A General Approach for Cache-Oblivious Range Reporting and Approximate Range Counting

Peyman Afshani*
MADALGO
Dept. of Computer Science
University of Aarhus
IT-Parken, Aabogade 34
DK-8200 Aarhus N
Denmark
peyman@madalgo.au.dk

Chris Hamilton†
Faculty of Computer Science
Dalhousie University
6050 University Avenue
Halifax, NS B3H 1W5
Canada
chamilton@cs.dal.ca

Norbert Zeh‡
Faculty of Computer Science
Dalhousie University
6050 University Avenue
Halifax, NS B3H 1W5
Canada
nzeh@cs.dal.ca

ABSTRACT

We present cache-oblivious solutions to two important variants of range searching: range reporting and approximate range counting. The main contribution of our paper is a general approach for constructing cache-oblivious data structures that provide relative $(1+\varepsilon)$-approximations for a general class of range counting queries. This class includes three-sided range counting, 3-d dominance counting, and 3-d halfspace range counting. Our technique allows us to obtain data structures that use linear space and answer queries in the optimal query bound of $O(\log_B(N/K))$ block transfers in the worst case, where K is the number of points in the query range. Using the same technique, we also obtain the first approximate 3-d halfspace range counting and 3-d dominance counting data structures with a *worst-case* query time of $O(\log(N/K))$ in internal memory.

An easy but important consequence of our main result is the existence of $O(N \log N)$-space cache-oblivious data structures with an optimal query bound of $O(\log_B N + K/B)$ block transfers for the reporting versions of the above problems. Using standard reductions, these data structures allow us to obtain the first cache-oblivious data structures that use near-linear space and achieve the optimal query bound for circular range reporting and K-nearest neighbour searching in the plane, as well as for orthogonal range reporting in three dimensions.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms

Algorithms, Design, Theory

Keywords

Memory hierarchies, cache-obliviousness, data structures, range searching

1. INTRODUCTION

Range searching is one of the most fundamental problems in computational geometry. Given a set S of N points in $\mathbb{R}^d$, the task is to preprocess S so that all points in a query region can be counted (*range counting*) or reported (*range reporting*) efficiently. *Approximate range counting* asks for an approximation of the number K of points in the query range that is no less than K and no greater than $(1+\varepsilon)K$.

Typical range searching problems are expressed in more specific terms depending on the shape of the query: simplices, halfspaces, circles, and axis-aligned boxes lead to *simplex range searching*, *halfspace range searching*, *circular range searching*, and *orthogonal range searching* problems, respectively. Two important special cases of orthogonal range searching have also been studied extensively: for *three-sided queries* in the plane, one side of the bounding box is fixed at infinity; for 3-d *dominance queries*, the "bottom-left" vertex of the box is the point $(-\infty, -\infty, -\infty)$.

In this paper, we propose a general approach based on shallow cuttings [33] that allows us to obtain cache-oblivious approximate range counting structures and cache-oblivious range reporting structures with the optimal query bound for a wide range of problems. We also obtain the first internal-memory structures for approximate 3-d halfspace range counting and approximate 3-d dominance counting that achieve the optimal query bound in the worst case.

For most of these problems, no non-trivial cache-oblivious approximate counting structures were known before. Cache-oblivious reporting structures with the same space and query bounds were known only for three-sided range reporting [5, 9, 13]. The structure of [9] can be seen as using some notion

*Part of this work was done while visiting Dalhousie University.

†Supported by a Killam Graduate Scholarship.

‡Supported in part by the Natural Sciences and Engineering Research Council of Canada, the Canadian Foundation for Innovation, and the Canada Research Chairs programme.

SCG'09, June 8–10, 2009, Aarhus, Denmark.

of shallow cuttings for three-sided range queries but uses a counting structure tailored specifically to orthogonal range queries. In contrast, our approach can be used to obtain approximate counting and reporting structures for any range searching problem with shallow cuttings, without the need for specialized techniques.

1.1 Model of Computation and Related Work

Most previous work on range searching has focused on *internal-memory* models of computation, such as the RAM model or the pointer machine model. An important characteristic of these models is that the cost to access a data item is independent of the memory location where it is stored.

The *input-output model* (or *I/O model*) [8] and the *cache-oblivious model* [21] are the two most widely accepted models that take non-uniform memory access costs into account. The I/O model has two levels of memory: a slow but conceptually unlimited *external memory* and a fast *internal memory* with capacity M. All computation has to happen on data in internal memory. The transfer of data between internal and external memory happens in blocks of B consecutive data items; the complexity of an algorithm is the number of such *block transfers* it performs.

The cache-oblivious model provides a simple framework for designing algorithms for multi-level memory hierarchies. In this model the algorithm is oblivious of the details of the memory hierarchy but is analyzed in the I/O model, assuming the block transfers necessary to bring the data accessed by the algorithm into memory are performed by an *offline optimal* paging algorithm, that is, one that performs the minimum number of block transfers for the memory access sequence of the algorithm. Since the algorithm is designed without reference to M or B, the analysis can be applied to *any* two consecutive levels of the memory hierarchy. In particular, if the analysis shows that the algorithm incurs an optimal number of block transfers with respect to two levels of the memory hierarchy, then it does so simultaneously at all levels of a multi-level memory hierarchy. See [21] for more details and for a justification for assuming an offline optimal paging algorithm.

Our discussion of previous work starts with a review of the most relevant work in internal memory and then moves on to results obtained in the I/O model and the cache-oblivious model.

Internal memory. In internal memory, linear-space structures with the optimal query bound of $O(\log N + K)$ are known for three-sided range reporting [34], 3-d dominance reporting [1, 30], and 3-d halfspace range reporting [3], as well as for a number of related problems; K denotes the number of points in the query range. Using standard reductions, the results for halfspace range reporting imply the same results for circular range reporting and 2-d K-nearest neighbour searching.

Exactly *counting* the number of points in a query range seems significantly harder than reporting if the query bound is to be independent of K. For three-sided range counting, Chazelle [19] obtained a linear-space structure with query time $O(\log N)$, which also immediately implies an $O(N \log N)$-space structure with query time $O(\log^2 N)$ for 3-d dominance counting. For exact halfspace range counting, Matoušek [31] obtained a linear-space structure with a query bound of $O(N^{2/3})$, and this is conjectured to be the best possible. As a result, much effort has been put into obtaining *approximate* halfspace range counting structures with polylogarithmic query bounds.

