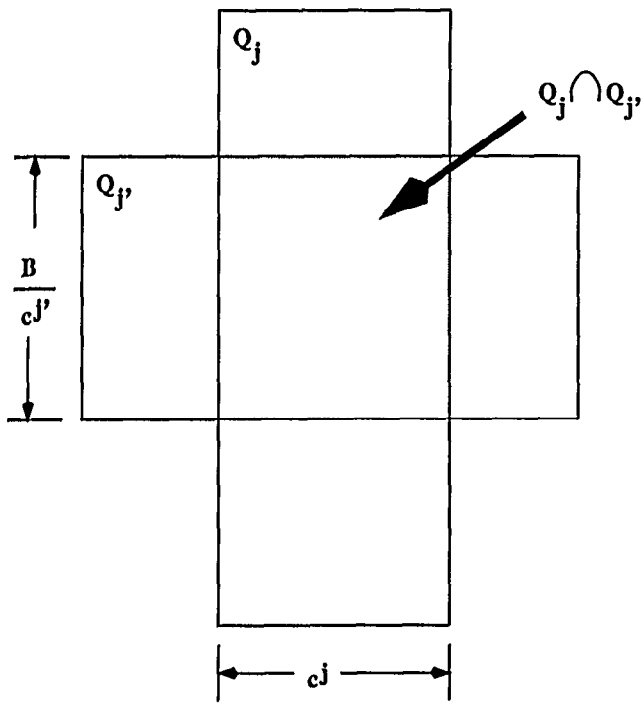

Figure 1: Two rectangles of sizes $c^j \times \frac{B}{c^j}$ and $c^{j'} \times \frac{B}{c^{j'}}$, $j < j'$, intersecting at a maximum number of points $\frac{B}{c^{j'-j}}$.

We are now ready to apply the Redundancy Theorem. From the theorem,

$$\begin{aligned} r &\geq \frac{\varepsilon-2}{2\varepsilon}\frac{MB}{n^2} \\ &= \frac{\varepsilon-2}{2\varepsilon}\frac{1}{n^2}\left(B\frac{n^2}{B}(1+\log_c B)\right) \\ &= \frac{\varepsilon-2}{2\varepsilon}(1+\log_c B) \\ &\geq \frac{\varepsilon-2}{2\varepsilon}\log_c B \\ &= \frac{\varepsilon-2}{2\varepsilon}\frac{\log B}{\log(2\varepsilon^2A^2)} \end{aligned}$$

and thus we have the $\Omega(\frac{\log B}{\log A})$ result conjectured by [7].

5.2 d-dimensional queries

We can generalize the above technique to d-dimensional queries. We consider queries of size B, with dimensions $c^{j_1} \times c^{j_2} \times \ldots \times c^{j_d}$, for all nonnegative integer $j_1, j_2, \ldots, j_d$, such that $\sum_{k=1}^{d} j_k = \log_c B$. For each sequence $j_1, j_2, \ldots, j_d$, we partition the d-dimensional cube into n^d/B (hyper)rectangles, of dimensions $c^{j_1} \times c^{j_2} \times \ldots \times c^{j_d}$.

In order to select the appropriate value for c, we consider the size of pairwise intersections of rectangles with different dimensions. It is easy to see that $c = 2(\varepsilon A)^2$ is applicable in this case also, guaranteeing that the intersection of any two rectangles will have size at most $\frac{B}{2(\varepsilon A)^2}$.

We also use the well-known fact that the number of distinct sequences of d nonnegative integers, whose sum is n, is given by

$$\binom{n+d-1}{d-1}$$

(cf. Bose-Einstein distribution).

