On a Model of Indexability and its Bounds for Range Queries

JOSEPH M. HELLERSTEIN
University of California, Berkeley
and
ELIAS KOUTSOUPIAS
University of California, Los Angeles
and
DANIEL P. MIRANKER
University of Texas, Austin
and
CHRISTOS H. PAPADIMITRIOU
University of California, Berkeley and VASILIS SAMOLADAS
University of Texas, Austin

We develop a theoretical framework to characterize the hardness of indexing data sets on block-access memory devices like hard disks. We define an indexing workload by a data set and a set of potential queries. For a workload we can construct an indexing scheme, which is a collection of fixed-sized subsets of the data. We identify two measures of efficiency for an indexing scheme on a workload: *storage redundancy*, r (how many times each item in the data set is stored), and *access overhead*, A (how many times more blocks than necessary does a query retrieve).

For many interesting families of workloads, there exists a trade-off between storage redundancy and access overhead. Given a desired access overhead A, there is a minimum redundancy that any indexing scheme must exhibit. We prove a lower-bound theorem for deriving the minimum redundancy. By applying this theorem we show interesting upper and lower bounds and trade-offs between A and r in the case of multi-dimensional range queries and set queries.

Categories and Subject Descriptors: H.2.2 [**Information Systems**]: Database Management—*Access Methods*

General Terms: Theory

Additional Key Words and Phrases: Index, indexability, database, redundancy, lower bounds

1. INTRODUCTION

Upon its definition, the B-tree promptly proved to be an effective access method for the primary applications of relational databases [Bayer and McCreight 1972]. The success and ubiquity of the relational data model arguably owes much to the timely definition of the B-tree. Since then, a major thrust of database research has been to extend the relational

Hellerstein's work was supported by NSF Grant IRI-9703972, NASA grant FDNAGW-5198, and a Sloan Foundation Fellowship.

model and relational systems to manage more complex types and more expressive query languages. The B-tree is widely recognized to be an inadequate data structure in many of the novel contexts, and no single, general-purpose successor has emerged to enable the diversity of applications and requirements for contemporary information systems. Therefore, it is important to develop general methodologies and tools for the design of new indexing methods, as well as mathematical tools to, a priori, evaluate their performance and identify their limitations.

A systems approach to the "generalized indexing" problem has been proposed and implemented [Hellerstein et al. 1995; Kornacker et al. 1997; Aoki 1998; Kornacker 1999]. The results highlighted the need for theoretical tools to rigorously analyze indexing problems. To aid developers of new indexes in this general framework, a kind of *theory of indexability* is required: a mathematical model that allows the performance and scalability of an indexing scheme to be evaluated much as complexity theory is used to evaluate algorithms. Where complexity theory considers in-memory data structures, a theory of indexability must consider the impact of disk-based *secondary storage*.

Our pragmatic results focus on the multi-dimensional range search problem, a common workload for many advanced applications. An enormous amount of experimental research has been devoted to this problem: a recent survey cites over 50 different multi-dimensional data structures [Gaede and Günther 1998]. Many commercial vendors of Object-Relational Database Systems and Geographic Information Systems use one of these structures, typically some variant of the R-tree [Guttman 1984], the Grid File [Nievergelt et al. 1984] or disk-resident adaptations of the quad-tree [Samet 1989]. This research is primarily experimental. Analytic research on these structures has concentrated on probabilistic and empirical studies of their average-case performance, under various data and query distributions.

At the same period of time that heuristic disk-based indices such as the R-Tree were introduced, the computational geometry community was studying main-memory data structures for range searching, paying little attention to secondary memory. In contrast to most multi-dimensional indexing research by the database community, the work in computational geometry is mostly theoretically oriented, with an emphasis on worst-case asymptotic performance. We believe that the striking contrast between these two approaches to the same problem arose from a fundamental fact: for two-dimensional range searching (and more so for higher dimensions), optimal query cost cannot be achieved with space proportional to the data set, but instead requires *significant* storage redundancy, typically by a *multiplicative factor* at least logarithmic to the size of the data set [Chazelle 1990a]. Of course, redundancy has often been used in databases to accelerate performance: index structures are themselves typically redundant to the data sets they index, and the addition of logarithmic space is standard for upper levels in search trees. However, the space cost of redundancy in databases has rarely been as high as a logarithmic *multiple* to the size of the data set. This is only reasonable: databases usually store very big data sets, on top of which a logarithmic factor of redundancy makes the solution considerably more expensive in space. Also, high redundancy increases the I/O cost of online updates, at least proportionally to the redundancy. For these reasons, low-redundancy access methods are typically used in practice [Kornacker 1999; Kanth et al. 1999][1].

[1]A notable exception to this rule is the inverted index technique widely used for text retrieval (see, e.g., [Witten et al. 1999]), in which each document identifier is replicated in the index about as many times as terms in the

Thus, database research concentrated on data structures with low redundancy, with very bad worst-case behavior, but with the hope of reasonable average-case behavior for real workloads. Lowering the observed average-case cost has typically been achieved through problem-specific heuristics, which take into account the particularities of various data sets and query workloads. For example, access methods for two-dimensional geographic data have been differentiated from access methods for temporal queries over one-dimensional data, by a choice of heuristics appropriate to the expected distributions of respective data sets and typical queries, despite the fact that from a conceptual point of view the two problems are equivalent.

The two approaches presented above (ad hoc and application-dependent indexing schemes versus highly redundant computational geometry data structures) are in a sense extreme: one penalizes worst-case query performance by keeping space linear, the other strongly favors query performance without regard to the storage cost becoming prohibitive. The results in this paper strive to reconcile these two approaches by exposing and studying the fundamental trade-off between I/O time and space for these problems, and investigating techniques that are parameterized by the total space or the desired worst-case query cost.

1.1 Indexing Workloads and Indexing Schemes

Database access methods must be evaluated in the context of a particular *workload*. A workload consists of an *instance* of a database (a finite subset of some domain), together with a set of queries (a given set of subsets of the instance). In one-dimensional indices such as B-trees, for example, the instance is some totally ordered finite set, and the most common queries considered are *range queries*, that is to say, intervals of this order. Other common workloads include multi-dimensional point sets with range queries, sets of intervals with stabbing queries, and powersets with intersection or inclusion queries.

In what we term indexability theory, the workload plays a role similar to the role a partially recursive language plays in complexity or decidability theory: it is the unit whose complexity must be characterized[2]. For each workload we have a space of possible indexing schemes; the analog of algorithms that partially decide the language. Such an indexing scheme is a collection of B-subsets of the instance, which we call blocks. The *block size* B is assumed fixed and very large (usually in the hundreds). The union of the blocks exhausts the instance. Each query is answered by retrieving a set of blocks, whose union is a superset of the query.

