

# Optimal-Cost Construction of Shallow Cuttings for 3-D Dominance Ranges in the I/O-Model

Yakov Nekrich 

Michigan Technological University, Houghton, MI, USA

Saladi Rahul 

Indian Institute of Science, Bangalore, India

---

## Abstract

Shallow cuttings are a fundamental tool in computational geometry and spatial databases for solving offline and online range searching problems. For a set  $P$  of  $N$  points in 3-D, at SODA'14, Afshani and Tsakalidis designed an optimal  $O(N \log_2 N)$  time algorithm that constructs shallow cuttings for 3-D dominance ranges in internal memory. Even though shallow cuttings are used in the I/O-model to design space and query efficient range searching data structures, an efficient construction of them is not known till now. In this paper, we design an optimal-cost algorithm to construct shallow cuttings for 3-D dominance ranges. The number of I/Os performed by the algorithm is  $O\left(\frac{N}{B} \log_{M/B}\left(\frac{N}{B}\right)\right)$ , where  $B$  is the block size and  $M$  is the memory size.

As two applications of the optimal-cost construction algorithm, we design fast algorithms for offline 3-D dominance reporting and offline 3-D approximate dominance counting. We believe that our algorithm will find further applications in offline 3-D range searching problems and in improving construction cost of data structures for 3-D range searching problems.

**2012 ACM Subject Classification** Theory of computation → Data structures and algorithms for data management; Theory of computation → Computational geometry

**Keywords and phrases** Data Structures, I/O-efficient algorithms, Orthogonal Range Searching

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2026.81

**Related Version** Full Version: <https://arxiv.org/abs/2603.21337>

**Funding** *Yakov Nekrich*: Supported by the National Science Foundation under NSF grant 2203278. *Saladi Rahul*: Supported by the Walmart Center for Tech Excellence at IISc (CSR Grant WMGT-230001) and Anusandhan National Research Foundation (SP/ANRF-24-0043 CRG/2023/005776).

**Acknowledgements** The authors thank the reviewers whose detailed feedback helped in improving the presentation of the paper.

## 1 Introduction

*Shallow cuttings* are one of the most fundamental tools used for designing range searching data structures in computational geometry [1, 2, 3, 4, 13] and spatial databases [16, 17, 6]. The main contribution of this paper is the first known optimal-cost construction of shallow cuttings for *3-D dominance ranges* in the *I/O-model*. As a consequence, we obtain efficient algorithms for the *offline 3-D dominance reporting* problem and the *offline 3-D approximate counting* problem.

**Shallow cuttings for 3-D dominance ranges.** Consider two 3-D points  $p$  and  $q$ . A point  $p$  *dominates* another point  $q$  if and only if  $p$  has a higher coordinate value than  $q$  in each dimension. Let  $P$  be a set of  $N$  points in 3-D. A  $k$ -level *shallow cutting* of  $P$  [1] (See Figure 1(a)) is a collection  $\mathcal{C}$  of *cells* of the form  $(-\infty, x] \times (-\infty, y] \times (-\infty, z]$  satisfying the following three properties:



© Yakov Nekrich and Saladi Rahul;

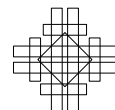
licensed under Creative Commons License CC-BY 4.0

42nd International Symposium on Computational Geometry (SoCG 2026).

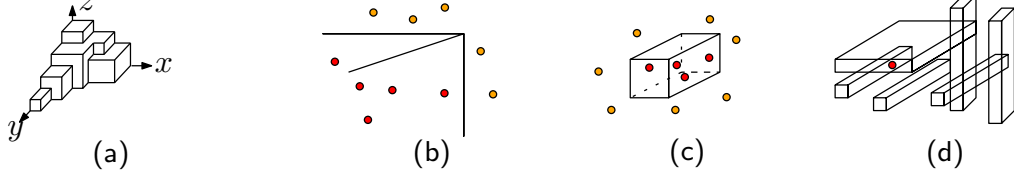
Editors: Hee-Kap Ahn, Michael Hoffmann, and Amir Nayyeri; Article No. 81; pp. 81:1–81:15

Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



1. The number of cells is  $O(n/k)$ , i.e.,  $|\mathcal{C}| = O(n/k)$ .
2. Each cell in  $\mathcal{C}$  contains at most  $10k$  points from  $P$ .
3. Any 3-D point that dominates at most  $k$  points of  $P$  lies inside at least one cell in  $\mathcal{C}$ .



■ **Figure 1** (a) A schematic representation of a  $k$ -level 3-D shallow cutting, (b) 3-D dominance reporting and approximate counting query, (c) 3-D orthogonal range reporting query, (d) 3-D rectangle stabbing query.