Aronov and Har-Peled [14] presented a general technique that can be used to construct an approximate range counting structure from a range emptiness structure; the obtained structure provides a correct approximation of the number of points in the query range with high probability. The cost of this transformation is an increase of the space bound by a factor of $\log N$ and an increase of the query bound by a factor of $\log N \log\log N$.

For halfspace range searching, linear-space range emptiness structures with $O(\log N)$ query time have been known for a while (see, e.g., [29]). Thus, the technique by Aronov and Har-Peled provides an $O(N \log N)$-space approximate halfspace range counting structure with $O(\log^2 N \log\log N)$ query time. Kaplan and Sharir [28] improved the query time by a $\log\log N$ factor using an interesting combinatorial lemma regarding the overlay of lower envelopes in a randomized incremental construction (see also [26, 27]). As Aronov and Har-Peled, they could guarantee correctness only with high-probability. Later, Aronov and Har-Peled showed in an updated version of their paper [15] that an $O(\log^2 N)$ query time can be obtained using $O(N \log N)$ space without applying the overlay lemma. Har-Peled and Sharir [24] showed that a worst-case query time of $O(\log N \log\log N)$ can be achieved using $O(N \log^{O(1)} N)$ space. This was improved by Afshani and Chan [2], who presented a linear-space structure with the same worst-case query bound, as well as another linear-space structure that uses the overlay lemma to achieve the optimal query time of $O(\log(N/K))$ in the expected case.

The fact that the overlay lemma is a crucial component of Afshani and Chan's optimal data structure has a number of limiting implications: the method does not generalize to other problems, unless a similar overlay lemma is proved for each such problem; a non-trivial modification of the overlay lemma would be required to use it in models such as the I/O or cache-oblivious model; and, finally, it cannot be used to obtain a worst-case query bound. The other methods discussed above have similar shortcomings in that they are tailored to internal-memory models or to specific problems. For example, many of the $\log N$-factors in the above complexity bounds are the result of applying Chernoff-type inequalities and cannot easily be reduced to $\log_B N$ in the I/O model or the cache-oblivious model.

I/O model and cache-oblivious model. In the I/O model, much work has focused on orthogonal range reporting. A number of linear-space structures have been proposed that achieve a query bound of $O(\sqrt{N/B} + K/B)$ in 2-d and $O((N/B)^{1-1/d} + K/B)$ in d dimensions [11, 22, 23, 25, 36, 38]. In [12], Arge et al. showed how to achieve a query bound of $O(\log_B N + K/B)$ for three-sided range reporting in the plane using linear space. They also showed that $\Theta(N \log_B N / \log_B \log_B N)$ space is sufficient and necessary to achieve a query bound of $O(\log_B N + K/B)$ block transfers for four-sided range reporting. For 3-d dominance reporting, a data structure by Afshani [1] achieves the optimal query bound of $O(\log_B N + K/B)$ using linear space. This structure also yields an orthogonal range reporting data structure that uses $O(N \log^3 N)$ space and achieves the same query bound. In [6], Agarwal et al. provided a number of results

Query type	Space	Query bound	References/Notes
3-d halfspace	$N \log N$	$\log^2 N \log\log N$	Aronov, Har-Peled [14] (MC)
(previous)	$N \log N$	$\log^2 N$	Aronov, Har-Peled [15] (MC)
	$N \log N$	$\log^2 N$	Kaplan, Sharir [26, 27, 28] (MC)
	$N \log^{O(1)} N$	$\log N \log\log N$	Har-Peled, Sharir [24] worst case
	N	$\log N \log\log N$	Afshani, Chan [2] worst case
	N	$\log(N/K)$	Afshani, Chan [2] (LV)
3-d halfspace	N	$\log(N/K)$	worst case
(new)	N	$\log_B(N/K)$	worst case, I/O & cache-oblivious model
3-d dominance	N	$\log_B(N/K)$	worst case, I/O & cache-oblivious model
(new)			

MC = Monte Carlo (result is obtained with high probability)
LV = Las Vegas (query bound is expected)

Table 1: A comparison of our results on approximate range counting with previous work.

on halfspace range reporting in three and higher dimensions. The result most relevant to our work is an $O(N \log N)$-space structure with an optimal query bound of $O(\log_B N + K/B)$ block transfers. Recently, Afshani and Chan [3] improved on this result by providing an $O(N \log^* N)$-space structure with query bound $O(\log_B N + K/B)$.

Much less is known about cache-oblivious range searching structures. Orthogonal range reporting queries in $\mathbb{R}^d$ can be answered using $O((N/B)^{1-1/d} + K/B)$ block transfers [5, 10]. Cache-oblivious range reporting structures with query bound $O(\log_B N \log\log N + K/B)$ and using $O(N \log N)$ space are easily obtained for three-sided, halfspace, and dominance queries using existing techniques. Thus, the interesting questions are whether cache-oblivious structures for these problems exist that achieve the optimal $O(\log_B N + K/B)$ query bound and how much space is necessary to achieve this bound.

For three-sided queries, structures with the optimal query bound of $O(\log_B N + K/B)$ and using $O(N \log N)$ space were proposed in [5, 9, 13]. The structure by Arge and Zeh [13] was obtained using a standard reduction to 2-d dominance queries, for which the paper presented a linear-space structure with optimal query bound. The structure by Arge et al. [9] can be seen as being based on some notion of shallow cuttings for three-sided range searching combined with a specialized 2-d dominance counting structure. For the remaining problems, such as 3-d dominance reporting and 3-d halfspace range reporting, as well as their approximate counting versions, no non-trivial results were known in the cache-oblivious model.

In [4], we make progress towards answering how much space is required to achieve the optimal query bound. In particular, we show that a cache-oblivious data structure that achieves the optimal query bound for three-sided range reporting, as well as for 3-d halfspace range reporting and 3-d dominance reporting, must use $\Omega(N(\log\log N)^{\varepsilon})$ space.

