

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

Peyman Afshani^{1,*} Chris Hamilton^{2,†} Norbert Zeh^{2,‡}

¹ MADALGO[§] Dept. of Computer Science, Aarhus University, IT Parken, Aabogade 34

² Faculty of Computer Science, Dalhousie University, Halifax, NS, B3H 1W5, Canada

peyman@madalgo.au.dk, {chamilton,nzeh}@cs.dal.ca

August 24, 2009

Abstract

We present cache-oblivious solutions to two important variants of range searching: range reporting and approximate range counting. Our main contribution 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. The constructed data structures 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. As a corollary, 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 almost 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.

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 give rise 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 range 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 $(-\infty, -\infty, -\infty)$.

*Part of this work was done while visiting Dalhousie University. Work was supported in part by the Danish National Research Foundation and the Danish Strategic Research Council.

[†]Supported by a Killam Graduate Scholarship.

[‡]Supported in part by the Natural Science and Engineering Research Council of Canada, the Canadian Foundation for Innovation, and the Canada Research Chairs program.

[§]Center for Massive Data Algorithmics—a center of the Danish National Research Foundation.

In this paper, we propose a general approach based on shallow cuttings [34] that allows us to obtain cache-oblivious approximate range counting structures and cache-oblivious range reporting structures with the optimal query bounds for a wide range of problems. 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 as achieved by our data structures were known only for three-sided range reporting [5, 9, 13]. The data structure of [9] can be seen as using some notion of shallow cuttings for three-sided range queries but uses a counting structure tailored specifically to orthogonal range queries.

Using our construction, we also obtain the first internal-memory data structures for approximate 3-d halfspace range counting and approximate 3-d dominance counting that achieve the optimal query bound in the *worst case*. No such data structure was known for approximate 3-d dominance counting, and the only existing data structure with the optimal query bound for 3-d halfspace range counting [2] achieved this bound only in the *expected sense*.

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. While these models provide a useful framework for studying the fundamental computational complexity of a problem, they ignore that modern computers are equipped with memory hierarchies whose access times vary by factors of up to 10^6 depending on the memory level currently holding the accessed data item.

Among the models proposed to capture these varying access costs in real memory hierarchies, the *input-output model* (or *I/O model*) [8] and the *cache-oblivious model* [22] are the most widely accepted ones, due to the balance they provide between simplicity (which is essential to facilitate the design and analysis of sophisticated algorithms) and accuracy in predicting the real performance of algorithms.

The I/O model consists of 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 a multi-level 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. See [22] for a more detailed discussion of the model 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 in the cache-oblivious model.

Internal memory In internal memory, linear-space data structures with the optimal query bound of $O(\log N + K)$ are known for three-sided range reporting [35], 3-d dominance reporting [1, 31], 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 data structure with query time $O(\log N)$, which also immediately implies an $O(N \log N)$ -space data structure with query time $O(\log^2 N)$ for 3-d dominance counting. For exact halfspace range counting, Matoušek [32]

obtained a linear-space data 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 one for range emptiness queries; the obtained data 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 long time (see, e.g., [30]). Thus, the technique by Aronov and Har-Peled provides an $O(N \log N)$ -space approximate halfspace range counting structure with a query time of $O(\log^2 N \log \log N)$. Kaplan and Sharir [29] improved the query time by a factor of $\log \log N$ using an interesting combinatorial lemma concerning the overlay of lower envelopes in a randomized incremental construction (see also [27, 28]). As Aronov and Har-Peled, they were able to 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 [25] 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 data structure with the same worst-case query bound, as well as another linear-space data 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 model or the 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 data structures have been proposed that achieve a query bound of $O(\sqrt{N/B} + K/B)$ block transfers in 2-d and $O((N/B)^{1-1/d} + K/B)$ block transfers in d dimensions [11, 24, 23, 26, 37, 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 general 2-d orthogonal range reporting. For 3-d dominance reporting, a data structure by Afshani [1] achieves the optimal query bound of $O(\log_B N + K/B)$ block transfers using linear space. This data structure also yields a 3-d orthogonal range reporting 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 on halfspace range reporting in three and higher dimensions. The result most relevant to our work is an $O(N \log N)$ -space data 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 data structure with a query bound of $O(\log_B N + K/B)$ block transfers.