**I/O-model of computation.** We will construct shallow cuttings for 3-D dominance ranges in the I/O-model of computation [21]. In the era of big-data, it is natural that the input will not fit completely in the internal memory, and hence, resides in the hard-disk (or the external memory). In this model, the internal memory has  $M$  words, and the hard-disk has been formatted into *blocks* of  $B$  words. An I/O operation exchanges a block of data between the disk and the memory. The space of a data structure is the number of blocks occupied in the external memory, and the running time of an algorithm is the number of I/Os performed. The CPU calculation is free. A natural assumption is that  $M = \Omega(B)$ .

**3-D orthogonal range searching.** Orthogonal range searching and related problems are foundational problems on which the fields of computational geometry and spatial databases grew. Naturally, they have been well studied in the I/O-model of computation. Consider the following problems for which the state-of-the-art data structures in the I/O-model extensively rely on the clever use of shallow cuttings for 3-D dominance ranges:

1. In the *3-D orthogonal range reporting* problem (Figure 1(c)), the input is a set  $P$  of  $N$  points in 3-D. Preprocess  $P$  into a space-efficient data structure, so that given an axis-aligned query box  $q = [x_1, x_2] \times [y_1, y_2] \times [z_1, z_2]$ , the goal is to quickly report  $P \cap q$  (the points of  $P$  lying inside  $q$ ). At FOCS'09, Afshani, Arge, and Larsen [2] presented a data structure for this problem with space  $O\left(\frac{N}{B} \left(\frac{\log N}{\log \log_B N}\right)^3\right)$  and query I/Os  $O(\log_B N + k/B)$ , where  $k$  is the number of points reported.
2. The building block for the above problem is the *3-D dominance reporting* problem (Figure 1(b)), where the query is a 3-D dominance range  $q = (-\infty, x] \times (-\infty, y] \times (-\infty, z]$ . At ESA'08, Afshani [14] presented a data structure for this problem with space  $O(N/B)$  and query I/Os  $O(\log_B N + k/B)$ . This is optimal in terms of space and query I/Os.
3. At SoCG'09, Afshani, Hamilton, and Zeh presented a general approach to design data structures to answer the *relative  $(1 + \varepsilon)$ -approximate counting* query for a general class of problems including the 3-D dominance setting. Here the goal is to output the estimate of  $|P \cap q|$ . An optimal solution in terms of space and query I/Os was obtained.
4. *Rectangle stabbing reporting* (Figure 1(d)) is the “reverse” of orthogonal range reporting where the input is a set of axis-aligned 3-D boxes and the query is a 3-D point. Arguably, rectangle stabbing reporting is as important as orthogonal range reporting and is well studied in the literature. The current state-of-the-art results in internal memory models of computation for rectangle stabbing reporting by Rahul [15] at SODA'15 and Chan,

Nekrich, Rahul, and Tsakalidis [9] at ICALP'18 rely on shallow cuttings for 3-D dominance ranges. It is unlikely that a linear-space data structure in the I/O-model can be constructed without using shallow cuttings.

**Lack of fast construction of data structures.** An important criteria in the design of the data structures is to optimize the I/Os required to construct the data structure. A severe limitation of *all* the above solutions is the lack of such fast construction algorithms. In fact, quoting from one of the above papers (Afshani, Hamilton, and Zeh [4]),

*“All data structures presented in this paper are static, and no efficient construction methods for these structures are known in the I/O or cache-oblivious model. The main obstacle is the lack of an I/O-efficient or cache-oblivious construction procedure for shallow cuttings.”*

We believe that our work on optimal-cost construction of shallow cuttings for 3-D dominance ranges will trigger further work to design fast construction of data structures for the above mentioned problems.

**Lack of optimal solutions for offline 3-D orthogonal range searching.** In the *offline* orthogonal range searching problem, we are given a set  $Q$  of queries upfront along with the point set  $P$ . Many 2-D offline problems in computational geometry (including range searching type problems) have been optimally solved in the I/O-model. See the survey paper by Vitter (Theorem 8.1 in [21] provides a comprehensive list of more than ten problems in 2-D which are solved I/O-optimally).

However, to the best of our knowledge, offline 3-D orthogonal range searching problems have not received the same attention in spite of their fundamental nature. Again, the lack of optimal-cost shallow cuttings in 3-D is the major bottleneck.

## 1.1 Our results

The main result in this paper is an optimal-cost construction of the  $k$ -level shallow cutting for 3-D dominance ranges in the I/O-model.