1.2 New Results

In this paper, we obtain the first cache-oblivious data structures with the optimal query bound of $O(\log_B N + K/B)$ block transfers for 3-d halfspace range reporting, 3-d dominance reporting and, as a consequence, for circular range reporting, 2-d K-nearest neighbour searching, and 3-d orthogonal range reporting. All our structures, except the 3-d orthogonal range reporting structure, use $O(N \log N)$ space. Using a standard transformation, our 3-d dominance structure also provides a new $O(N \log N)$-space structure with optimal query bound for three-sided range reporting, thereby matching the previous results obtained in [5, 9, 13]. These results are fairly easy to obtain using standard constructions based on shallow cuttings, once the output size K of a query can be efficiently determined or at least approximated.

Our main technical contribution, therefore, is a general framework for constructing cache-oblivious structures for the approximate counting versions of the above problems. These structures use linear space and provide guaranteed $(1+\varepsilon)$-approximate answers in the optimal $O(\log_B(N/K))$ query bound in the worst case. This is in contrast to previous results even in internal memory, where the optimal query bound was not achieved in the worst case before, even using superlinear space. The only previous structure with the optimal query bound, by Afshani and Chan [3], achieves this bound only in the expected case. Thus, our construction also provides new worst-case optimal structures for approximate halfspace range counting and approximate dominance counting in the pointer machine model.[1] Table 1 compares our results with previous work.

The main tool used in our data structures is shallow cuttings, which can be obtained for a general class of problems, albeit using a randomized construction [7]. The use of shallow cuttings in problems related to range searching is by now fairly standard; however, our results are obtained by combining them with other standard techniques in a novel way that proves powerful enough to achieve the above series of new results.

2. PRELIMINARIES

We provide a framework for constructing an approximate range counting structure for any range searching problem which, through application of duality or other techniques, can be translated into the following type of "aboveness reporting problem". Let $\mathcal{F}$ be a collection of continuous and

[1]It is easy to verify that we use only operations available on a pointer machine equipped with the necessary algebraic operations to compute intersections of curves and determine the side of a curve a point is on.

totally defined algebraic functions $f : \mathbb{R}^2 \to \mathbb{R}$ of constant degree. Each such function defines a continuous surface in $\mathbb{R}^3$ consisting of points $(x, y, f(x, y))$, and we do not distinguish between a function and the surface it defines. We say that function f *passes below* point $q = (x_q, y_q, z_q)$ if $f(x_q, y_q) < z_q$. Our goal is to preprocess $\mathcal{F}$ so that, given any query point q, we can efficiently count (approximately) how many functions in $\mathcal{F}$ pass below q. Since we assume that this counting query is an alternative representation of a range counting query, we refer to counting the functions that pass below a query point as range counting throughout this paper.

The *arrangement* of $\mathcal{F}$ is a subdivision of $\mathbb{R}^3$ into 0-, 1-, 2-, and 3-*cells*, which are maximal connected sets of points contained in 3, 2, 1, or 0 functions in $\mathcal{F}$, respectively. We define the *level* of a point q to be the number of functions in $\mathcal{F}$ that pass below q. The *k-level* or $(\leq k)$*-level* of $\mathcal{F}$ is the closure of the set of points in $\mathbb{R}^3$ that have level k or at most k, respectively. The 0-level of $\mathcal{F}$ is also known as the *lower envelope* of $\mathcal{F}$. Any k- or $(\leq k)$-level is a collection $\mathcal{C}$ of cells. Its *complexity* $|\mathcal{C}|$ is defined as the number of cells in $\mathcal{C}$.

A *shallow cutting for the* $(\leq k)$*-level* of $\mathcal{F}$ is a collection $\mathcal{C}$ of disjoint cells that cover the $(\leq k)$-level of $\mathcal{F}$ and have the property that every cell $C \in \mathcal{C}$ intersects $O(k)$ functions in $\mathcal{F}$; the set of these functions is the *conflict list* Δ_C of C. W.l.o.g., we can assume that every cell in $\mathcal{C}$ intersects the $(\leq k)$-level of $\mathcal{F}$ (otherwise, we can obtain a smaller cutting for the $(\leq k)$-level by removing all cells from $\mathcal{C}$ that do not intersect this level). Under this assumption, we can add all functions that pass below a cell $C \in \mathcal{C}$ to Δ_C without increasing the size of Δ_C by more than k. This ensures that, for every point $p \in C$, all the functions that pass below p are included in Δ_C.

For our approximate range counting framework to be applicable, the set of functions, $\mathcal{F}$, has to satisfy the following conditions:

(i) For every k, there exists a shallow cutting $\mathcal{C}$ for the $(\leq k)$-level of $\mathcal{F}$ consisting of $O(|\mathcal{F}|/k)$ cells.

(ii) Given a query point q and a shallow cutting $\mathcal{C}$ for the $(\leq k)$-level of $\mathcal{F}$, there exists a cache-oblivious data structure that finds a cell in $\mathcal{C}$ that contains q or reports that no such cell exists. We denote this data structure as the *point location structure* $\mathcal{L}(\mathcal{C})$ of $\mathcal{C}$ and require that it uses $O(|\mathcal{C}|)$ space and supports queries using $O(\log_B |\mathcal{C}|)$ block transfers.

(iii) Consider the 2-d arrangement $\mathcal{A}$ formed by projecting shallow cuttings for a number of $(\leq k)$-levels onto the xy-plane. Then there exists a cache-oblivious point location data structure for $\mathcal{A}$ with $O(\log_B |\mathcal{A}|)$ query bound and with space complexity polynomial in $|\mathcal{A}|$.

By using results by Agarwal et al. [7], it is possible to replace (i) with a weaker condition that implies (i):

(i′) The lower envelope of every subset $\mathcal{F}' \subset \mathcal{F}$ has complexity $O(|\mathcal{F}'|)$.

For the problems we are interested in—3-d halfspace range searching and 3-d dominance searching—shallow cuttings satisfying condition (i) exist [1, 33]. In addition, for these problems, the functions in $\mathcal{F}$ can be decomposed into linear functions, which implies that condition (iii) can be satisfied using the cache-oblivious planar point location structure by Bender et al. [16]; condition (ii) can be satisfied using the same structure through projection of the cells into the xy-plane [1, 18]. We discuss this in more detail in Section 3.4.