Thus, the total number of queries (each of size B) will be

$$M = \frac{n^d}{B}\binom{\frac{\log B}{\log 2(\varepsilon A)^2}+d-1}{d-1}$$

and for the redundancy we have

$$r \geq \frac{\varepsilon-2}{2\varepsilon}\binom{\frac{\log B}{\log 2(\varepsilon A)^2}+d-1}{d-1}$$

For d a constant, the above quantity is a polynomial of degree $d-1$. Thus, we have shown the following theorem:

Theorem 5.1 *For workload $\mathcal{R}_n^d$, the storage redundancy is bound by*

$$r = \binom{\Omega\left(\frac{\log B}{\log A}\right)+d-1}{d-1} = \Omega\left(\left(\frac{\log B}{\log A}\right)^{d-1}\right)$$

6 Deriving Indexing Schemes

We have seen that we can derive lower bounds for the storage redundancy by selecting as large a number as possible of queries with small pairwise intersections. The process of selecting these queries depends on the "topology" of the workload in question (the term topology is used in an intuitive manner).

It is only natural to consider the merit of using a similar process for selecting blocks (instead of queries), in order to construct indexing schemes for a desired access overhead A. In this section we shall apply this idea to $\mathcal{R}_n^d$, the d-dimensional range query workload. In the lower-bounds analysis of $\mathcal{R}_n^d$, we considered queries of size B. In this analysis, we choose these queries to serve as the blocks.

In order to facilitate our analysis, we provide the following simple fact:

Proposition 1 *Let $n \geq 1$, and consider the intervals $[0 : n-1], [n : 2n-1], [2n : 3n-1], \ldots$, partitioning the natural numbers. Then, interval $[a : b]$, $0 \leq a \leq b$, of size $s = b-a+1$, intersects at most $\lceil \frac{s-1}{n} \rceil + 1$ intervals.*

Proof: Left as an exercise to the reader.□

We now consider workloads $\mathcal{R}_n^d$, for some fixed d. For our analysis, we employ parameter c. For each sequence $j_1, j_2, \ldots, j_d$ of integers, such that $\sum_{k=1}^{d} j_k = \log_c B$, we partition the d-dimensional mesh into blocks of dimensions $c^{j_1} \times c^{j_2} \times \ldots \times c^{j_d}$. Thus, we have a redundancy:

$$r = \binom{\log_c B + d - 1}{d-1} = O\left(\log_c^{d-1} B\right) \qquad (2)$$

We now have to compute the access overhead for this selection of blocks, and parameter c. Consider any query of dimensions $X_1 \times X_2 \times \ldots \times X_d$. Let

$$V = \prod_{i=1}^{d} X_i$$

be the size of this query. Also, define a parameter λ as

$$\lambda = \left(\frac{V}{B}\right)^{1/d}$$

Finally, we define the scaled dimensions, $\hat{X}_i$, $1 \le i \le d$, as

$$\hat{X}_i = \frac{X_i}{\lambda}$$

Notice that $\prod_{i=1}^{d} \hat{X}_i = B$.

We will assume that a cover for our query consists only of blocks of fixed aspect, i.e. blocks of dimension $c^{j_1} \times c^{j_2} \times \ldots \times c^{j_d}$ for fixed $j_1, j_2, \ldots, j_d$. By applying Proposition 1, the number of blocks f will be at most

$$\begin{aligned} f &= \prod_{i=1}^{d} \left(\left\lceil \frac{X_i - 1}{c^{j_i}} \right\rceil + 1 \right) \\ &\le \prod_{i=1}^{d} \left(\frac{X_i}{c^{j_i}} + 2 \right) \\ &= \prod_{i=1}^{d} \left(\lambda \frac{\hat{X}_i}{c^{j_i}} + 2 \right) \end{aligned}$$

But because $\prod_{i=1}^{d} \hat{X}_i = B$, it is possible to choose j_i so that we have

$$\frac{\hat{X}_i}{c^{j_i}} \le c \text{ for all } i, \quad 1 \le i \le d$$

Thus, we have

$$f \le (\lambda c + 2)^d$$

In the case where $\lambda c \le 2$, we have $f \le 4^d$. Thus, we require that $A \ge 4^d$. Now, assume $\lambda c > 2$. In this case,

$$\begin{aligned} f &\le (2\lambda c)^d \\ &= (2c)^d \frac{V}{B} \\ &\le (2c)^d \left\lceil \frac{V}{B} \right\rceil \end{aligned}$$

From this we get that

$$A = (2c)^d \ge 4^d$$

which gives, for $c \ge 2$,

$$c = \frac{A^{1/d}}{2}$$

By replacing this expression into Eq.2 we get the desired result. Thus, we have shown the following theorem:

Theorem 6.1 *For the $\mathcal{R}_n^d$ workload, the redundancy of any optimal indexing scheme will be*

$$r = \binom{\Theta\left(\frac{\log B}{\log A}\right) + d - 1}{d-1} = \Theta\left(\left(\frac{\log B}{\log A}\right)^{d-1}\right)$$

for access overhead $A \ge 4^d$.