Much less is known about cache-oblivious range searching. 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 a query bound of $O(\log_B N \log \log N + K/B)$ block transfers and using $O(N \log N)$ space are easily obtained for three-sided, 3-d halfspace, and 3-d dominance queries using existing techniques (see the remarks in Section 4 for more details). Thus, the interesting questions are whether cache-oblivious data structures for these problems exist that achieve the optimal query bound of $O(\log_B N + K/B)$ block transfers and how much space is necessary to achieve this bound.

For three-sided queries, data structures with the optimal query bound and using $O(N \log N)$ space were proposed in [5, 9, 13]. The data structure by Arge and Zeh [13] was obtained using a standard reduction to

Query type	Model	Space	Query bound	References
3-d halfspace	int. mem.	$N \log N$	$\log^2 N \log \log N$ (MC)	[14]
		$N \log N$	$\log^2 N$ (MC)	[29, 27, 28, 15]
		$N \log^{O(1)} N$	$\log N \log \log N$ (WC)	[25]
		N	$\log N \log \log N$ (WC)	[2]
		N	$\log(N/K)$ (LV)	[2]
3-d halfspace	int. mem.	N	$\log(N/K)$ (WC)	new
	c.-o. & I/O	N	$\log_B(N/K)$ (WC)	new
3-d dominance	int. mem.	N	$\log(N/K)$ (WC)	new
	c.-o. & I/O	N	$\log_B(N/K)$ (WC)	new

Table 1: A comparison of our results on approximate range counting with previous work. In the listing of query bounds, WC refers to worst-case bounds; LV to Las Vegas bounds, that is, query bounds that hold in the expected sense; and MC to Monte Carlo bounds, that is, the answer to a query is correct with high probability.

2-d dominance queries, for which the paper presented a linear-space data structure with the optimal query bound. The data 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 recently made progress towards answering how much space is required to achieve the optimal query bound. In particular, we showed that a cache-oblivious data structure that achieves the optimal (or in fact even a much weaker) query bound for three-sided range reporting, 3-d halfspace range reporting or 3-d dominance reporting must use $\Omega(N(\log \log N)^\epsilon)$ 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 data structures, except the 3-d orthogonal range reporting structure, use $O(N \log N)$ space. Using a standard transformation, our 3-d dominance reporting structure also provides a new $O(N \log N)$ -space data structure with the 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 of a query can be efficiently determined or at least approximated.

Our main technical contribution, therefore, is a general framework for constructing cache-oblivious data structures for the approximate counting versions of the above problems. These data structures use linear space and provide guaranteed $(1 + \epsilon)$ -approximate answers using $O(\log_B(N/K))$ block transfers in the worst case, which is optimal. 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 data structure with the optimal query bound, by Afshani and Chan [2], achieves this bound only in the expected case. Thus, our construction also provides new worst-case optimal data structures for approximate 3-d halfspace range counting and approximate 3-d 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; the novelty of our approach lies in the manner in which we combine them with other equally standard techniques to obtain the above series of new results.

¹It 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.

2 Preliminaries

The framework presented in this paper is applicable to any range searching problem that, through application of duality or other techniques, can be translated into the following type of “aboveness reporting problem” and satisfies a number of additional properties discussed below. Let \mathcal{F} be a collection of continuous and totally defined algebraic functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ of constant degree. Each such function defines a continuous surface in \mathbb{R}^3 consisting of the points $(x, y, f(x, y))$, and we do not distinguish between a function and the surface it defines. We say that a function f *passes below* a 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 report or (approximately) count the functions in \mathcal{F} passing below q . Since we assume that such an aboveness query is an alternative representation of a range query, we refer to reporting or counting the functions that pass below a query point as range reporting or range counting throughout this paper.

The *arrangement* of a collection \mathcal{F} of functions is a subdivision of \mathbb{R}^3 into *cells*. Each such cell is a maximal connected set of points that are contained in a subset $\mathcal{F}' \subseteq \mathcal{F}$ of functions and in no other functions in \mathcal{F} . 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 at 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} . 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 shallow cutting for the $(\leq k)$ -level by removing all cells from \mathcal{C} that do not intersect this level). The *conflict list* Δ_C of a cell C is the set of functions in \mathcal{F} that intersect C or pass below it. By the above assumption, we have $|\Delta_C| = O(k)$ and, for every point $p \in C$, all functions in \mathcal{F} that pass below p are included in Δ_C .