Our approach suppresses important aspects of indexing, such as the algorithms for determining the partition of the instance into blocks (possibly with repetitions), as well as the algorithms for determining, given a query, the blocks in the indexing scheme that cover it (e.g., the cost of traversing a tree to its leaf level). Furthermore, we also ignore the storage and retrieval costs necessary to support such algorithms, e.g., auxiliary information such as “directories” or “internal nodes”. These omissions are justified in three ways: first, we are mostly interested in lower bounds, and therefore we are free to disregard aspects of the complexity of the problem. Second, in practice, these aspects do not appear to be the source of design difficulties or of complexity—it appears that good assignment of data items to

document. This replication means that online, concurrent updates to text indexes are not widely supported in practice in text retrieval systems.

[2]More accurately, the analog of a language is a *family of workloads*, one for each cardinality of the instance. Such growing families of workloads allow us to focus on asymptotic analysis and ignore additive constants.

blocks tends to suggest efficient traversal algorithms, and to have low storage overhead. Third, secondary storage techniques such as buffer management mask and absorb many of these auxiliary cost components. However, our model also ignores the dynamic aspect of the problem, i.e., the cost of insertion and deletion. Its consideration *could* be a source of added complexity, and in a more general model the source of more powerful lower bounds.

In this paper we propose a model for the indexing of a data set with respect to a given workload, and explore in its light the fundamental properties and trade-offs of indexing, with an emphasis on lower bounds. In particular, we introduce a lower-bound theorem that is applicable to arbitrary workloads —although it is not guaranteed to always yield tight bounds. We also analyze within this model a number of interesting families of workloads, including multi-dimensional point sets with range queries, and powersets with subsumption queries. Besides revealing some interesting laws, our results indicate positive prospects for the use of limited redundancy. For example, for two-dimensional range queries, even a small amount of redundancy can significantly decrease the worst-case query cost.

2. RELATED WORK

This work was initially motivated by the work of Hellerstein, Naughton and Pfeffer on the Generalized Search Tree (or *GiST*) [Hellerstein et al. 1995]. The GiST is an extensible template 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.”

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.

Previous work on index data structures concentrated on the study of specialized problems. In the area of multi-dimensional indexing, data structures are often classified into two categories: those that partition the data set, such as R-trees and their variants, and those that partition the search space, such as quad-trees and their variants. In both categories, most of the proposed algorithms are based on heuristics, and all have relatively bad worst-case asymptotic performance. It is not clear whether this classification has any definitive bearing on performance, and no clear winner has emerged among the many proposals, even for well-understood families of workloads. A comprehensive exposition of the relevant work can be found in the survey of spatial access methods of Gaede and Günther [Gaede and Günther 1998], and the survey of temporal access methods of Salzberg and Tsotras [Salzberg and Tsotras 1999].

This situation has been changing in the past few years, mostly due to the work of Kanellakis, Vitter, and their collaborators. In [Kanellakis et al. 1993], it was shown that multi-dimensional range search generalizes indexing problems in new database paradigms such as constraint databases and class hierarchies. In subsequent publications, [Ramaswamy and Subramanian 1994; Ramaswamy and Kanellakis 1995; Subramanian and Ramaswamy 1995; Vengroff and Vitter 1996] asymptotically efficient dynamic algorithms are presented for two-sided and three-sided range queries, and for interval stabbing queries. An optimal solution to the interval management problem has recently been found by Arge and

Vitter [Arge and Vitter 1996]. Most of this work involves upper bounds, and is therefore mainly concerned with the analysis of the searching aspect of the problem. There are two exceptions: First, in [Kanellakis et al. 1996] there is an argument (proof of Lemma 2.7) that anticipates our Theorem 4.1, namely, that the access overhead must be $\sqrt{B}$ *in the special case in which the blocks are restricted to be rectangular.* Second, in the last section of [Subramanian and Ramaswamy 1995], there is an interesting lower bound, where it is shown (by extending a result by Chazelle [Chazelle 1990a] to the case of block accesses) that storage redundancy $\Omega(\log n / \log \log_B n)$ is necessary if *additive* (as opposed to our multiplicative) access overhead is to remain polynomial in $\log_B n$. Also related are the results in [Nodine et al. 1993], who use cost metrics similar to ours, to characterize the locality in external graph searching.

The question of lower bounds in multi-dimensional searching has been addressed in [Mehlhorn 1984], without, however, our emphasis on block accesses. Similar work is presented in [Smid and Overmars 1990], where lower bounds are derived in a model involving binary trees with certain further restrictions; the block size is considered in that paper as a function of n, the number of points. Finally, in the database literature there has been extensive analysis (worst case, expected case, or empirical/experimental) of many access methods for multi-dimensional searching (see, for example, [Pagel et al. 1993; Faloutsos and Kamel 1994; Belussi and Faloutsos 1995]). More recently, the ideas presented here have been used in a more rigorous framework for empirically analyzing and tuning indexing performance [Shah et al. 1999].

The concept of a space/time tradeoff in main-memory range searching has been studied thoroughly [Fredman 1980; 1981; Yao 1982; Vaidya 1989; Chazelle 1990a; 1990b; 1995]. All these works consider variants of either the RAM machine, or the pointer machine. These memory models are fundamentally different from block-structured secondary memory.

The *cell probe model*, originally introduced by Yao [Yao 1981], is a general framework for dealing with data structure problems, especially valuable for proving lower bounds, and space-time trade-offs in particular. Let f be any mapping from query $Q \in \{0,1\}^q$ and dataset $d \in \{0,1\}^n$ to the answer $f(Q,d)$ of query Q over d. The cell probe model assumes the existence of a memory of s cells, each cell of b bits. Let t be the maximum, over all Q and d, of the least number of cells that must be accessed in order to compute $f(Q,d)$. We are interested in trade-offs between s and t, with b a parameter of the model. Miltersen et.al [Miltersen et al. 1995] proposed some general lower bounds techniques, employing asymmetric communication complexity, and applied them to certain data structure problems related to set membership.

The cell probe model is more general than the indexability model in this paper, because in it memory can be organized in an arbitrary way, whereas in indexability we assume that the memory contains explicit representations of the records. The cell probe model has been used in the past to derive lower bounds in geometric problems; for example, [Chakrabarti et al. 1999; Barkol and Rabani 2000] applied this model to the *nearest neighbor* problem, a pure search problem for which indexability yields trivial results. However, to date the cell probe model has not been applied to range *reporting* problems, which is the class of problems with which indexability is concerned. By "reporting problems" we mean, informally, data structure problems in which the output of the algorithm must be a set of records (think of them as strings or pointers), and the algorithm is not allowed to look inside these records. Reporting problems are an appropriate framework for database storage problems, as they reflect the data independence present in databases. For reporting problems, it makes sense to restrict the data structure solutions so that each memory location holds a record, as we do in the indexability framework, forfeiting the generality of the cell probe model. As one of the referees pointed out to us, by restricting the records stored to be single bits, the cell probe model can be adapted to prove certain lower bounds similar to our bounds for set workload reporting problems (Section 7), starting from the communication complexity results of [Miltersen et al. 1995]. These bounds are quantitatively weaker than ours, but hold in a more general model (albeit a model the generality of which is inaccessible to the database problems of interest here). It would be interesting to find such cell probe lower bounds for range search workloads (our main concern in this paper).