3. APPROXIMATE RANGE COUNTING

This section presents the main result of our paper: a general framework for constructing cache-oblivious approximate range counting structures for any range searching problem that satisfies the conditions discussed in Section 2. The following theorem states this precisely.

THEOREM 1. *For a set $\mathcal{F}$ of N functions satisfying conditions (i)–(iii), there exists a cache-oblivious data structure that uses linear space and supports $(1+\varepsilon)$-approximate range counting queries using $O(\log_B(N/K))$ block transfers in the worst case, where K is the actual value of the count.*

Throughout this paper, we use q to refer to a particular query point, and K to denote the number of functions in $\mathcal{F}$ that pass below q. Our goal is to compute a number K' that satisfies $K \leq K' \leq (1+\varepsilon)K$. The first step towards proving Theorem 1 is to show that the difficult part of the problem is to obtain *any* constant-factor approximation of K.

LEMMA 1. *Consider a set $\mathcal{F}$ of N functions satisfying conditions (i)–(iii), and assume there exists a linear-space cache-oblivious data structure $\mathcal{D}$ that supports c-approximate range counting queries over $\mathcal{F}$ using $O(\log_B(N/K))$ block transfers, where c is an arbitrary constant and K is the actual value of the count. Then there exists a cache-oblivious data structure that uses linear space and supports $(1+\varepsilon)$-approximate range counting queries using $O(\log_B(N/K))$ block transfers.*

PROOF. To obtain a linear-space structure that supports $(1+\varepsilon)$-approximate range counting queries over $\mathcal{F}$, we use $\mathcal{D}$, as well as shallow cuttings $\mathcal{C}_i$, for $0 \leq i \leq \lceil \log N \rceil$; $\mathcal{C}_i$ is a shallow cutting for the $(\leq 2^i)$-level of $\mathcal{F}$. Each cutting $\mathcal{C}_i$ is represented using its point location structure $\mathcal{L}(\mathcal{C}_i)$. Every cell $C \in \mathcal{C}_i$ stores a constant-size list $\Delta'_C \subseteq \Delta_C$ such that, for any query point $q \in C$ and a sufficiently small constant δ, an approximation of K to within an additive error of $\delta|\Delta_C|$ can be obtained from the level of q in Δ'_C. Such a constant-size sublist of Δ_C can be obtained using techniques in VC-dimension: the VC-dimension of a set system defined by a set of algebraic functions of constant degree is constant (see, e.g., [32]), which implies that there is a subset Δ'_C of Δ_C as stated above whose size depends only on δ and is thus constant [39].

The size of the data structure is clearly linear: $\mathcal{D}$ uses linear space; the total size of all shallow cuttings $\mathcal{C}_i$ is linear; and, hence, their point location structures $\mathcal{L}(\mathcal{C}_i)$ use linear space, and storing a constant-size list for each cell in each cutting $\mathcal{C}_i$ also uses linear space.

Now consider a query point q. We use $\mathcal{D}$ to obtain a constant-factor approximation K' of K. The shallow cutting $\mathcal{C}_i$ with $i = \lceil \log K' \rceil$ contains q, and we can use $\mathcal{L}(\mathcal{C}_i)$ to find the cell $C \in \mathcal{C}_i$ that contains q. Since the size of Δ_C is $O(K)$, it now suffices to approximate K to within an additive error of $\delta|\Delta_C|$, for a sufficiently small δ, and we can obtain such an approximation by computing q's level in Δ'_C, which we do by comparing q against every function in Δ'_C. The query on $\mathcal{D}$

uses $O(\log_B(N/K))$ block transfers; the query on $\mathcal{L}(\mathcal{C}_i)$ uses $O(\log_B|\mathcal{C}_i|) = O(\log_B(N/K))$ block transfers; and scanning the constant-size list Δ'_C adds $O(1)$ block transfers to the query cost. This completes the proof. □

By Lemma 1, it suffices to obtain a constant-factor approximate range counting structure for $\mathcal{F}$. We split its construction into three parts. We say that a query q is *polynomial* or *polylogarithmic* if $K \geq N^\alpha$ or $K \geq \log^\alpha N$, respectively, for some constant $\alpha > 0$. The first step is to obtain a structure for polynomial queries that uses *sublinear* space and achieves the query bound stated in Lemma 1. Using this structure, we can obtain a structure for polylogarithmic queries. By applying this construction a second time, the structure for polylogarithmic queries can be made to support arbitrary approximate range counting queries. Next we discuss these three steps in detail.

3.1 A Structure for Polynomial Queries

In this section, we prove that we can achieve the desired query bound for polynomial queries using *sublinear* space, as stated in the next lemma. This is crucial to ensure that the space bound of our structure for polylogarithmic queries is linear.

LEMMA 2. *For a set $\mathcal{F}$ of N functions satisfying conditions (i)–(iii), there exists a cache-oblivious data structure that uses $O(\sqrt{N})$ space and supports approximate range counting for polynomial queries using $O(\log_B(N/K))$ block transfers in the worst case, where K is the actual value of the count.*

Let c be an integer constant such that the conflict list of any cell C in a shallow cutting for the $(\leq k)$-level of $\mathcal{F}$ has size less than $2^c k$. By the definition of a shallow cutting, such a constant exits. To obtain a structure as in the lemma, we construct a shallow cutting $\mathcal{C}_i$ for the $(\leq N/2^i)$-level of $\mathcal{F}$, for each $1 \leq i \leq c+2\delta\log N$, where δ is a constant to be chosen later. Let $\mathcal{A}_i$ be the 2-d arrangement obtained by projecting the cells of $\mathcal{C}_i$ into the xy-plane. Next we construct arrangements $\mathcal{A}_0^*, \mathcal{A}_1^*, \ldots, \mathcal{A}_t^*$, where $t = \lfloor\log(2\delta\log N)\rfloor$. Arrangement $\mathcal{A}_i^*$ is obtained by overlaying arrangements $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{c+2^i}$. Since $|\mathcal{C}_j| = O(2^j)$, the total number of cells in cuttings $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{c+2^i}$ is $O(2^{2^i})$. Since the boundary of each cell in each cutting $\mathcal{C}_j$ is an algebraic curve of constant degree, this implies that $\mathcal{A}_i^*$ has size polynomial in 2^{2^i}. In particular, the size of the largest arrangement $\mathcal{A}_t^*$—and, thus, the total size of all arrangements $\mathcal{A}_0^*, \mathcal{A}_1^*, \ldots, \mathcal{A}_t^*$— is polynomial in $2^{2^t} = N^{2\delta}$.