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

- (i) For every k , there exists a shallow cutting for the $(\leq k)$ -level of \mathcal{F} consisting of $O(|\mathcal{F}|/k)$ cells, each bounded by a constant number of algebraic curves of constant degree.
- (ii) For every shallow cutting \mathcal{C} as in (i), there exists a cache-oblivious *point location structure* $\mathcal{L}(\mathcal{C})$. For any query point q , this data structure finds the cell C of \mathcal{C} that contains q , or reports that no such cell exists. We require that this data structure 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 a number of shallow cuttings as in (i) into the xy -plane. Then there exists a cache-oblivious point location structure for \mathcal{A} with a query bound of $O(\log_B |\mathcal{A}|)$ block transfers 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 [34, 1]. 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], and condition (ii) can be satisfied using the same data 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 a cache-oblivious approximate range counting structure 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 linear-space cache-oblivious data structure that supports $(1+\varepsilon)$ -approximate range counting queries using $O(\log_B(N/K))$ block transfers.*

Proof. The $(1+\varepsilon)$ -approximate range counting structure we construct for \mathcal{F} consists of \mathcal{D} and data structures representing shallow cuttings $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_n$, where $n = \lceil \log N \rceil$ and \mathcal{C}_i is a shallow cutting for the $(\leq 2^i)$ -level of \mathcal{F} . The data structure representing each shallow cutting \mathcal{C}_i consists of the point location structure $\mathcal{L}(\mathcal{C}_i)$ of \mathcal{C}_i and a collection of δ -approximate conflict lists for the cells of \mathcal{C}_i , where $\delta > 0$ is a constant defined below. The δ -approximate conflict list $\tilde{\Delta}_C$ of a cell $C \in \mathcal{C}_i$ is a sublist of Δ_C of constant size such that the level of a query point $q \in C$ can be approximated to within an additive error of $\delta|\Delta_C|$ using only the level of q in $\tilde{\Delta}_C$. Appendix A discusses how to obtain such a sublist $\tilde{\Delta}_C$ of Δ_C for each cell $C \in \mathcal{C}_i$, using results in VC-dimension.

The size of the data structure is linear. By condition (i), each shallow cutting \mathcal{C}_i has size $O(N/2^i)$, which implies that the δ -approximate conflict lists of the cells of \mathcal{C}_i and, by condition (ii), $\mathcal{L}(\mathcal{C}_i)$ use $O(N/2^i)$ space. By summing over all shallow cuttings $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_n$, we obtain that the data structures representing them use linear space, as does \mathcal{D} .

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 because $K' \geq K$, and we can use $\mathcal{L}(\mathcal{C}_i)$ to find the cell $C \in \mathcal{C}_i$ that contains q . Next we compute the level of q in $\tilde{\Delta}_C$ and use it to compute an approximation K'' of K that satisfies $K \leq K'' \leq K + \delta|\Delta_C|$. Since $K' \leq c_1K$ and $|\Delta_C| \leq c_2K'$, for some constants c_1 and c_2 , K'' is a $(1+\varepsilon)$ -approximation of K if we choose $\delta \leq \varepsilon/(c_1c_2)$.

The cost of this query procedure is $O(\log_B(N/K))$: querying \mathcal{D} takes $O(\log_B(N/K))$ block transfers, as does querying $\mathcal{L}(\mathcal{C}_i)$, by condition (ii) and because $|\mathcal{C}_i| = O(N/K)$; given the cell $C \in \mathcal{C}_i$ that contains q , scanning $\tilde{\Delta}_C$ to compute the level of q in $\tilde{\Delta}_C$ takes $O(1)$ block transfers because $|\tilde{\Delta}_C| = O(1)$. \square

By Lemma 1, it suffices to construct a range counting structure for \mathcal{F} that approximates K to within any constant factor. We split the construction of such a structure 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 data structure for polynomial queries that uses *sublinear* space and achieves the query bound stated in Lemma 1. Using this structure, we can obtain a data 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 and, thus, can be used as the data structure \mathcal{D} in Lemma 1. The next three subsections discuss these three parts of our construction 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. The sublinear space bound is crucial to ensure that our data structure

for polylogarithmic queries, discussed in Section 3.2, uses linear space.

Throughout the next two sections, we fix c to 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 exists.