Interestingly, this theorem imposes an absolute lower bound on the access overhead $A \ge 4^d$. It is of course possible to achieve access overhead arbitrarily close, or equal, to 1, and for these small values of A the lower bound on r still holds. However, it is not clear if the upper bound on r is achievable for small A.

7 Discussion and Future Work

7.1 Multidimensional Range Queries

Lower-bound results on indexing schemes imply lower bounds for the corresponding indexing problems. In general, this is not true for upper-bound results, because in indexing schemes we ignore the search problem that indexing data structures have to solve.

For multidimensional range queries in particular, our results provide the first lower bounds for indexing structures, for dimensions higher than 2. In particular, we have the following:

Theorem 7.1 *Any external data structure that answers d-dimensional range queries in time $O\left(\left\lceil \frac{|Q|}{B} \right\rceil\right)$ will consume $\Omega\left(\frac{N}{B} \log^{d-1} B\right)$ space.*

We believe this is the first lower bound on storage redundancy, for $d > 2$. However, this lower bound is probably very weak.

For $d = 2$, there is a lower bound by Subramanian and Ramaswamy [16], of $\Omega\left(\frac{N}{B} \frac{\log N}{\log \log_B N}\right)$, for query I/O cost bounded by $O\left(\log_B^c N + \frac{|Q|}{B}\right)$. Although stronger than that of theorem 7.1, even this is not tight. Elsewhere in these proceedings, Koutsoupias and Taylor [10] show that logarithmic redundancy $r = \Omega(\log N)$ is needed, if the access overhead is to be $A < B$. Also, slightly higher redundancy $r = 4\log(N/B)$ is sufficient for $A \le 4$, for any 2-dimensional workload. Thus, the trade-off for 2-dimensional workloads exhibits threshold behaviour.

These results give rise to two fundamental open problems. First, none of these results reveals the trade-off between r and A, for given N and B. This is not a moot question, because the hidden constant in the $r = \Omega(\log N)$ lower bound is extremely small, small enough to matter for reasonable values of N, and also it is not independent of B. Our Redundancy Theorem may be applicable to this problem, deriving interesting refinements to these complexities. The second open problem is that of extending these results to higher dimensions. In [10] it is conjectured that redundancy $r = \Theta(\log^{d-1} N)$ is required for d-dimensional workloads.

Apart from the general problem of multidimensional range queries, our results on $\mathcal{R}_n^d$ are applicable in two other areas of recent interest: Multidimensional OLAP (MOLAP) [19] and external multidimensional arrays [15]. In MOLAP, the challenge is to materialize a datacube to disk, so that range queries can be answered efficiently. Usually, the datacube undergoes some compression, to improve space utilization. External multidimensional arrays are useful in scientific and engineering applications, when the (usually numeric) data does not fit into main memory. These types of data structures are strongly relevant to indexing schemes, because they *do not have a search component.* Especially for arrays, workload $\mathcal{R}_n^d$ is a relatively accurate model for a d-dimensional (dense) array. It is not accidental that the ideas in [15] are similar to ours, and their experimental results are consistent with our theoretical predictions.

7.2 Locality versus Search

It has been said earlier that indexing schemes do not model the search problem of data indexing, but only the locality aspects that arise. However, the two issues are not unrelated. Indeed, many indexing structures, such as a B-trees,

R-trees etc. are hierarchical. Consider such a hierarchical indexing structure. Each level in the hierarchical structure reflects the locality of the level under it. It seems quite likely that for such structures a recursive analysis may be carried out, where each recursive step of the analysis will employ locality arguments.