For each arrangement $\mathcal{A}_i^*$, we store a point location structure as in condition (iii). These structures are stored consecutively in memory, in order of increasing i. For each face f of each arrangement $\mathcal{A}_i^*$, we store the list of cells of cuttings $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{c+2^i}$ that project onto f. If more than one cell from a cutting $\mathcal{C}_j$ projects onto f, we store only the highest one. Hence, each face of $\mathcal{A}_i^*$ stores a list of length $c + 2^i = O(2^i)$.

The point location structure for each arrangement $\mathcal{A}_i^*$ has size polynomial in $|\mathcal{A}_i^*|$, and the list associated with each cell of $\mathcal{A}_i^*$ has size $O(2^i) = O(\log N)$. Thus, by the bound on the total size of these arrangements obtained above, the resulting data structure uses space polynomial in $N^{2\delta}$, and we can choose δ small enough to bound the space used by the structure by $O(\sqrt{N})$.

To answer a query using the constructed data structure, we start from $i = 0$ and find the face f of $\mathcal{A}_i^*$ that contains the projection of the query point q. Next, we inspect the $c + 2^i$ cells that were projected onto f and test for each whether it contains q. If all inspected cells contain q, we either increase i by one and iterate this procedure or, if $i = t$, report that K is too small, and the query fails. If at least one of the tried cells does not contain q, we output $K' = |\Delta_C|$, where C is the "smallest" cell that projected onto f and contains q; more precisely, C belongs to a shallow cutting $\mathcal{C}_j$ such that $q \in \mathcal{C}_j$ and $q \notin \mathcal{C}_{j+1}$.

Now, as long as $K \geq N^{1-\delta} \geq N/2^{2^{\lfloor\log(2\delta\log N)\rfloor}} = N/2^{2^t}$, the choice of constant c implies that $q \notin \mathcal{C}_{c+2^t}$, and the query does not fail. Moreover, we have $N/2^{j+1} \leq K \leq K' = |\Delta_C| = O(N/2^j)$ in this case, that is, the value K' we report is a constant-factor approximation of K.

To bound the cost of a successful query, we observe that $i \leq \lceil\log j\rceil$ when the query terminates. We can assume that this is an equality, as the cost of the query is lower if the query terminates earlier. The combined size of the first $i_0 := \log\log B - a$ arrangements $\mathcal{A}_i^*$, for some constant a, is $O(B)$. Thus, these structures can be queried using $O(1)$ block transfers, and the query cost is $O(1)$ when $i \leq i_0$. For $i > i_0$, we observe that $2^{i-1} < j \leq 2^i$, which implies that $2^{2^{i-1}} < 2^j \leq 2^{2^i}$ and, thus, $2^{2^i} = O((N/K)^2)$. The query cost for the case $i > i_0$ is therefore bounded by

$$\sum_{i'=i_0}^{i} O(\log_B 2^{2^{i'}} + 2^{i'}/B) = O(\log_B 2^{2^i} + 2^i/B)$$
$$= O(\log_B(N/K)).$$

For an unsuccessful query, a similar analysis shows that the query fails after $O(\log_B N^\delta)$ block transfers, which is $O(\log_B(N/K))$, as $K < N^{1-\delta}$ in this case.

3.2 A Structure for Polylogarithmic Queries

In this section, we present the second building block of our final structure, a linear-space structure for answering polylogarithmic approximate range counting queries, as summarized in the following lemma. Note that the query bound of our structure is $O(\log_B N)$, not $O(\log_B(N/K))$. This is sufficient for the purposes of our final structure discussed in Section 3.3.

LEMMA 3. *For a set $\mathcal{F}$ of N functions that satisfy conditions (i)–(iii), there exists a cache-oblivious data structure that uses $O(N)$ space and supports approximate range counting for polylogarithmic queries using $O(\log_B N)$ block transfers in the worst case.*

We prove Lemma 3 by constructing a structure that applies Lemma 2 recursively. We start with a shallow cutting $\mathcal{C}$ for the $(\leq N^{1-\delta})$-level of $\mathcal{F}$. In the memory layout, we represent $\mathcal{F}$ using the structure from Lemma 2, followed by the point location structure $\mathcal{L}(\mathcal{C})$ for $\mathcal{C}$, which in turn is followed by structures representing the cells of $\mathcal{C}$. Each such structure representing a cell $C \in \mathcal{C}$ is constructed by recursively applying this construction to Δ_C. The recursion stops as soon as we obtain conflict lists of size at most $\log^\tau N$, for some constant τ.

To answer a query with a query point q, we use $\mathcal{L}(\mathcal{C})$ to decide whether q is contained in $\mathcal{C}$ and, if so, determine the cell $C \in \mathcal{C}$ that contains q. If $q \in \mathcal{C}$, we recurse on

the structure representing Δ_C or report a failure if we are already at the last level of recursion in the structure. If $q \notin \mathcal{C}$, then q is a polynomial query and can be answered using the structure from Lemma 2.

Correctness. First assume that the query does not fail. If q belongs to a cell C of $\mathcal{C}$, then Δ_C contains all functions in $\mathcal{F}$ that pass below q, and an inductive argument shows that recursing on Δ_C produces the desired approximation of K. If $q \notin \mathcal{C}$, then q is a polynomial query, and Lemma 2 shows that we obtain the desired approximation of K in this case as well.

If, on the other hand, the query fails, this means that q is contained in the shallow cutting used at the last level of recursion. Since each cell of this cutting has a conflict list of size at most $\log^\tau N$, this means that $K \leq \log^\tau N$. Thus, for $K > \log^\tau N$, the query procedure does not fail.