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 queries using $O(\log_B(N/K))$ block transfers in the worst case, as long as the count K satisfies $K \geq N^{1-\delta}/2^c$, for a sufficiently small constant $\delta > 0$.*

Data structure To obtain a data structure as in Lemma 2, we choose a constant $\delta > 0$ to be defined later and construct a shallow cutting \mathcal{C}_i for the $(\leq N/2^i)$ -level of \mathcal{F} , for each $0 \leq i \leq 2(c + \delta \log N)$. 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 $\mathcal{A}_0^*, \mathcal{A}_1^*, \dots, \mathcal{A}_t^*$, where $t = \lfloor \log(2\delta \log N) \rfloor$ and \mathcal{A}_i^* is obtained by overlaying arrangements $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_{2c+2^i}$. We represent each arrangement \mathcal{A}_i^* using a point location structure as in condition (iii) and store for each face $f \in \mathcal{A}_i^*$, the list of cells of the shallow cuttings $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{2c+2^i}$ that project onto f ; if more than one cell of a shallow cutting \mathcal{C}_j projects onto f , we store only the highest one. These representations of arrangements $\mathcal{A}_0^*, \mathcal{A}_1^*, \dots, \mathcal{A}_t^*$ are stored consecutively in memory, in order of increasing indices.

Space bound To bound the space used by this data structure, we observe that $|\mathcal{C}_j| = O(2^j)$ and, hence, that the total number of cells in the shallow cuttings $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{2c+2^i}$ is $O(2^{2^i})$. This implies that \mathcal{A}_i^* has size polynomial in 2^{2^i} because, by condition (i), each cell in each shallow cutting \mathcal{C}_j is bounded by a constant number of algebraic curves of constant degree. In particular, the size of the largest arrangement \mathcal{A}_t^* —and, thus, the total size of all arrangements $\mathcal{A}_0^*, \mathcal{A}_1^*, \dots, \mathcal{A}_t^*$ —is polynomial in $2^{2^t} \leq N^{2^\delta}$. Since each face of an arrangement \mathcal{A}_i^* stores a list of $c + 1 + 2^i = O(\log N)$ cells and, by condition (iii), the size of the point location structure for each arrangement \mathcal{A}_i^* is polynomial in $|\mathcal{A}_i^*|$, this implies that the entire data structure uses space polynomial in N^{2^δ} . By choosing a sufficiently small $\delta > 0$, we can thus ensure that the data structure uses $O(\sqrt{N})$ space.

Query procedure To answer a query q , note that it suffices to find an index j such that \mathcal{C}_j contains q but \mathcal{C}_{j+1} does not: the former condition implies that $K \leq 2^c N/2^j$, because q is contained in a cell of \mathcal{C}_j ; the latter condition implies that $K > N/2^{j+1}$, because q does not belong to the $(\leq N/2^{j+1})$ -level of \mathcal{F} ; thus, $K' = 2^c N/2^j$ is a constant-factor approximation of K .

To find such an index j , we start from $i = 0$ and decide for each of the shallow cuttings $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{2c+2^i}$ whether it contains q . If we find an index $0 \leq j < 2c + 2^i$ such that $q \in \mathcal{C}_j$ but $q \notin \mathcal{C}_{j+1}$, we report $K' = 2^c N/2^j$. Otherwise, if $i < t$, we increment i and repeat this procedure or, if $i = t$, report that K is too small, and the query fails.

To determine in iteration i which of the shallow cuttings $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{2c+2^i}$ contain the query point q , note that $q \in \mathcal{C}_j$, for some $0 \leq j \leq c + 2^i$ if and only if q lies in or below the cell C of \mathcal{C}_j stored with the face f of \mathcal{A}_i^* that contains the projection of q into the xy -plane. Hence, we implement the i th iteration of our query procedure by finding this face f , using the point location structure of \mathcal{A}_i^* , and then scanning the list of cells of arrangements $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{2c+2^i}$ stored with f .

Correctness We have already argued that the query procedure outputs a constant-factor approximation of K unless it fails. Thus, it suffices to show that the query procedure does not fail for any $K \geq N^{1-\delta}/2^c$. In this case, however, we have $K \geq N^{1-\delta}/2^c \geq N/2^{2^{\lfloor \log(2\delta \log N) \rfloor + c}} = N/2^{2^t + c}$, and the choice of the constant c implies that $q \notin \mathcal{C}_{2c+2^t}$, that is, the query does not fail.