3. DEFINITIONS

In this section, we set out a simple framework for defining an indexing problem, and for measuring the efficiency of a particular indexing scheme for the problem.

3.1 Indexing Workloads

Indexing schemes must be evaluated in the context of a particular *workload*, consisting of a finite subset of some domain together with a set of queries. More formally, we have the following definition:

DEFINITION 3.1. *A workload W is a tuple $W = (D, I, \mathcal{Q})$, where D is a non-empty set (the* domain*), $I \subseteq D$ is a non-empty finite set (the* instance*), and $\mathcal{Q}$ is a set of subsets of I (the* query set*).*

A workload we consider extensively is the set of two-dimensional range queries. This workload consists of the domain $\mathbb{R}^2$, the instance $I = \{(i, j) : 1 \leq i, j, \leq n\}$, and the family of “range queries” $Q[a, b, c, d] = \{(i, j) : a \leq i \leq b, c \leq j \leq d\}$, one for each quadruple (a, b, c, d) with $1 \leq a \leq b \leq n, 1 \leq c \leq d \leq n$. Notice that this is a *family of workloads*, with instances of increasing cardinality, one for each $n \geq 0$. Another family of workloads (the set inclusion queries) has as its domain, for each n, all subsets of $\{1, 2, \ldots, n\}$, and for each subset I of the domain, the set of queries $\mathcal{Q} = \{Q_S : S \subseteq \{1, 2, \ldots, n\}\}$, where $Q_S = \{T \in I : T \subseteq S\}$.

In the terminology of combinatorics, W is a simple hypergraph, where I is the vertex set, and $\mathcal{Q}$ is the edge set. The hypergraph abstraction has been used in related work to measure the quality of existing indexing schemes on particular workloads [Shah et al. 1999]. We do not use this terminology here, choosing instead to define terms more natural for databases. There is no analog of the domain D in hypergraphs. We could have dropped it from our definition, but it is suggestive of a parameterization of workloads. For example, all two-dimensional range-query workloads have the same domain.

3.2 Indexing Schemes

DEFINITION 3.2. *An indexing scheme $\mathcal{S} = (W, \mathcal{B})$ consists of a workload $W = (D, I, \mathcal{Q})$, and for some positive integer B a set $\mathcal{B}$ 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 the total number of blocks $|\mathcal{B}|$. Notice that an indexing scheme is a simple, B-regular hypergraph with vertex set I.

As a convention in this paper, we will use lower-case letters from the end of the alphabet, x, y, z to represent elements of I, 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.

3.3 Performance Measures

We now define two performance measures on indexing schemes, *redundancy* and *access overhead*, evaluating the performance of the scheme in terms of space and I/O, respectively. In particular, redundancy measures the amount of space needed by the indexing scheme, while access overhead measures the amount of I/O required by queries. In both cases, the measures are normalized by the ideal performance (linear space and size of the query, respectively). In the following definitions, let $\mathcal{S} = (W, \mathcal{B})$ be an indexing scheme of block size B on workload $W = (D, I, \mathcal{Q})$, and let $N = |I|$.

3.3.1 *Storage Redundancy*

DEFINITION 3.3. *The redundancy $r(x)$ of $x \in I$ is 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 the number of blocks is $K = \frac{rN}{B}$.

We also define the maximum redundancy $\hat{r}$ in $\mathcal{S}$, as $\hat{r} = \max_{x \in I} r(x)$.

3.3.2 *Access Overhead*

DEFINITION 3.4. *A set of blocks, $U \subseteq \mathcal{B}$,* covers *a query $Q \in \mathcal{Q}$, iff $Q \subseteq \bigcup_{b \in U} b$.*

DEFINITION 3.5. *A cover set, $C_Q \subseteq \mathcal{B}$, for query $Q \in \mathcal{Q}$ 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 $Q \in \mathcal{Q}$ is defined as*

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

where $C_Q \subseteq \mathcal{B}$ is a cover set for Q.

It is easy to see that $1 \leq A(Q) \leq B$, since any query Q will be covered by at most $|Q|$ blocks.

Informally, $A(Q)$ models the observed cost of query Q normalized by its ideal cost, 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 $\mathcal{S}$, to be the maximum of $A(Q)$ over all queries.

DEFINITION 3.7. *The access overhead A for indexing scheme $\mathcal{S}$ is*

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

Notice that, although the redundancy is defined as an average (over all data items), the access overhead is a maximum (over all queries). This is less arbitrary than it may seem at first. By averaging over all data items we capture the true (worst-case) space performance of the indexing scheme, while averaging I/O performance over all queries would be much less defensible since queries are generally not equiprobable, and guarantees, and thus worst-case analysis, are desirable in the context of query response time.

3.4 Some Trivial Bounds and Tradeoffs

Based on standard properties of databases and disks, we assume that the number of objects N is always much greater than the block size B, although B is not limited in any concrete way.

For some indexing scheme $\mathcal{S}$, the minimum possible redundancy is 1, when $\mathcal{B}$ is a partition of I, and the maximum redundancy is $\binom{N-1}{B-1}$, when $\mathcal{B} = \binom{I}{B}$.[3] For $\mathcal{S}$ having maximum redundancy, A is exactly 1, which is minimum; in that case, every query Q can be covered by a set of disjoint blocks whose union contains Q. Also, for $r = 1$ it is easy to devise a problem where $A = B$, which is maximum (e.g., $\mathcal{Q} = \binom{I}{B}$).

4. TRADEOFFS FOR A TWO-DIMENSIONAL WORKLOAD

Given this framework for indexability, we proceed to examine some families of workloads that have received significant attention in the indexing literature. Our main goal is to expose the tradeoffs in lower bounds of r and A for these workloads, delimiting the potential efficiency of indexes for these workloads. We start with some simple positive results (upper bounds) that are useful in two ways. First, they illustrate the framework of indexing schemes. And second, they allow us to conclude later that the lower bounds of this paper are tight.

Our main lower bound results are driven by the *Redundancy Theorem* that we develop in Section 5. However, we do not need the Redundancy Theorem to obtain our first interesting lower bound, which is presented in the second part of section.

4.1 Two-Dimensional Queries

We shall consider here workloads over the two-dimensional domain $\mathbb{R}^2$, with $I = \{(i, j) : 1 \leq i, j, \leq n\}$, and 2-d range queries over this instance. We are interested in determining the minimum possible access overhead when the redundancy r is fixed.