Analysis. It remains to analyze the space and query bounds of our structure. Let $S(N)$ be its space complexity for a set of N functions. Since $\mathcal{C}$ is a shallow cutting for the $(\leq N^{1-\delta})$-level of $\mathcal{F}$, $\mathcal{C}$ has $O(N^\delta)$ cells, each with a conflict list of size $O(N^{1-\delta})$. Since we do not recurse on conflict lists of size $\log^\tau N$ or less, this implies that $S(N)$ is bounded by the following recurrence.

$$S(x) \leq \begin{cases} ax^\delta S(x^{1-\delta}) + O(\sqrt{x}) & x > \log^\tau N \\ O(1) & x \leq \log^\tau N \end{cases},$$

for an appropriate constant $a > 0$. After i steps of recursion applied to $S(N)$, we have

$$S(N) \leq a^i N^{1-(1-\delta)^i} S(N^{(1-\delta)^i}) + O(a^i N^{1-(1-\delta)^i/2}).$$

For $i = \log_{(1-\delta)}(\log\log^\tau N / \log N) = O((\log\log N)/\delta)$, we obtain

$$S(N) = O\left(a^{O((\log\log N)/\delta)} \frac{N}{\log^\tau N} S(\log^\tau N) + a^{O((\log\log N)/\delta)} \frac{N}{\log^{\tau/2} N}\right).$$

By choosing τ large enough, we obtain $S(N) = O(N)$ because $S(\log^\tau N) = O(1)$. The query bound obeys the recurrence

$$Q(N) = \begin{cases} Q(N^{1-\delta}) + O(\log_B N) & N > B \\ O(1) & N \leq B \end{cases},$$

since $S(B) = O(B)$ and, hence, the entire recursive structure for a conflict list of size B fits in $O(1)$ blocks. This recurrence is easily seen to yield $Q(N) = O(\log_B N)$. This completes the proof.

3.3 The Final Structure

In this section, we combine Lemmas 2 and 3 to obtain a linear-space structure that supports constant-factor approximate range queries with the optimal query bound, for any query range. By Lemma 1, this implies Theorem 1.

LEMMA 4. *For a set $\mathcal{F}$ of N functions satisfying conditions (i)–(iii), there exists a cache-oblivious data structure that uses linear space and supports approximate range counting queries using $O(\log_B(N/K))$ block transfers in the worst case.*

PROOF. Our final structure consists of the following components. We compute shallow cuttings $\mathcal{C}_1$ and $\mathcal{C}_2$ for the $(\leq \log^\tau N)$-level and for the $(\leq \log N)$-level of $\mathcal{F}$, respectively. We represent $\mathcal{F}$ using two structures, one for polynomial queries and one for polylogarithmic queries, and we store point location structures $\mathcal{L}(\mathcal{C}_1)$ and $\mathcal{L}(\mathcal{C}_2)$ for the two shallow cuttings $\mathcal{C}_1$ and $\mathcal{C}_2$. For every cell $C \in \mathcal{C}_1$, we represent Δ_C using a structure for polylogarithmic queries. For every cell $C \in \mathcal{C}_2$, we simply store the list of functions in Δ_C contiguously.

By Lemmas 2 and 3, the structures representing $\mathcal{F}$ use $O(N)$ space in total. The structure representing each cell C in $\mathcal{C}_1$ has size $O(|\Delta_C|) = O(\log^\tau N)$, and the number of cells in $\mathcal{C}_1$ is $O(N/\log^\tau N)$. Thus, the representation of $\mathcal{C}_1$ uses linear space. Similarly, the conflict list of each cell C in $\mathcal{C}_2$ can be stored in $O(\log N)$ space, and there are $O(N/\log N)$ such cells. Hence, the space consumption of the entire structure is linear.

To answer a query, we query the structure for polynomial queries on $\mathcal{F}$ and report the answer if this query succeeds. This takes $O(\log_B(N/K))$ block transfers, by Lemma 2. If the query fails, we have $K < N^{1-\delta}$. Hence, $\log_B N = O(\log_B(N/K))$, and it suffices to achieve a query bound of $O(\log_B N)$ in this case. To do so, we first query $\mathcal{L}(\mathcal{C}_2)$ to decide whether $q \in \mathcal{C}_2$. This takes $O(\log_B N)$ block transfers. If $q \in \mathcal{C}_2$ (which means $K = O(\log N)$), the query also returns a cell $C \in \mathcal{C}_2$ such that $q \in C$, and we spend another $O((\log N)/B)$ block transfers to scan Δ_C and determine K exactly. If $q \notin \mathcal{C}_2$, we have $\log N < K$. We query $\mathcal{L}(\mathcal{C}_1)$ to decide whether $q \in \mathcal{C}_1$. If so, then $\log N < K = O(\log^\tau N)$ and the query returns a cell $C \in \mathcal{C}_1$ that contains q. We query the polylogarithmic structure representing Δ_C, which takes $O(\log_B N)$ block transfers and produces a constant-factor approximation of K because q is polylogarithmic for Δ_C. Finally, if $q \notin \mathcal{C}_1$, then q is polylogarithmic for $\mathcal{F}$, and we can use the polylogarithmic structure of $\mathcal{F}$ to obtain a constant-factor approximation of K using $O(\log_B N)$ block transfers. Since the query bound is $O(\log_B(N/K))$ in all cases, this finishes the proof of the lemma. □

3.4 Applications

By verifying that halfspace range counting and dominance counting satisfy conditions (i)–(iii), we obtain the following result as an immediate consequence of Theorem 1.

THEOREM 2. *There exist cache-oblivious data structures that use linear space and respectively support approximate 3-d halfspace range counting queries and approximate 3-d dominance counting queries using $O(\log_B(N/K))$ block transfers and $O(\log(N/K))$ time in the worst case.*

PROOF. The dual problem to 3-d halfspace range counting is to count all the planes in a set $\mathcal{F}$ that pass below a query point q. It is well known that the lower envelope of a set of N linear functions corresponds to the convex hull of the points dual to the functions and, thus, has worst-case complexity $O(N)$. Therefore, the set $\mathcal{F}$ of planes satisfies condition (i′) and, hence, condition (i). In fact, halfspace range searching was the problem used by Matoušek to introduce the notion of shallow cuttings [33]. A structure satisfying condition (ii) can be obtained by projecting the cells of the given shallow cutting into the plane and preprocessing the resulting planar straight-line subdivision for

point location queries (see [3, 17]). Using the linear-space cache-oblivious planar point location structure by Bender et al. [16], point location queries on this arrangement can be answered using $O(\log_B N)$ block transfers. The same structure can be used to satisfy condition (iii).