Query bound To bound the cost of a successful query, we observe that $2c + 2^{i-1} \leq j < 2c + 2^i$ when the query terminates. Since the size of the representation of each arrangement \mathcal{A}_i^* is polynomial in 2^{2^i} , the combined size of the representations of the first $i_0 := \log \log B - a$ arrangements $\mathcal{A}_0^*, \mathcal{A}_1^*, \dots, \mathcal{A}_{i_0}^*$ is $O(B)$,

for an appropriate constant a . 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 $K = \Theta(N/2^j)$ and $2^{2c+2^{i-1}} \leq 2^j < 2^{2c+2^i}$. This implies that $2^{2^i} = O((N/K)^2)$. The query cost for the case $i > i_0$ is therefore bounded by

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

For an unsuccessful query, the query terminates when $i = t$. By substituting this into the previous summation, we obtain a query bound of $O(\log_B N^\delta)$ block transfers, which is $O(\log_B(N/K))$, as $K \leq N^{1-\delta}/2^c$ in this case.

3.2 A Structure for Polylogarithmic Queries

In this section, we present the second building block of our data structure, a linear-space data structure for answering polylogarithmic approximate range counting queries, as summarized in the following lemma. Note that the query bound is $O(\log_B N)$, not $O(\log_B(N/K))$. This is sufficient for the purposes of our final data 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 queries using $O(\log_B N)$ block transfers in the worst case, as long as the count K satisfies $K > \log^\tau N$, for a sufficiently large constant $\tau > 0$.*

Data structure To obtain a data structure as in Lemma 3, we apply Lemma 2 recursively. Let \mathcal{D} be the data structure for \mathcal{F} provided by Lemma 2, and let \mathcal{C} be a shallow cutting for the $(\leq N^{1-\delta}/2^c)$ -level of \mathcal{F} , for the same constants c and δ as in Section 3.1. In the memory layout, we represent \mathcal{F} using the point location structure $\mathcal{L}(\mathcal{C})$ for \mathcal{C} , the data structure \mathcal{D} , and data structures representing the cells of \mathcal{C} , arranged in this order. The data structure representing each cell $C \in \mathcal{C}$ is constructed by recursively applying the construction just described to Δ_C . The recursion stops as soon as we obtain conflict lists of size at most $\log^\tau N$, for a constant τ to be chosen below.

Space bound Let $S(N)$ be the space complexity of our data structure for a set of N functions. Since \mathcal{C} is a shallow cutting for the $(\leq N^{1-\delta}/2^c)$ -level of \mathcal{F} , \mathcal{C} has $O(N^\delta)$ cells, each with a conflict list of size at most $N^{1-\delta}$. Since we do not recurse on conflict lists of size $\log^\tau N$ or less, this implies that $S(N)$ is bounded by the 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 $i = \log_{(1-\delta)}(\log \log^\tau N / \log N)$, we obtain

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

as $\log_{(1-\delta)}(\log \log^\tau N / \log N) \leq (\log \log N)/\delta$, for all $\tau \geq 1$ and $N \geq 4$. By choosing τ large enough, we thus obtain $S(N) = O(N)$ because $S(\log^\tau N) = O(1)$.

Query procedure 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 data structure representing Δ_C or report a failure if we are already at the last level of recursion in the structure. If $q \notin \mathcal{C}$, we use \mathcal{D} to answer the query.

Correctness First assume that the query does not fail. If $q \notin \mathcal{C}$, then $K \geq N^{1-\delta}/2^c$, and Lemma 2 shows that \mathcal{D} provides the desired approximation of K in this case. 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 . Thus, in both cases, we obtain a correct approximation of K .

If the query fails, on the other hand, then q is contained in a 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.

Query bound The query bound obeys the recurrence

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

The bound for $N \leq B$ follows because $S(B) = O(B)$ and, hence, the entire recursive structure for a conflict list of size B fits in $O(1)$ blocks. The bound for $N > B$ follows because querying $\mathcal{L}(\mathcal{C})$ and \mathcal{D} requires $O(\log_B N)$ block transfers, and querying the structure for the conflict list Δ_C in the case when $q \in \mathcal{C}$ takes $Q(|\Delta_C|) \leq Q(N^{1-\delta})$ block transfers. This recurrence is easily seen to yield $Q(N) = O(\log_B N)$, that is, the constructed data structure achieves the query bound claimed in Lemma 3.