PROPOSITION 1. *For each integer r, there is an indexing scheme $\mathcal{S}_r$ with redundancy r and access overhead $2B^{\frac{1}{2r}} + 2$.*

PROOF. The main idea for the indexing scheme $\mathcal{S}_r$ is that each query Q of $x \times y$ points will be covered by disjoint blocks of $\mathcal{S}_r$ that have "almost" the same aspect ratio y/x with Q. The ideal situation is to have blocks with aspect ratio y/x, so that the query Q is tiled nicely by these blocks; compare this with the worst case when the query Q is "long and narrow" and it is covered by "short and wide" blocks. Because of the restriction on the

[3]For set S and $n \geq 0$, $\binom{S}{n}$ denotes the set of all n-subsets of S.

redundancy r of the indexing scheme $\mathcal{S}_r$, it is not possible to have blocks for each aspect ratio. However, we can choose blocks so that any aspect ratio can be approximated.

More precisely, for each $i = 1, 2, \ldots, r$, our indexing scheme $\mathcal{S}_r$ contains all $B^{\frac{2i-1}{2r}} \times B^{\frac{2r-2i+1}{2r}}$ blocks that partition I. The aspect ratios $B^{\frac{r-2i+1}{r}}$, for $i = 1, 2, \ldots, r$, of these blocks are evenly distributed. It is immediate that $\mathcal{S}_r$ has redundancy r (maximum as well as average). To see that $\mathcal{S}_r$ has access overhead at most $2B^{\frac{1}{2r}} + 2$, consider the set of $B^{\frac{j}{r}} \times B^{\frac{r-j}{r}}$ queries, $j = 0, 1, \ldots, r$. Clearly, the best coverage of such a query is by blocks that have almost the same aspect ratio, that is, blocks of size $B^{\frac{2j-1}{2r}} \times B^{\frac{2r-2j+1}{2r}}$ or blocks of size $B^{\frac{2j+1}{2r}} \times B^{\frac{2r-2j-1}{2r}}$. In both cases, when the query is "aligned" with the blocks, it requires $B^{\frac{1}{2r}}$ blocks (either one row of $B^{\frac{1}{2r}}$ blocks or one column of $B^{\frac{1}{2r}}$ blocks). For non-aligned queries the ratio can be as high as $2B^{\frac{1}{2r}} + 2$; to see this, consider the case where an aligned query is satisfied by a row of $B^{\frac{1}{2r}}$ blocks. If we shift this query out of horizontal and vertical alignment, we need two rows of blocks instead of one, and at one of the ends we need an additional column of two blocks as well. On the other hand, it is not difficult to see that these are the worst queries for this indexing scheme.

□

If the access ratio is A, the above scheme has both average and maximum redundancy $r = \Omega(\log B / \log A)$. We will show that this is the best possible relation between r and A. Indeed, in the remainder of this section we prove that this is the case when the maximum redundancy is one. We will defer the study of the general case after we introduce our lower-bound theorem.

4.2 A lower bound for redundancy $r = 1$

We will show that up to a constant factor the above indexing scheme is optimal when $r = 1$. In [Kanellakis et al. 1993], the result below was shown for the special case when the blocks are restricted to be rectangular.

THEOREM 4.1. *Any indexing scheme for 2-dimensional range queries with redundancy $r = 1$ has access overhead at least $A = B^{\frac{1}{2}}$. For the d-dimensional case, the lower bound is $A = B^{1-\frac{1}{d}}$.*

PROOF. We consider first the 2-dimensional case, the general case being a straightforward generalization. For simplicity, we assume that n is a multiple of B.

For the lower bound, we consider only queries of size $1 \times B$ and $B \times 1$. The queries of size $1 \times B$ partition the instance and so do the queries of size $B \times 1$. The total number of queries is $2n^2/B$.

Now fix a block $b \in \mathcal{B}$ that intersects x_1 horizontal lines and x_2 vertical lines (by a "line" we mean a set of data points of the form $\{(1, j), (2, j), \ldots, (n, j)\}$ or $\{(i, 1), (i, 2), \ldots, (i, n)\}$). Since every block has B points, we must have $x_1 x_2 \geq B$; hence $x_1 + x_2$ is at least $2B^{\frac{1}{2}}$. Therefore, every block intersects at least $2B^{\frac{1}{2}}$ of the above queries. Taking into account that the number of queries is twice the number of blocks, we can conclude that, on the average, every query of the above collection is intersected by $B^{\frac{1}{2}}$ blocks at least. (To see this in detail, consider the number of pairs of intersecting blocks and queries; it is no less than $2B^{\frac{1}{2}}$ times the total number of blocks, which is $2B^{\frac{1}{2}} n^2/B$; since there are $2n^2/B$ queries in total in the collection, the average number of intersecting blocks per query is $B^{\frac{1}{2}}$.) When the redundancy is $r = 1$, all these blocks are needed to cover the query.

Notice that we showed not only that there exists a query with access overhead $B^{\frac{1}{2}}$, but that this is the expected access overhead for a random query from the above set.

The generalization to the d-dimensional case is straightforward (for example we now have $x_1 + \ldots + x_d \geq dB^{\frac{1}{d}}$ which gives access overhead at least $B^{1-\frac{1}{d}}$). □

5. THE REDUNDANCY THEOREM

We now turn our attention to a workload-independent analysis of the indexability model that culminates with 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; it arises in extremal set theory. The reader is warned that the notation does not correspond to indexing schemes.

THEOREM 5.1. *Let $S_1, S_2, \ldots, S_a$ ($a \geq 1$) be non-empty finite sets, $S = S_1 \cup S_2 \cup \ldots \cup S_a$ be their union, and $L \leq |S|$ 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,

$$k \geq \frac{|S|}{L} - a \tag{1}$$

PROOF. Let $P_1, \ldots, P_k$ be sets that satisfy the properties of the theorem and let P be their union, $P = P_1 \cup P_2 \cup \ldots \cup P_k$. The maximality of k guarantees that P contains all but at most L elements from every S_i, $i = 1, \ldots, a$. That is, $|S_i \setminus P| < L$ (otherwise we can add any subset of L elements of $S_i \setminus P$ to the collection of P_i's). We can now estimate $|S \setminus P| = |(\bigcup S_i) \setminus P| = |\bigcup(S_i \setminus P)| \leq \sum_{i=1}^{a} |S_i \setminus P| < aL$. Since every P_j has cardinality L we conclude that $kL = |P| > |S| - aL$ which implies the desired $k > |S|/L - a$.

□

To apply the above theorem to the domain of indexing schemes, we define a convenient concept, *flakes*, to capture the overlap of queries and blocks.

DEFINITION 5.2. *Let $\mathcal{S} = (W, \mathcal{B})$ be an indexing scheme on workload $W = (D, I, \mathcal{Q})$. A flake is any set of objects $F \subseteq I$ such that for some query Q and some block b, $F \subseteq Q \cap b$.*

Note that a flake is a subset (potentially a proper subset) of the intersection of a block and a query. The flexibility to deal with proper subsets will allow us to consider flakes of a fixed size, allowing us to apply certain combinatorial results below.