► **Theorem 1.** *The  $k$ -level shallow cutting for 3-D dominance ranges on  $N$  points can be constructed in  $O\left(\frac{N}{B} \log_{M/B}\left(\frac{N}{B}\right)\right)$  I/Os. The construction cost is optimal in the I/O-model.*

To demonstrate the impact of our new construction, we use shallow cuttings for 3-D dominance ranges to obtain efficient algorithms for two fundamental problems: *offline 3-D dominance approximate counting* and *offline 3-D dominance reporting*.

► **Theorem 2 (Application-1).** *Let  $P$  be a set of  $N$  points in 3-D and  $Q$  be a set of 3-D dominance ranges. In the offline 3-D dominance approximate counting problem, for any  $q \in Q$ , the goal is to return a value in the range  $[(1 - \varepsilon)|P \cap q|, |P \cap q|]$ , where  $\varepsilon \in (0, 1)$  is a fixed value. There is a deterministic algorithm to solve the problem in  $O_\varepsilon\left(\frac{|P|}{B} \left(\log_{M/B} \frac{|P|}{B}\right)^{O(1)} + \frac{|Q|}{B} \log_{M/B} \frac{|P|}{B}\right)$  I/Os. The notation  $O_\varepsilon(\cdot)$  hides the dependency on  $\varepsilon$ .*

The key feature in the above theorem is that the construction cost bound has the term  $M/B$  in the base of the logarithm which is desirable in the I/O-model (instead of *base-2* obtained from straightforward adaptation of internal memory algorithms or *base- $B$*  obtained from straightforward adaption of external memory data structures for online problems). In

terms of techniques, even though the final output is a relative approximation, the crucial ingredients in the algorithm are an additive error counting structure that is combined with a non-trivial *van-Emde-Boas-style recursion*.

► **Theorem 3** (Application-2). *Let  $P$  be a set of  $N$  points in 3-D and  $Q$  be a set of 3-D dominance ranges. In the offline 3-D dominance reporting problem, for any  $q \in Q$ , the goal is to report  $P \cap q$ . The problem can be solved in  $O\left(\frac{|P|+|Q|}{B} \log_{M/B} \frac{|P|}{B} + \frac{K}{B}\right)$  I/Os, where  $K$  is the output size.*

The final algorithm is a combination of several ideas: filtering search (to charge the I/Os performed for a query to its output-size), optimal  $k$ -level shallow cuttings (which were not known before), roughly  $O(\log \log N)$  levels of shallow cuttings, and a cleverly chosen base-case. The non-trivial part of the analysis is bounding the number of I/Os to perform the so-called offline-find-any procedures (defined later).

To prove the above theorems, we make use of data structures with specific requirements. For example, in Theorem 4 data structures with fast construction I/Os are considered and in Theorem 5 a data structure with large additive error is considered. Although there exist data structures with desired bounds to handle some of these queries [11, 12, 18, 19], we are not aware of any prior work that obtains these results using a common framework, which might be useful for other problems.

► **Theorem 4** (Reporting, counting, and selection). *Let  $P$  be a set of  $N$  points in 3-D and  $\gamma$  be a fixed constant in  $(0, 1)$ . Then there exists a data structure that can be constructed in  $O(\text{sort}(N)) = O\left(\frac{N}{B} \log_{M/B} \left(\frac{N}{B}\right)\right)$  I/Os and then used to answer a*

- *3-D dominance reporting query in  $O((N/B)^\gamma + t/B)$  I/Os, where  $t$  is the number of points reported. Given a 3-D dominance query range  $q$ , the goal is to report the points in  $P \cap q$ .*
- *3-D dominance counting query in  $O((N/B)^\gamma)$  I/Os. Given a 3-D dominance query range  $q$ , the goal is to report  $|P \cap q|$ .*
- *3-D dominance  $x, y, z$ -selection query in  $O((N/B)^\gamma)$  I/Os. In an  $x$ -selection query, given a query point  $q = (q_x, q_y, q_z)$  and an integer  $k' \in [1, N]$ , the goal is to return a point  $q_1 = (q'_x, q_y, q_z)$  that dominates  $k'$  points in  $P$ . Analogously, in a  $y$ -selection query (resp.,  $z$ -selection query), the goal is to return a point  $q_2 = (q_x, q'_y, q_z)$  (resp.,  $q_3 = (q_x, q_y, q'_z)$ ) that dominates  $k'$  points in  $P$ .*