For dominance reporting, we can represent every input point p using a range $\overline{p}$ containing all points that dominate p. The boundary of this range is not a totally defined function. However, a small perturbation turns the boundary of $\overline{p}$ into a totally defined function composed of three linear functions. This allows us to phrase a dominance query with a query point q as identifying all such boundary functions that pass below q. Thus, our framework can be applied to dominance reporting as well, if we can verify conditions (i)–(iii). It is not difficult to show, however, that the lower envelope of this set of functions has linear complexity (see, e.g., [1]). Thus, conditions (i′) and (i) are satisfied once again. Furthermore, as with halfspace queries, conditions (ii) and (iii) reduce to point location in a planar straight-line subdivision [1] and, hence, can be satisfied using the point location structure by Bender et al. □

4. RANGE REPORTING

We can use the approximate range counting structure provided by Theorem 1 as a building block to quite easily obtain a cache-oblivious data structure that answers range reporting queries for any problem that fits in our framework. This data structure uses $O(N \log N)$ space.

Given a set $\mathcal{F}$ of N functions satisfying conditions (i)–(iii), such a structure can be obtained as follows. For $0 \leq i \leq \log N$, let $\mathcal{C}_i$ be a shallow cutting for the $(\leq 2^i)$-level of $\mathcal{F}$. For each cell $C \in \mathcal{C}_i$, we store the conflict list Δ_C contiguously. Since $\mathcal{C}_i$ contains $O(N/2^i)$ cells and each cell has a conflict list of size $O(2^i)$, the representation of each cutting $\mathcal{C}_i$ uses linear space. As there are $\log N$ such cuttings, the total space consumption is $O(N \log N)$. Finally, we add an approximate counting structure for $\mathcal{F}$ as in Theorem 1, as well as a point location structure $\mathcal{L}(\mathcal{C}_i)$ for each cutting $\mathcal{C}_i$. This adds only $O(N)$ to the total space bound.

To answer a range reporting query with query point q, we query the counting structure to obtain a 2-approximation K' of K. This incurs $O(\log_B N)$ block transfers. Next we use another $O(\log_B N)$ block transfers to query $\mathcal{L}(\mathcal{C}_i)$, for $i = \lceil \log K' \rceil$, and determine the cell $C \in \mathcal{C}_i$ that contains point q. Finally, we scan the conflict list Δ_C and output all functions in Δ_C that pass below q. This incurs another $O(1 + |\Delta_C|/B) = O(1 + K/B)$ block transfers. The total query cost is thus $O(\log_B N + K/B)$, and we obtain the following theorem.

THEOREM 3. *For a given set $\mathcal{F}$ of N functions satisfying conditions (i)–(iii), there exists a cache-oblivious data structure that uses $O(N \log N)$ space and supports range reporting queries using $O(\log_B N + K/B)$ block transfers, where K is the output size of the query.*

Following the discussion in Section 3.4, this immediately implies the following corollary.

COROLLARY 1. *There exist cache-oblivious data structures that use $O(N \log N)$ space and support 3-d dominance reporting and 3-d halfspace range reporting queries using $O(\log_B N + K/B)$ block transfers.*

Remarks. The idea of building a hierarchy of $O(\log N)$ shallow cuttings is a standard technique that has been used many times in the past (see, e.g., [18, 37]). Thus, the results in this section are easy consequences of our main result (Theorem 1). However, no cache-oblivious 3-d halfspace range reporting or 3-d dominance reporting structures matching the bounds in Corollary 1 were known before. This further underlines the importance of Theorem 1.

Using equally standard techniques (see, e.g., [35]), we can obtain further results on three-sided range reporting, circular range reporting, 2-d K-nearest neighbour searching, and 3-d orthogonal range reporting. Again, except for three-sided range reporting, similar results were not known in the cache-oblivious model before. While circular range reporting queries directly translate into 3-d halfspace range reporting queries, K-nearest neighbour queries do not; a K-nearest neighbour query is equivalent to reporting the K lowest planes stabbed by a vertical line ℓ. It is easily verified, however, that the hierarchy of cuttings $\mathcal{C}_i$ above can be used to find these planes: we identify the cutting $\mathcal{C}_i$ with $i = \lceil \log K \rceil$ and use $\mathcal{L}(\mathcal{C}_i)$ (which is a planar point location structure on the xy-projection of $\mathcal{C}_i$) to find the cell $C \in \mathcal{C}_i$ stabbed by ℓ. Then we apply a linear-time selection algorithm (e.g., see [20, Chapter 9]) to Δ_C to find the K lowest planes in Δ_C stabbed by ℓ. This takes $O(|\Delta_C|/B) = O(K/B)$ block transfers.

COROLLARY 2. *There exist cache-oblivious data structures that use $O(N \log N)$ space and achieve the optimal query bound of $O(\log_B N + K/B)$ block transfers for three-sided and circular range reporting, and for 2-d K-nearest neighbour searching.*

COROLLARY 3. *There exists a cache-oblivious 3-d range reporting data structure that uses $O(N \log^4 N)$ space and supports queries using $O(\log_B N + K/B)$ block transfers.*

5. CONCLUSIONS

In this paper, we have provided a general framework for constructing cache-oblivious data structures for approximate range counting and exact range reporting for range searching problems that have shallow cuttings. This includes three-sided range searching, for which matching results were obtained before using different techniques, as well as 3-d dominance searching and 3-d halfspace range searching, for which no such cache-oblivious structures were known before.

The obtained counting structures use linear space, while the reporting structures use $O(N \log N)$ space, which is a $\log N$ factor away from the space needed to obtain equivalent query complexities in internal memory or in the I/O model. However, as we show in [4], it is in fact impossible to achieve the optimal query bound of $O(\log_B N + K/B)$ for these range reporting problems using linear space.