3.3 The Final Structure

In this section, we combine Lemmas 2 and 3 to obtain a linear-space data 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.*

Data structure Our final data structure consists of the following components. Let \mathcal{C}_s and \mathcal{C}_b be shallow cuttings for the $(\leq \log N)$ -level and for the $(\leq \log^\tau N)$ -level of \mathcal{F} , respectively. We represent \mathcal{F} using four data structures: a structure \mathcal{D}_b for polynomial queries, a structure \mathcal{D}_s for polylogarithmic queries, and the two point location structures $\mathcal{L}(\mathcal{C}_s)$ and $\mathcal{L}(\mathcal{C}_b)$ for the two shallow cuttings \mathcal{C}_s and \mathcal{C}_b . In addition, we store for each cell $C \in \mathcal{C}_b$, a data structure \mathcal{D}_C for polylogarithmic queries over Δ_C , and for each cell $C \in \mathcal{C}_s$, the list of functions in Δ_C .

Space bound By Lemmas 2 and 3, both, \mathcal{D}_b and \mathcal{D}_s , use linear space. The data structure representing each cell C in \mathcal{C}_b has size $O(|\Delta_C|) = O(\log^\tau N)$, and the number of cells in \mathcal{C}_b is $O(N/\log^\tau N)$. Thus, the representation of \mathcal{C}_b uses linear space. Similarly, the conflict list of each cell C in \mathcal{C}_s can be stored in $O(\log N)$ space, and there are $O(N/\log N)$ such cells, that is, the representation of \mathcal{C}_s uses linear space. This shows that the entire data structure uses linear space.

Query procedure Given a query point q , we first query \mathcal{D}_b to see whether q is polynomial ($K \geq N^{1-\delta}/2^c$) and, if so, obtain the desired approximation of K . If this fails, we try \mathcal{D}_s . If this also fails, that is, $K \leq \log^\tau N$, then $q \in \mathcal{C}_b$. In this case, we query $\mathcal{L}(\mathcal{C}_s)$ to decide whether $q \in \mathcal{C}_s$ and, if so, determine the cell $C \in \mathcal{C}_s$ that contains q . If $q \in \mathcal{C}_s$, we scan Δ_C to count the number of functions passing below q . If $q \notin \mathcal{C}_s$, we query

$\mathcal{L}(\mathcal{C}_b)$ to find the cell $C \in \mathcal{C}_b$ that contains q and then use \mathcal{D}_C to obtain the desired approximation of K . (As we argue next, the query on \mathcal{D}_C does not fail.)

Correctness By Lemmas 2 and 3, if one of the queries on \mathcal{D}_b and \mathcal{D}_s succeeds, it reports the correct answer. As already observed, if both queries fail, then $q \in \mathcal{C}_b$. If $q \in \mathcal{C}_s$, then all functions passing below q belong to Δ_C , where C is the cell of \mathcal{C}_s that contains q ; thus, scanning Δ_C in this case allows us to determine K exactly. If $q \notin \mathcal{C}_s$, then $K \geq \log N$. This implies in particular that q is a polylogarithmic query for the conflict list Δ_C of the cell C of \mathcal{C}_b that contains q . Hence, the query on \mathcal{D}_C does not fail in this case and reports a correct constant-factor approximation of K .

Query bound We divide the analysis of the query cost into polynomial queries and sub-polynomial queries. If $K \geq N^{1-\delta}$, the query on \mathcal{D}_b succeeds and takes $O(\log_B(N/K))$ block transfers, by Lemma 2. Since no further queries are performed after a successful query on \mathcal{D}_b , this shows that polynomial queries can be answered using $O(\log_B(N/K))$ block transfers.