We now have the following lemma on flakes:

LEMMA 5.1 FLAKING LEMMA. *Let $\mathcal{S}$ be an indexing scheme, A be its access overhead, and ϑ be a real number in the interval $[2, \frac{B}{A}]$ such that $\frac{B}{\vartheta A}$ is an integer. Then, any query Q with $|Q| \geq B/2$ will contain at least $(\vartheta - 2)A\frac{|Q|}{B}$ pair-wise disjoint flakes of size $\frac{B}{\vartheta A}$.*

PROOF. The parameter ϑ exists only to guarantee that $\frac{B}{\vartheta A}$ is integer.

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

$$a = A(Q)\left\lceil \frac{|Q|}{B} \right\rceil \leq A\left\lceil \frac{|Q|}{B} \right\rceil \leq 2A\frac{|Q|}{B}$$

(because $|Q| \geq B/2$). We apply Theorem 5.1 on S_i for $L = \frac{B}{\vartheta A}$, and conclude that the number k of flakes of size $\frac{B}{\vartheta A}$ is at least

$$\begin{aligned} k &\geq \frac{|Q|}{\frac{B}{\vartheta A}} - a \\ &\geq \vartheta A\frac{|Q|}{B} - 2A\frac{|Q|}{B} \\ &= (\vartheta - 2)A\frac{|Q|}{B} \end{aligned}$$

□

We proceed to prove a second technical tool from extremal set theory. In coding theory, under a slightly different statement, this result is known as Johnson's bound [Johnson 1962]. Again, the notation does not correspond to indexing schemes.

THEOREM 5.3 JOHNSON'S BOUND. *Let S be a finite set, and $S_1, S_2, \ldots, S_k$ be subsets of S, each of size at least $\alpha|S|$, such that the intersection of any two of them is of size at most $\beta|S|$. If $\beta < \frac{\alpha^2}{2-\alpha}$, the number of subsets k is at most α/β.*

PROOF. Since $S_1, S_2, \ldots, S_t$, $t \leq k$, are subsets of S, their union $S_1 \cup S_2 \cup \ldots \cup S_t$ is also a subset of S and therefore

$$|\bigcup_{j=1}^{t} S_j| \leq |S|.$$

It follows that

$$\sum_{j=1}^{t} |S_j| - \sum_{j=1}^{t}\sum_{l=j+1}^{t} |S_j \cap S_l| \leq |S|.$$

By the assumptions about the sizes of the subsets and their pairwise intersection, the last inequality implies that

$$t\alpha|S| - \binom{t}{2}\beta|S| \leq |S|.$$

Therefore, every $t \leq k$ must satisfy the inequality $\alpha t - \beta\binom{t}{2} - 1 \leq 0$. It immediately follows that if a positive integer t does not satisfy this inequality, then the number k of subsets must be less than t. So, in order to upper bound the number k of subsets, we need to guarantee that the above inequality is not satisfied by at least one positive integer. This can be easily done if we require that the two roots of the polynomial $\alpha t - \beta\binom{t}{2} - 1$ differ by more than 1. Since the roots of the polynomial are

$$\frac{\alpha + \beta/2 \pm \sqrt{(\alpha + \beta/2)^2 - 2\beta}}{\beta},$$

it is easy to verify that they differ by more than 1 when $\beta < \alpha^2/(2-\alpha)$.

But then, the number of subsets is at most equal to the minimum root of the above polynomial. Thus

$$k \leq \frac{\alpha + \beta/2 - \sqrt{(\alpha+\beta/2)^2 - 2\beta}}{\beta}.$$

This last inequality implies that $k \leq \alpha/\beta$. □

Note that the hypotheses of the above lemma cannot be improved by a factor of more than 2, because when $\beta \geq \alpha^2$, the number of possible subsets is unbounded, i.e., it is an increasing function of $|S|$.

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

THEOREM 5.4. *Let $\mathcal{S}$ be an indexing scheme, and let $Q_1, Q_2, \ldots, Q_M$ be queries, such that for every i, $1 \leq i \leq M$:*

(*1*) $|Q_i| \geq B/2$, *and*

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

Then, the redundancy is bounded by

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

where ϑ is any real number in the interval $[2, \frac{B}{A}]$ such that $\frac{B}{\vartheta A}$ is integer.

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

5.0.0.1 *Step 1.* Consider any query Q_i. By the flaking lemma, this query contains at least $(\vartheta - 2)A\frac{|Q_i|}{B}$ *disjoint* flakes of size $\frac{B}{\vartheta A}$. Let F be such a flake. F cannot be contained in 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 $|Q_i \cap Q_j| \geq \frac{B}{\vartheta A} > \frac{B}{2(\vartheta A)^2}$. We conclude that

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

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

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

Substituting we get

$$f_1/f_2 = \frac{\vartheta - 2}{2\vartheta} \sum_{i=1}^{M} \frac{|Q_i|}{B}$$

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

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

□

Notice that the theorem is useful only for access overhead $A = O(\sqrt{B})$: either all queries are disjoint (implying access overhead 1), or for non-disjoint queries Q_i and Q_j, the second premise in the statement of the theorem implies that $\frac{B}{2(\vartheta A)^2} \geq 1$.

We now simplify the theorem by removing the parameter ϑ.

THEOREM 5.5 REDUNDANCY THEOREM. *Let $\mathcal{S}$ be an indexing scheme with access overhead $A \leq \sqrt{B}/4$, and let $Q_1, Q_2, \ldots, Q_M$ be queries, such that for every i, $1 \leq i \leq M$:*

(1) $|Q_i| \geq B/2$, *and*
(2) $|Q_i \cap Q_j| \leq \frac{B}{16A^2}$ *for all $j \neq i$, $1 \leq j \leq M$.*

Then, the redundancy is bounded by

$$r \geq \frac{1}{12N} \sum_{i=1}^{M} |Q_i|.$$

PROOF. Let $\vartheta_1 = 12/5$ and $\vartheta_2 = 2\sqrt{2}$. We first show that there exists $\vartheta \in [\vartheta_1, \vartheta_2]$ such that $\frac{B}{\vartheta A}$ is integer. This follows from $\frac{B}{\vartheta_1 A} - \frac{B}{\vartheta_2 A} = (\frac{1}{\vartheta_1} - \frac{1}{\vartheta_2})\frac{B}{A} \geq (\frac{1}{\vartheta_1} - \frac{1}{\vartheta_2})\frac{B}{A^2} \geq (\frac{1}{\vartheta_1} - \frac{1}{\vartheta_2})16 > 1$.

Using such a ϑ in Theorem 5.4, the second premise becomes

$$|Q_i \cap Q_j| \leq \frac{B}{16A^2} = \frac{B}{2(\vartheta_2 A)^2} \leq \frac{B}{2(\vartheta A)^2}$$

and the factor $\frac{\vartheta-2}{2\vartheta}$ of the conclusion becomes

$$\frac{\vartheta - 2}{2\vartheta} \geq \frac{\vartheta_1 - 2}{2\vartheta_1} = \frac{1}{12}.$$

□

Observe that given any set of queries $\mathcal{M} = \{Q_1, \ldots, Q_M\}$, we can construct blocks for each query independently, for a total of

$$T_{\mathcal{M}} = \sum_{i=1}^{M} \left\lceil \frac{|Q_i|}{B} \right\rceil$$

blocks, achieving a perfect access overhead of one, with redundancy $r = T_{\mathcal{M}} \frac{B}{N} \geq \frac{\sum_{i=1}^{M} |Q_i|}{N}$. The Redundancy Theorem states that when the queries intersect pairwise in at most $\frac{B}{16A^2}$ elements for some A, increasing the access overhead to A does not yield an improvement in space by more than a constant factor of $T_{\mathcal{M}}$.