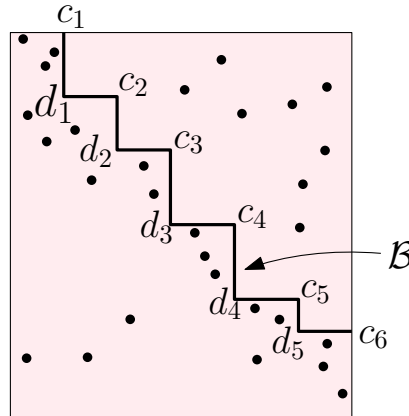
► **Theorem 5** (Offline additive error counting). *Let  $P$  be a set of  $N$  points in 3-D and  $Q$  be a set of 3-D dominance ranges. In the offline additive error counting problem, for each query  $q \in Q$ , the goal is to return a value  $n_q$  such that  $|P \cap q| - n_q \leq \varepsilon N^{2/3} B^{1/3}$ . The data structure can be constructed using  $O_\varepsilon(\text{sort}(N))$  I/Os and the queries are answered by using  $O_\varepsilon(\text{sort}(N) + \text{sort}(|Q|))$  I/Os.*

Our solution for additive error query can be extended to handle smaller values of error (such as  $\varepsilon \sqrt{NB}$ ). However, for our purpose an additive error of  $\varepsilon N^{2/3} B^{1/3}$  suffices.

The remaining paper is structured as follows. In Section 2, we discuss some preliminaries. In Section 3, we give an overview of two previous constructions of shallow cuttings for 3-D dominance ranges in internal memory. Then, in Section 4 we present our optimal-cost construction in the I/O-model. This is followed by two applications of our construction, offline 3-D dominance approximate counting (in Section 5) and offline 3-D dominance reporting (in the full version). In the full version, we also present the details of our supporting structures (Theorem 4 and Theorem 5) and potential directions for future work.

## 2 Preliminaries

**Definitions.** The *conflict list* of a cell  $C \in \mathcal{C}$  is defined as the points of  $P$  that are inside  $C$ . The *apex point* of a cell or a 3-D dominance range  $(-\infty, x] \times (-\infty, y] \times (-\infty, z]$  is defined as  $(x, y, z)$ . The *depth* of a point  $p = (x_p, y_p, z_p)$  w.r.t. a pointset  $P$  is the number of points of  $P$  lying in the cell  $(-\infty, x_p] \times (-\infty, y_p] \times (-\infty, z_p]$ .



■ **Figure 2** An example of a  $k$ -level 2-D shallow cutting. Here  $k = 3$ , so each inner corner  $d_i$  dominates three points and each outer corner  $c_i$  dominates six points.

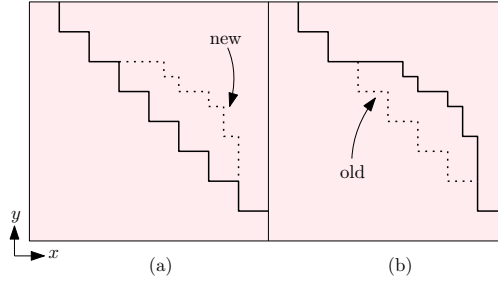
**Shallow cuttings for 2-D dominance ranges.** As a warm-up, we first introduce shallow cuttings for 2-D dominance ranges. Let  $P$  be a set of  $N$  points in 2-D. The collection  $\mathcal{C}$  of cells in the  $k$ -level shallow cutting of  $P$  will be of form  $(-\infty, x] \times (-\infty, y]$ . Consider the union of the cells in  $\mathcal{C}$ . We will focus on constructing the *outer boundary* of the union that is shown in Figure 2. The outer boundary  $\mathcal{B}$  is a 1-dimensional, monotone, and orthogonal curve. Specifically,  $\mathcal{B}$  starts from  $y = +\infty$  and consists of alternating vertical and horizontal segments. Let the sequence of endpoints of  $\mathcal{B}$  be  $c_1, d_1, c_2, d_2, \dots, d_{t-1}$  and  $c_t$ . The  $c_i$ 's are the *outer corners* and the  $d_i$ 's are the *inner corners*. The number of points of  $P$  dominated by any outer corner (resp., inner corner) will be at most  $2k$  (resp., equal to  $k$ ). The  $k$ -level shallow cutting for 2-D dominance ranges can be constructed in  $O(\text{sort}(N))$  I/Os. (Note that the  $2k$  bound is tighter than the  $10k$  bound we use for 3-D shallow cuttings.)

**Offline-find-any procedure.** The input to the *offline-find-any* problem is a collection of cells  $\mathcal{C}$  in the  $k$ -level shallow cutting of  $P$ , for any  $1 \leq k \leq n$ , and a set  $Q$  of 3-D query points. In the offline-find-any procedure, for each  $q \in Q$ , the goal is to find *any* cell in  $\mathcal{C}$  that contains  $q$  or report that none of them contain  $q$ . The distribution-sweeping approach in the survey paper of Vitter [21] can be used to solve this problem optimally.