In hierarchical structures for multidimensional range queries, any particular level will employ *Minimum Bounding Rectangles (MBR)* to aggregate the data in the level under it. The Russian Doll tree (RD-tree)[8] is a hierarchical structure for the indexing of arbitrary sets of integers. In correspondence with MBR, RD-trees employ *rangesets*, to aggregate the data from level to level. In both cases, the common feature is a *combination procedure*: a set of MBRs combined, determine a minimal MBR subsuming them all. A set of rangesets can be combined to form a minimal rangeset subsuming them.

In our Redundancy Theorem, our basic combinatorial tool is flakes. Indeed, the Redundancy Theorem is a flake-counting theorem. It may be possible to refine the definition of flakes, and devise a *combination procedure* for (the refined) flakes. This approach may lead to results (and algorithms) on the performance of generalized hierarchical structures.

7.3 Uniformity and Independence

In a pioneering article, Faloutsos and Kamel [5] demonstrated experimentally that the performance of existing indexing structures (R-trees) for multidimensional queries can be accurately predicted from the fractal dimension of the indexed dataset. Their results strongly suggest the existence of a theoretical *average-case* result, with the fractal dimension replacing the embedding dimension in the costs. A natural question is to examine whether there exists in fact a stronger, *worst-case* result of a similar nature. This approach will further allow us to model multidimensional queries better that $\mathcal{R}_n^d$ or the fibonacci workload does, ameliorating the negative consequences of the results of [10].

On this subject, the authors of [10] state that a theoretical approach along these lines is not likely to succeed, and that in fact there is no relation between the fractal dimension of a point set and its indexability. Based on their own results, they show that:

1. Topological transformations that do not change the indexing properties of a set, such as stretching the embedding space selectively, can change the fractal dimension of this set dramatically, and

2. Topological transformations that do not change the fractal dimension of a set, such as rotation, change the indexing properties of the set quite dramatically.

Although their arguments are irrefutable, we do not share the conclusions of [10] that fractal dimension and indexability are unrelated. The arguments of [10] are based in two implicit assumptions, an assumption on the definition and use of fractal dimension, and an assumption on the definition of "indexability".

Agreeing on a definition of fractal dimension is crucial. In many occasions, implicit use of different definitions resulted in contradictory results and caused much confusion [4]. The definition used in the experiments of Faloutsos and Kamel is the box-counting fractal dimension, a standard approach for characterizing real, finite data sets. Although reasonable in an experimental setting, since it is easy to compute, it is probably not suitable for theoretical arguments. Indeed, the notion that by a single number we can characterize the combinatorial properties of multidimensional indexing is rather far-fetched. But, going in the opposite direction could be very fruitful: determine a set of combinatorial conditions on problems, under which fractal dimension is relevant to indexing. Then, evaluate these conditions experimentally, for real datasets. We conjecture that such conditions, if found, will be applicable to the majority of real datasets.

Our second point has to do with the definition of "indexability". The results in [10] suggest that the tradeoff between redundancy and overhead changes dramatically when a 2-dimensional data set is rotated. However, access overhead is a worst-case cost metric. Under an average-case metric, such as the *expected access overhead* $\bar{A}$, defined as

$$\bar{A} = \sum_{i=1}^{q} w_i A(Q_i)$$

for appropriate weights w_i, this may not be so. In fact, the experimental results of Faloutsos and Kamel can be seen as Monte Carlo estimations of $\bar{A}$ on real data sets. We argue that $\bar{A}$ is an equally interesting metric of "indexability", as A is. Also, $\bar{A}$ is less biased than A in favor of small queries, in the sense that A is basically determined by queries of size close to the block size B, at least for multidimensional workloads.

In conclusion, we believe that the arguments in [10], although thought-provoking, do not support such a general claim, that the fractal dimension is unrelated to indexing. In fact, we conjecture that under appropriate assumptions and definitions, the fractal dimension can indeed determine indexing properties of datasets. This approach can be generalized, by undertaking a study of topological properties of workloads that determine their indexability. The results of [12], which employ cost metrics similar to ours, are based on such topological properties for undirected graphs, and provide a good starting point for such an endeavor.