6. LOWER BOUNDS FOR MULTI-DIMENSIONAL RANGE QUERIES

We now 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 domain is $\mathbb{R}^d$, with instance $I = [1:n]^d$ and query set

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

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

6.1 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 $B/2$, and with pairwise intersections at most $\frac{B}{16A^2}$. 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 $n \times n$ grid, 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 the parameter c.

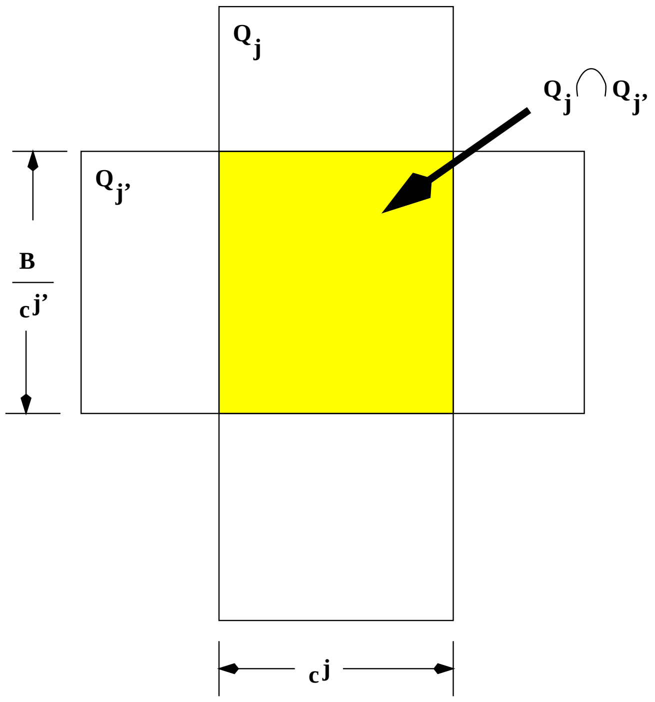

Fig. 1. Two rectangles of sizes $c^j \times \frac{B}{c^j}$ and $c^{j'} \times \frac{B}{c^{j'}}$, $j < j'$, intersecting in at most $\frac{B}{c^{j'-j}}$ points.

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. Figure 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 $c = 16A^2$.

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

$$\begin{aligned} r &\geq \frac{1}{12}\frac{MB}{n^2} \\ &= \frac{1}{12}\frac{1}{n^2}\left(B\frac{n^2}{B}(1+\log_c B)\right) \\ &= \frac{1}{12}(1+\log_c B) \\ &\geq \frac{1}{12}\log_c B \\ &= \frac{1}{12}\frac{\log B}{\log(16A^2)} \end{aligned}$$

and thus we have

$$r = \Omega\left(\frac{\log B}{\log A}\right)$$

6.2 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 $c = 16A^2$ is applicable in this case also, guaranteeing that the intersection of any two rectangles will have size at most $\frac{B}{16A^2}$.

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

$$M = \frac{n^d}{B}\binom{\log_c B + d - 1}{d-1} = \frac{n^d}{B}\binom{\frac{\log B}{\log(16A^2)} + d - 1}{d-1}$$

and for the redundancy we have

$$r \geq \frac{1}{12}\binom{\frac{\log B}{\log(16A^2)} + d - 1}{d-1}$$

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

THEOREM 6.1. *For workload $\mathcal{R}_n^d$, the storage redundancy is bound by*

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

6.3 Fibonacci Workload

So far, our trade-offs have depended only on the block size B, but not on the size of the instance. Unfortunately, this is not always the case. In this section, we study a family of workloads for two-dimensional range queries that exhibits much worse performance.

Using our framework of indexing schemes, it was shown in [Koutsoupias and Taylor 1998] that there exist simple 2-dimensional workloads with tradeoffs that depend on the instance size. In particular, they studied range queries of the Fibonacci lattice (to be defined shortly) and showed that any indexing scheme with redundancy less than $\Theta(\log n)$ has the worst possible overhead $A = B$. The bound $\Theta(\log n)$ is tight up to a constant factor. They later extended the results to random sets of points and higher dimensions [Koutsoupias and Taylor 1999].

Here we illustrate the power of the Redundancy Theorem by extending the results for the Fibonacci lattice when the access overhead is small, $A = O(\sqrt{B})$. Furthermore, we give the precise trade-off between redundancy and access overhead.

We now define the Fibonacci lattice, which is the regular lattice rotated appropriately. Let $n = f_k$ be the k-th Fibonacci number. The Fibonacci lattice F_n is the set of points defined by:

$$F_n = \{(i, if_{k-1} \bmod n) : i = 0, 1, \ldots, n-1\} \text{ for } n = f_k.$$

The Fibonacci workload over domain $\mathbb{R}^2$ is defined by taking the Fibonacci lattice as the instance I, and all rectangular queries as $\mathcal{Q}$.

We will only need the following property of the Fibonacci lattice, from [Fiat and Shamir 1989]:

PROPOSITION 2. *For the Fibonacci lattice F_n of n points, and for $t \geq 0$, any rectangle with area $t \cdot n$ contains between $\lfloor t/c_1 \rfloor$ and $\lceil t/c_2 \rceil$ points, where $c_1 \approx 1.9$ and $c_2 \approx 0.45$.*

Now we apply the Redundancy Theorem to the Fibonacci workload. We have to define an appropriate set of queries $Q_1, \ldots, Q_M$, each of cardinality at least $B/2$.

We consider rectangles of area $a = c_1 Bn/2$. By Proposition 2, each such rectangle will contain at least $B/2$ points. Let c be a parameter to be specified later. We consider rectangles of dimensions $c^i \times \frac{a}{c^i}$, for appropriate values of i. For each such aspect ratio, we partition the Fibonacci lattice into non-overlapping rectangles, in a tiling fashion. Each of these rectangles will define a query.