If $K < N^{1-\delta}$, the cost of the query procedure is bounded by the cost of one query on each of \mathcal{D}_b , \mathcal{D}_s , $\mathcal{L}(\mathcal{C}_b)$, and $\mathcal{L}(\mathcal{C}_s)$, plus the cost of querying the approximate counting structure \mathcal{D}_C associated with a cell C of \mathcal{C}_b or scanning the conflict list $\Delta_{C'}$ of a cell C' of \mathcal{C}_s . Querying any of the point location or approximate counting structures takes $O(\log_B N)$ block transfers, by Lemmas 2 and 3 and by condition (ii). Scanning $\Delta_{C'}$ takes $O((\log N)/B) = O(\log_B N)$ block transfers, as $|\Delta_{C'}| = O(\log N)$. Hence, the total query cost for sub-polynomial queries is $O(\log_B N)$ block transfers. Since $K < N^{1-\delta}$, however, we have $O(\log_B N) = O(\log_B(N^\delta)) = O(\log_B(N/K))$, that is, the query procedure achieves the query bound claimed in Lemma 4 in this case as well.

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 [34]. 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 data structure can be used to satisfy condition (iii).

For dominance counting, we can represent every input point p using a range \bar{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 \bar{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 that any collection of such boundary functions satisfies 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. \square

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 and achieves the optimal query bound.

Given a set \mathcal{F} of N functions satisfying conditions (i)–(iii), such a data 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 shallow cutting \mathcal{C}_i uses linear space. As there are $\log N$ shallow 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 shallow cutting \mathcal{C}_i . This adds only $O(N)$ to the total space bound.

To answer a range reporting query with a 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 the 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.*

Using standard reductions from three-sided range reporting to 3-d dominance reporting and from circular range reporting to 3-d halfspace range reporting. (see, e.g., [36]), Corollary 1 immediately implies further results on cache-oblivious three-sided range reporting and circular range reporting. Moreover, the construction of Theorem 3 can also be used to obtain a cache-oblivious data structure for K -nearest neighbour searching: a K -nearest neighbour query is equivalent to reporting the K lowest planes stabbed by a vertical line ℓ ; we can identify these planes using $O(\log_B N + K/B)$ block transfers by identifying the cutting \mathcal{C}_i with $i = \lceil \log K \rceil$, using $\mathcal{L}(\mathcal{C}_i)$ (which is a planar point location structure on the xy -projection of \mathcal{C}_i) to find a cell $C \in \mathcal{C}_i$ stabbed by ℓ , and finally applying a linear-time selection algorithm (e.g., see [21, Chapter 9]) to Δ_C to find the K lowest planes in Δ_C stabbed by ℓ . Except for three-sided range reporting, similar results were not known in the cache-oblivious model before.

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.*

The final consequence of Corollary 1 is the first cache-oblivious data structure for 3-d orthogonal range reporting using the optimal query bound. This structure is obtained using a standard reduction of this problem to 3-d dominance reporting [20].

Corollary 3. *There exists a cache-oblivious 3-d range reporting 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 appropriate

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.

References

- [1] Peyman 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] Peyman Afshani and Timothy M. Chan. On approximate range counting and depth. *Discrete and Computational Geometry*, 42:3–21, 2009.
- [3] Peyman Afshani and Timothy 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] Peyman Afshani, Chris Hamilton, and Norbert Zeh. Cache-oblivious range reporting with optimal queries requires superlinear space. In *Proceedings of the 25th ACM Symposium on Computational Geometry*, pages 277–286, 2009.
- [5] Pankaj K. Agarwal, Lars Arge, Andrew Danner, and Bryan 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] Pankaj K. Agarwal, Lars Arge, Jeff Erickson, Paolo G. Franciosa, and Jeffrey S. Vitter. Efficient searching with linear constraints. *Journal of Computer and System Sciences*, 61(2):194–216, 2000.
- [7] Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal on Computing*, 29(3):912–953, 2000.
- [8] Alok Aggarwal and Jeffrey Scott Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, pages 1116–1127, 1988.
- [9] Lars Arge, Gerth Stølting Brodal, Rolf Fagerberg, and Morten Laustsen. Cache-oblivious planar orthogonal range searching and counting. In *Proceedings of the 21st ACM Symposium on Computational Geometry*, pages 160–169, 2005.
- [10] Lars Arge, Mark de Berg, and Herman J. Haverkort. Cache-oblivious R-trees. In *Proceedings of the 21st ACM Symposium on Computational Geometry*, pages 170–179, 2005.
- [11] Lars Arge, Mark de Berg, Herman J. Haverkort, and Ke 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] Lars Arge, Vasilis Samoladas, and Jeffrey 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] Lars Arge and Norbert 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] Boris Aronov and Sariel 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] Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems, 2005. <http://valis.cs.uiuc.edu/~sariel/research/papers/04/depth/>.
- [16] Michael A. Bender, Richard Cole, and Rajeev 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] Timothy M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34:879–893, 2000.
- [18] Timothy 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] Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM Journal on Computing*, 17(3):427–462, 1988.
- [20] Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: II. applications. *Algorithmica*, 1:163–191, 1986.
- [21] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.
- [22] Matteo Frigo, Charles E. Leiserson, Harald Prokop, and Sridhar Ramachandran. Cache-oblivious algorithms. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 285–297, 1999.
- [23] Roberto Grossi and Giuseppe 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.
- [24] Roberto Grossi and Giuseppe F. Italiano. Efficient splitting and merging algorithms for order decomposable problems. *Information and Computation*, 154(1):1–33, 1999.
- [25] Sariel Har-Peled and Micha Sharir. Relative ε -approximations in geometry, 2006. <http://valis.cs.uiuc.edu/~sariel/research/papers/06/relative/>.
- [26] 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.
- [27] Haim Kaplan, Edgar Ramos, and Micha Sharir. The overlay of minimization diagrams in a randomized incremental construction. manuscript.
- [28] Haim Kaplan, Edgar Ramos, and Micha Sharir. Range minima queries with respect to a random permutation, and approximate range counting. To appear in *Discrete and Computational Geometry*.
- [29] Haim Kaplan and Micha 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.