Our reporting structures follow the standard framework of cache-oblivious geometric search structures: obtain an approximation of the output size of the query and then query the appropriate level in a multi-level reporting structure whose levels are tailored to support different output sizes. Since our counting structures use linear space and provide good enough approximations of the output size, the main challenge in obtaining more space-efficient cache-oblivious range reporting structures is to reduce the space required by such structures that know the output size.

6. REFERENCES

[1] P. Afshani. On dominance reporting in 3D. In *Proceedings of the 16th European Symposium on Algorithms*, volume 5193 of *Lecture Notes in Computer Science*, pages 41–51. Springer-Verlag, 2008.

[2] P. Afshani and T. M. Chan. On approximate range counting and depth. In *Proceedings of the 23rd ACM Symposium on Computational Geometry*, pages 337–343, 2007.

[3] P. Afshani and T. M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms*, pages 180–186, 2009.

[4] P. Afshani, C. Hamilton, and N. Zeh. Cache-oblivious range reporting with optimal queries requires superlinear space. In *Proceedings of the 25th ACM Symposium on Computational Geometry*, 2009.

[5] P. K. Agarwal, L. Arge, A. Danner, and B. Holland-Minkley. Cache-oblivious data structures for orthogonal range searching. In *Proceedings of the 19th ACM Symposium on Computational Geometry*, pages 237–245, 2003.

[6] P. K. Agarwal, L. Arge, J. Erickson, P. G. Franciosa, and J. S. Vitter. Efficient searching with linear constraints. *Journal of Computer and System Sciences*, 61(2):194–216, 2000.

[7] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal on Computing*, 29(3):912–953, 2000.

[8] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, pages 1116–1127, 1988.

[9] L. Arge, G. S. Brodal, R. Fagerberg, and M. Laustsen. Cache-oblivious planar orthogonal range searching and counting. In *Proceedings of the 21st ACM Symposium on Computational Geometry*, pages 160–169, 2005.

[10] L. Arge, M. de Berg, and H. J. Haverkort. Cache-oblivious R-trees. In *Proceedings of the 21st ACM Symposium on Computational Geometry*, pages 170–179, 2005.

[11] L. Arge, M. de Berg, H. J. Haverkort, and K. Yi. The priority R-tree: A practically efficient and worst-case optimal R-tree. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 347–358, 2004.

[12] L. Arge, V. Samoladas, and J. S. Vitter. On two-dimensional indexability and optimal range search indexing. In *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 346–357, 1999.

[13] L. Arge and N. Zeh. Simple and semi-dynamic structures for cache-oblivious orthogonal range searching. In *Proceedings of the 22nd ACM Symposium on Computational Geometry*, pages 158–166, 2006.

[14] B. Aronov and S. Har-Peled. On approximating the depth and related problems. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms*, pages 886–894, 2005.

[15] B. Aronov and S. Har-Peled. On approximating the depth and related problems, 2005. http://valis.cs.uiuc.edu/~sariel/research/papers/04/depth/.

[16] M. A. Bender, R. Cole, and R. Raman. Exponential structures for efficient cache-oblivious algorithms. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 195–207. Springer-Verlag, 2002.

[17] T. M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34:879–893, 2000.

[18] T. M. Chan. Random sampling, halfspace range reporting, and construction of $(\leq k)$-levels in three dimensions. *SIAM Journal on Computing*, 30(2):561–575, 2000.

[19] B. Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing*, 17(3):427–462, 1988.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[21] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 285–297, 1999.

[22] R. Grossi and G. F. Italiano. Efficient cross-tree for external memory. In J. Abello and J. S. Vitter, editors, *External Memory Algorithms and Visualization*, pages 87–106. American Mathematical Society, 1999.

[23] R. Grossi and G. F. Italiano. Efficient splitting and merging algorithms for order decomposable problems. *Information and Computation*, 154(1):1–33, 1999.

[24] S. Har-Peled and M. Sharir. Relative ε-approximations in geometry, 2006. http://valis.cs.uiuc.edu/~sariel/research/papers/06/relative/.

[25] K. V. R. Kanth and A. K. Singh. Optimal dynamic range searching in non-replicated index structures. In *Proceedings of the International Conference on Database Theory*, volume 1540 of *Lecture Notes in Computer Science*, pages 257–276. Springer-Verlag, 1999.

[26] H. Kaplan, E. Ramos, and M. Sharir. The overlay of minimization diagrams in a randomized incremental construction. manuscript.

[27] H. Kaplan, E. Ramos, and M. Sharir. Range minima queries with respect to a random permutation, and approximate range counting. To appear in *Discrete and Computational Geometry*.

[28] H. Kaplan and M. Sharir. Randomized incremental constructions of three-dimensional convex hulls and planar Voronoi diagrams, and approximate range counting. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms*, pages 484–493, 2006.

[29] D. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12:28–35, 1983.

[30] C. Makris and A. Tsakalidis. Algorithms for three-dimensional dominance searching in linear space. *Information Processing Letters*, 66(6):277–283, 1998.

[31] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(2):157–182, 1993.

[32] J. Matoušek. Geometric set systems. *European Congress of Mathematics*, 2:1–27, 1998.

[33] J. Matoušek. Reporting points in halfspaces. *Computational Geometry: Theory and Applications*, 2(3):169–186, 1992.

[34] E. M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–276, 1985.

[35] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 3rd edition, October 1990.

[36] O. Procopiuc, P. K. Agarwal, L. Arge, and J. S. Vitter. Bkd-tree: A dynamic scalable kd-tree. In *Proceedings of the 8th International Symposium on Advances in Spatial and Temporal Databases*, volume 2750 of *Lecture Notes in Computer Science*, pages 46–65. Springer-Verlag, 2003.

[37] E. A. Ramos. On range reporting, ray shooting and k-level construction. In *Proceedings of the 15th ACM Symposium on Computational Geometry*, pages 390–399, 1999.

[38] J. Robinson. The K-D-B tree: A search structure for large dimensional dynamic indexes. In *Proceedings of the SIGMOD International Conference on Management of Data*, pages 10–18, 1981.

[39] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264–280, 1971.