Because no rectangle can have a side longer than n, we must constrain i to obey

$$c^i \leq n \quad \text{and} \quad \frac{a}{c^i} \leq n$$

From these, we compute that i must range between $\log_c \frac{c_1 B}{2}$ and $\log_c n$, i.e., approximately $\log_c \frac{2n}{c_1 B}$ aspect ratios. Since for each i we cover the whole set of points, the Redundancy Theorem gives

$$r \geq \frac{1}{12} \log_c \frac{2n}{c_1 B} = \Omega\left(\frac{\log \frac{n}{B}}{\log c}\right)$$

Now we specify an appropriate value of parameter c that satisfies the second premise of the Redundancy Theorem—which states that no two queries can intersect by more than $\frac{B}{16A^2}$ points. We observe that rectangles of the same aspect ratio do not intersect, and rectangles of different aspect ratios have intersections of area at most a/c. Again by Proposition 2, it suffices to have

$$\left\lceil \frac{a/c}{c_2 n} \right\rceil \leq \frac{B}{16A^2}$$

which is satisfied by

$$c \approx 8\frac{c_1}{c_2}A^2.$$

Thus, we have the following theorem:

THEOREM 6.2. *For the Fibonacci workload, any indexing scheme with the access overhead $A \leq \sqrt{B}/4$ must have redundancy*

$$r = \Omega\left(\frac{\log(n/B)}{\log A}\right).$$

The Fibonacci lattice is only one of many low-discrepancy [Matousek 1999], planar point sets we could have used. For example, we could have used the point set used by Chazelle [Chazelle 1990a], in his proof of a lower bound for range search in the pointer machine model. Matousek [Matousek 1999] discusses the discrepancy properties of the Fibonacci lattice, and many other point sets. However, none of these will improve the trade-off of Theorem 6.2 by more than a small constant factor.

7. SET WORKLOADS

We now turn our attention to the problem of indexing for arbitrary sets. An interesting workload is the λ-set workload $\mathcal{K}_{n,\lambda}$, whose instance is the set $\{1, \ldots, n\}$ and whose query set is the set of all λ-subsets of the instance. We show that these workloads are far worse than 2-dimensional queries.

Our Redundancy Theorem is applicable only when $\lambda > B/2$. In practice we are also interested in workloads with small values for λ. To analyze these workloads, we prove a corollary of the following famous theorem by Turán [Turán 1941; J.H. van Lint and Wilson 1992]:

THEOREM 7.1 TURÁN'S THEOREM. *If a simple graph of n vertices has more than*

$$\frac{(p-2)n^2}{2(p-1)} - \frac{r(p-1-r)}{2(p-1)} \quad (r = n \bmod p)$$

edges, then it contains a complete graph of p vertices (a p-clique).

For a given graph, an *independent set* is a subset of its vertices such that there is no edge between any pair of these vertices.

COROLLARY 1. *In a simple graph $G(V, E)$, with $|V| = n$, if*

$$|E| \leq \frac{n^2 - n(p-1)}{2(p-1)}$$

then G has an independent set of size p.

PROOF. Let $\tilde{G}(V, \tilde{E})$ be the graph with

$$\tilde{E} = \left\{ (v_1, v_2) \in \binom{V}{2} \;\middle|\; (v_1, v_2) \notin E \right\}$$

Then,

$$|\tilde{E}| = \binom{n}{2} - |E| > \frac{(p-2)n^2}{2(p-1)}$$

and thus by Turán's Theorem $\tilde{G}$ has a p-clique. The vertices of the clique form an independent set in G. □

We now show a lower bound for set workloads.

THEOREM 7.2. *For workload $\mathcal{K}_{n,\lambda}(I, \mathcal{Q})$, $B \geq \lambda$, any indexing scheme with redundancy*

$$r < \frac{n - \lambda + 1}{(\lambda - 1)(B - 1)}$$

has the worst possible access overhead $A = \lambda$.

PROOF. Construct graph $G(I, E)$ where $(x_1, x_2) \in E$ iff there exists a block containing both x_1 and x_2. This graph will have at most

$$r\frac{n}{B}\binom{B}{2} < \frac{n^2 - n(\lambda - 1)}{2(\lambda - 1)}$$

edges. By Corollary 1, it has an independent set of size λ. This set, taken as a query, will require exactly λ distinct blocks to be covered (by the construction of G). □

The last theorem states that $\mathcal{K}_{n,\lambda}$ requires space at least *quadratic* in n/B to avoid the worst possible access overhead. We show that within a factor of 2, the bound of the theorem is tight.

THEOREM 7.3. *For workload $\mathcal{K}_{n,\lambda}$ and $B \geq \lambda$ there exists an indexing scheme of access overhead $A = \lambda - 1$ and redundancy*

$$r = \frac{2n}{(\lambda - 1)B} - 1$$

PROOF. We arbitrarily partition the instance into $\lambda - 1$ sets of roughly equal size, $S_1, \ldots, S_{\lambda-1}$. For each set S_i, we will construct suitable blocks so that for any $x, x' \in S_i$ there is a single block containing both. Then, for every query Q, some elements x_1 and x_2 will belong to the same set S_i, and thus will be covered by a single block, and so $A(Q) \leq \lambda - 1$.

To construct blocks for set S_i, we arbitrarily partition the set S_i into $k = \frac{2n}{(\lambda-1)B}$ sets $t_j, j = 1, \ldots, k$ of size $B/2$ each. For each pair of these sets we construct a block containing their union. Thus, for any pair of elements of S_i, there exists a block containing both.

For each of the $\lambda - 1$ sets S_i we constructed $\binom{k}{2}$ blocks. The total number of blocks constructed thus is

$$(\lambda - 1)\binom{\frac{2n}{(\lambda-1)B}}{2} = \frac{n}{B}\left(\frac{2n}{(\lambda - 1)B} - 1\right)$$

which yields the required redundancy. □

8. CONCLUSIONS

We have presented a new framework for the modeling and study of indexing in external memory. Our cost model is minimalistic, in that it ignores important parameters of external memory and indexing. This is not by accident but rather by design. There exist more precise (and more complex) cost models that are more accurate in predicting space and/or I/O indexing costs, e.g., models that include the search aspects of indexing, or models that describe hard disk performance more accurately. In our view, however, a successful model is not one that represents reality faithfully, but rather one that manages to capture the essence of a facet of the real world in a way that allows for deeper study and understanding of this facet.

Having argued in favor of the minimalistic aspects of indexability, we should stress that we expect indexability results to often carry over to more detailed models straightforwardly, and also to the implementation domain. Recent results by Arge, Samoladas and Vitter [Arge et al. 1999] indicate that it may be possible to employ indexability techniques as subroutines in external data structures, as part of a systematic approach to the "externalization" of main memory data structures. Kornacker, Shah, and Hellerstein have developed an index analysis tool called amdb [Shah et al. 1999]; among its features is a test for unit redundancy indexability, which serves as a concrete performance target for index developers.

REFERENCES

AOKI, P. M. 1998. Generalizing "search" in generalized search trees (extended abstract). In *Proceedings of the Fourteenth International Conference on Data Engineering, February 23-27, 1998, Orlando, Florida, USA*. 380–389.

ARGE, L., SAMOLADAS, V., AND VITTER, J. 1999. On two-dimensional indexability and optimal range search indexing. In *Proc. 18th ACM Symp. Principles of Database Systems.* 346–357.