- [30] David Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12:28–35, 1983.
- [31] Christos Makris and Athanasios Tsakalidis. Algorithms for three-dimensional dominance searching in linear space. *Information Processing Letters*, 66(6):277–283, 1998.
- [32] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(2):157–182, 1993.
- [33] J. Matoušek. Geometric set systems. *European Congress of Mathematics*, 2:1–27, 1998.
- [34] Jiri Matoušek. Reporting points in halfspaces. *Computational Geometry: Theory and Applications*, 2(3):169–186, 1992.
- [35] Edward M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–276, 1985.
- [36] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 3rd edition, October 1990.
- [37] 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.
- [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.

A Approximate Conflict Lists

In the proof of Lemma 1, we claimed that every set S of m algebraic functions of constant degree has a subset \tilde{S} of constant size such that, for any query point q with K functions in S passing below q , an approximation, K' , of K with $K \leq K' \leq K + \delta m$ can be computed from the number, \tilde{K} , of functions in \tilde{S} passing below q . VC-dimension theory defines the notion of an ε -approximation, which has exactly this property.

A set S of algebraic curves of constant degree defines a set system (S, \mathcal{R}) , where $\mathcal{R} = \{S_p \mid p \in \mathbb{R}^3\}$ and S_p is the set of functions in S that pass below the point p . In general, a set system (S, \mathcal{R}) consists of a base set S and a collection $\mathcal{R} \subseteq 2^S$ of subsets of S . An ε -approximation of S w.r.t. \mathcal{R} is a subset $\tilde{S} \subseteq S$ such that

$$\left| \frac{|\tilde{S} \cap R|}{|\tilde{S}|} - \frac{|R|}{|S|} \right| \leq \varepsilon,$$

for all $R \in \mathcal{R}$. In other words, as $|S| = m$, $|R| = K$, and $|\tilde{S} \cap R| = \tilde{K}$,

$$\frac{m\tilde{K}}{|\tilde{S}|} - \varepsilon m \leq K \leq \frac{m\tilde{K}}{|\tilde{S}|} + \varepsilon m.$$

This implies that, for $\varepsilon = \delta/2$, the value $K' := m\tilde{K}/|\tilde{S}| + \varepsilon m$ provides the desired approximation of K , that is, the subset of S needed in Lemma 1 is a $(\delta/2)$ -approximation. Since S is a set of algebraic functions of constant degree and (S, \mathcal{R}) is the corresponding set system defined above, we can show that S has a $(\delta/2)$ -approximation w.r.t. \mathcal{R} of constant size: Vapnik and Chervonenkis [39] showed that every set system of constant VC-dimension has an ε -approximation of size $O(\varepsilon^{-2} \log(\varepsilon^{-1}))$, and Matoušek [33] showed that the set system defined by a set of algebraic functions of constant degree has constant VC-dimension. Therefore, since $\delta = \Theta(1)$ in Lemma 1, S has a $(\delta/2)$ -approximation of constant size.