8 Conclusions

We have presented an analytical tool for indexing schemes, in the form of a lower-bound theorem for arbitrary indexing workloads, and we have demonstrated its utility by resolving a number of open problems in multidimensional range queries. We have provided a theoretical analysis of locality in d-dimensional range queries, with matching upper and lower bounds on the redundancy vs. access overhead tradeoff, and have provided useful insight into locality issues that arise in external data structures in general.

References

[1] L. Arge and J. S. Vitter. Optimal dynamic interval management in external memory. In *FOCS '96; 37th Annual Symposium on Foundations of Computer Science*, pages 560–569. IEEE, October 1996.

[2] Don Batory and Jeff Thomas. P2: A lightweight DBMS generator. Technical Report TR-95-26, University of Texas at Austin, Department of Computer Sciences, June 1995.

[3] Don S. Batory, J. R. Barnett, J. F. Garza, K. P. Smith, K. Tsukuda, B. C. Twochell, and T. E. Wise. Genesis: A reconfigurable database management system. Technical Report CS-TR-86-07, University of Texas, Austin, March 1, 1986.

[4] K. J. Falconer. *Fractal geometry : mathematical foundations and applications.* Wiley, 1990.

[5] C. Faloutsos and I. Kamel. Beyond uniformity and independence: Analysis of R- trees using the concept of fractal dimension. In *13th Symposium — 1994 May: Minneapolis; MN*, PROCEEDINGS OF THE ACM SIGACT SIGMOD SIGART SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS 1994, pages 4–13, 1994.

[6] Volker Gaede and Oliver Guenther. Multidimensional access methods. Technical Report TR-96-043, International Computer Science Institute, Berkeley, CA, October 1996.

[7] J.M. Hellerstein, E. Koutsoupias, and C.H. Papadimitriou. On the analysis of indexing schemes. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 1997.

[8] J.M. Hellerstein, J.E. Naughton, and A. Pfeffer. Generalized search trees for database systems. In *Proceedings of the 21st VLDB Conference*, 1995.

[9] P. C. Kanellakis, S. Ramaswamy, D. Vengroff, E., and J. S. Vitter. Indexing for data models with constraints and classes. In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems: May 25–28, 1993, Washington, DC*, volume 12, pages 233–243, 1993.

[10] E. Koutsoupias and David S. Taylor. Tight bounds for 2-dimensional indexing schemes. In *Proceedings of Symposium on Principles of Database Systems (PODS)*, 1998.

[11] Th. Lengauer. VLSI theory. In *Handbook of Theoretical Computer Science, Ed. Jan van Leeuwen, (Volume A: Algorithms and Complexity)*, volume 1. Elsevier and MIT Press, 1990.

[12] M. H. Nodine, M. T. Goodrich, and J. S. Vitter. Blocking for external graph searching. In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems: May 25–28, 1993, Washington, DC*, pages 222–232, 1993.

[13] S. Ramaswamy and P. C. Kanellakis. OODB indexing by class-division. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 24(2):139–150, June 1995.

[14] S. Ramaswamy and S. Subramanian. Path caching: A technique for optimal external searching. In *13th Symposium — 1994 May: Minneapolis; MN*, volume 13 of *PROCEEDINGS OF THE ACM SIGACT SIGMOD SIGART SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS 1994*, pages 25–35, 1994.

[15] S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. In *Proceedings of the 10th International Conference on Data Engineering*, pages 328–336, February 1994.

[16] Sairam Subramanian and Sridhar Ramaswamy. The P-range tree: A new data structure for range searching in secondary memory. In *Proceedings of the 6th Annual Symposium on Discrete Algorithms*, pages 378–387, January 1995.

[17] Darren Erick Vengroff and Jeffrey Scott Vitter. Efficient 3-D range searching in external memory. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 192–201, 22–24 May 1996.

[18] M. Yannakakis. Perspectives on database theory. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995.

[19] Y. Zhao, P.M. Deshpande, and J.F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In *Proc. of the 1997 ACM SIGMOD Conf. on Management of Data*, pages 159–170, May 1997.