ARGE, L. AND VITTER, J. 1996. Optimal dynamic interval management in external memory. In *FOCS '96; 37th Annual Symposium on Foundations of Computer Science.* IEEE Computer Society, 560–569.

BARKOL, O. AND RABANI, Y. 2000. Tighter bounds for nearest neighbor search and related problems in the cell probe model. In *Proc. 32nd ACM Symp. Theory of Computation.* 388–396.

BAYER, R. AND MCCREIGHT, E. 1972. Organization and maintenance of large ordered indexes. *Acta Informatica 1*, 173–189.

BELUSSI, A. AND FALOUTSOS, C. 1995. Estimating the selectivity of spatial queries using the 'correlation' fractal dimension. In *Proc. 21st Intl. Conf. Very Large Databases.* 299–310.

CHAKRABARTI, A., CHAZELLE, B., GUM, B., AND LVOV, A. 1999. A lower bound on the complexity of approximate nearest-neighbor searching on the hamming cube. In *Proc. 31st ACM Symp. Theory of Computation.* 305–311.

CHAZELLE, B. 1990a. Lower bounds for orthogonal range searching, i: the reporting case. *Journal of the ACM 37,* 2 (Apr.), 200–212.

CHAZELLE, B. 1990b. Lower bounds for orthogonal range searching, ii: the arithmetic model. *Journal of the ACM 37,* 3 (June), 439–463.

CHAZELLE, B. 1995. Lower bounds for off-line range searching. In *Proc. 27th ACM Symp. Theory of Computation.* 733–740.

FALOUTSOS, C. AND KAMEL, I. 1994. Beyond uniformity and independence: Analysis of R- trees using the concept of fractal dimension. In *Proc. 13th ACM Symp. Principles of Database Systems.* 4–13.

FIAT, A. AND SHAMIR, A. 1989. How to find a battleship. *Networks 19,* 361–371.

FREDMAN, M. 1980. The inherent complexity of dynamic data structures which accomodate range queries. In *Proc. IEEE Symp. on Foundations of Comp. Sci.* 191–199.

FREDMAN, M. 1981. Lower bounds on the complexity of some optimal data structures. *SIAM Journal of Computing 10,* 1–10.

GAEDE, V. AND GÜNTHER, O. 1998. Multidimensional access methods. *ACM Computing Surveys 30,* 2 (June), 170–231.

GUTTMAN, A. 1984. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*. 47–57.

HELLERSTEIN, J., NAUGHTON, J., AND PFEFFER, A. 1995. Generalized search trees for database systems. In *Proc. 21st Intl. Conf. Very Large Databases.* 562–573.

J.H. VAN LINT AND WILSON, R. 1992. *A Course in Combinatorics.* Cambridge University Press.

JOHNSON, S. 1962. A new upper bound for error-correcting codes. *IEEE Trans. Information Theory 8,* 203–207.

KANELLAKIS, P., RAMASWAMY, S., VENGROFF, D., AND VITTER, J. S. 1993. Indexing for data models with constraints and classes. In *Proc. 12th ACM Symp. Principles of Database Systems.* 233–243.

KANELLAKIS, P. C., RAMASWAMY, S., VENGROFF, D. E., AND VITTER, J. S. 1996. Indexing for data models with constraints and classes. *Journal of Computer and System Sciences 52,* 3, 589–612.

KANTH, K. V. R., RAVADA, S., SHARMA, J., AND BANERJEE, J. 1999. Indexing medium-dimensionality data in oracle. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*. 521–522.

KORNACKER, M. 1999. High-performance extensible indexing. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*. 699–708.

KORNACKER, M., MOHAN, C., AND HELLERSTEIN, J. M. 1997. Concurrency and recovery in generalized search trees. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*. 62–72.

KOUTSOUPIAS, E. AND TAYLOR, D. 1998. Tight bounds for 2-dimensional indexing schemes. In *Proc. 17th ACM Symp. Principles of Database Systems.* 52–58.

KOUTSOUPIAS, E. AND TAYLOR, D. 1999. Indexing schemes for random points. In *Proc. 10th ACM-SIAM Symp. Discrete Algorithms.* 596–602.

MATOUSEK, J. 1999. *Geometric Discrepancy.* Springer.

MEHLHORN, K. 1984. *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry.* Springer-Verlag, EATCS Monographs on Theoretical Computer Science.

MILTERSEN, P., NISAN, N., SAFRA, S., AND WIDGERSON, A. 1995. On data structures and asymmetric communication complexity. In *Proc. 27th ACM Symp. Theory of Computation.* 103–111.

NIEVERGELT, J., HINTERBERGER, H., AND SEVCIK, K. 1984. The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems 9,* 1 (Mar.), 38–71.

NODINE, M. H., GOODRICH, M. T., AND VITTER, J. S. 1993. Blocking for external graph searching. In *Proc. 12th ACM Symp. Principles of Database Systems.* 222–232.

PAGEL, B.-U., SIX, H.-W., TOBEN, H., AND WIDMAYER, P. 1993. Towards an analysis of range query performance in spatial data structures. In *Proc. 12th ACM Symp. Principles of Database Systems.* 214–221.

RAMASWAMY, S. AND KANELLAKIS, P. C. 1995. OODB indexing by class-division. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*. 139–150.

RAMASWAMY, S. AND SUBRAMANIAN, S. 1994. Path caching: A technique for optimal external searching. In *Proc. 13th ACM Symp. Principles of Database Systems.* 25–35.

SALZBERG, B. AND TSOTRAS, V. 1999. Comparison of access methods for time-evolving data. *ACM Computing Surveys 31,* 2, 158–221.

SAMET, H. 1989. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, MA.

SHAH, M., KORNACKER, M., AND HELLERSTEIN, J. 1999. Amdb: A visual access method development tool. In *User Interfaces to Data Intensive Systems (UIDIS).* 130–140.

SMID, M. AND OVERMARS, M. 1990. Maintaining range trees in secondary memory. Part II: Lower bounds. *Acta Informatica 27*, 453–480.

SUBRAMANIAN, S. AND RAMASWAMY, S. 1995. The P-range tree: A new data structure for range searching in secondary memory. In *Proc. 6th ACM-SIAM Symp. Discrete Algorithms.* 378–387.

TURÁN, P. 1941. An extremal problem in graph theory (in hungarian). *Mat. Fiz. Lapok. 48,* 435–452.

VAIDYA, P. 1989. Space-time trade-offs for orthogonal range queries. *SIAM Journal of Computing 18,* 4 (Aug.), 748–758.

VENGROFF, D. AND VITTER, J. 1996. Efficient 3-D range searching in external memory. In *Proc. 28th ACM Symp. Theory of Computation.* 192–201.

WITTEN, I. H., MOFFAT, A., AND BELL, T. C. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images,* Second ed. Morgan Kaufmann.

YAO, A. 1981. Should tables be sorted? *Journal of the ACM 28,* 3, 615–628.

YAO, A. 1982. Space-time tradeoff for answering range queries. In *Proc. 14th ACM Symp. Theory of Computation.* 128–136.