► **Lemma 6.** Given a set  $\mathcal{C}$  of cells in 3-D and a set  $Q$  of query points in 3-D, the offline-find-any procedure requires  $O\left(\left(\frac{|\mathcal{C}|+|Q|}{B}\right) \log_{M/B} \frac{|\mathcal{C}|}{B}\right)$  I/Os.

## 3 Overview of previous construction algorithms

In this section we give an overview of a couple of previous constructions whose ideas will be required in our final solution.



■ **Figure 3** (a) Outer boundary (in bold) before the patching procedure. The dotted boundary is the new curve created during the patching procedure. (b) Outer boundary (in bold) after the patching procedure. The dotted curve was part of the outer boundary before patching. Each outer corner created during the patching procedure corresponds to the apex point of a 3-D cell in the  $k$ -level cutting of  $P$ .

### 3.1 Afshani and Tsakalidis's construction (AT-construction)

The problem becomes challenging in 3-D. At SODA'14, Afshani and Tsakalidis [5] obtained an optimal internal memory algorithm that constructs a  $k$ -level shallow cutting in  $O(N \log_2 N)$  time. Their algorithm constructs the outer boundary  $\mathcal{B}$  of the union of the cells in  $\mathcal{C}$ . In 3-D the outer boundary is a 2-dimensional, monotone, and orthogonal surface (see Figure 1(a)). The algorithm combines the sweep-plane approach with a method of maintaining a 2-D  $k$ -level shallow cutting under deletions. Let  $z_1, \dots, z_N$  be the sorted sequence of  $P$  in decreasing order of their  $z$ -coordinate values. Starting from  $z = z_1$ , a plane parallel to  $xy$ -plane is moved in the  $-z$  direction. The *invariant* in the algorithm is to maintain a 2-D shallow cutting for the  $xy$ -projections of points of  $P$  below the sweep-plane, by ensuring that the depth of each (inner or outer) corner in the 2-D shallow cutting lies in the range  $[k, 10k]$ . When the sweep-plane reaches  $z_i$ , then  $z_i$  is deleted and if the invariant is violated, then the 2-D shallow cutting of  $xy$ -projections of  $z_{i+1}, \dots, z_n$  is updated via a procedure called *patching*. See Figure 3. The precise details of the patching procedure are not necessary for our final algorithm.

The running time of their algorithm can be summarized by the following equation:

$$T_1(N) = \underbrace{T_{\text{sort}}(N)}_{\text{sorting}} + \underbrace{\Theta(N) \cdot T_{\text{pred}}(N/k)}_{\text{bottleneck 1}} + \underbrace{O(N/k) \cdot T_{\text{aux}}(N)}_{\text{bottleneck 2}} \quad (1)$$

There are two bottlenecks in adapting their algorithm in the I/O-model. At any point in the sweep-plane, let  $\mathcal{L}$  be the ordering of the corners in the 2-D shallow cutting based on their  $x$ -coordinate values. Each time a point  $z_i \in P$  is deleted, a predecessor search is performed on  $\mathcal{L}$ . Also, the patching procedure leads to insertion and deletion of corners into  $\mathcal{L}$ . The predecessor search is performed to identify all the corners in  $\mathcal{L}$  dominating  $z_i$  and then to update the conflict lists of those corners (at no extra asymptotic cost). This is the first bottleneck, since it is not possible to perform  $\Theta(N)$  updates and  $\Theta(N)$  predecessor queries to the list  $\mathcal{L}$  in  $\text{sort}(N)$  I/Os. (In fact, in [8] it was established that the easier problem of batched predecessor queries requires  $T_{\text{pred}}(N) = \Omega(\log_B N)$  I/Os per query when the preprocessing cost is bounded by a polynomial in  $N$ .) It is not clear how to reduce the number of predecessor queries to  $o(N)$ , and the lower-bound discards the standard strategy in the I/O-model of performing predecessor search queries in batches.

The *second bottleneck* is the lack of I/O-efficient data structures for performing the patching procedure. The patching procedure involves constructing data structures for 3-D dominance  $x$ -selection and dominance  $y$ -selection queries on  $P$ , and querying them  $O(N/k)$

times. For our purpose, the data structures have to be constructed in  $O(\text{sort}(N))$  I/Os. In the full version, we present such data structures with query I/Os  $T_{aux}(N) = O((N/B)^\gamma)$  for some  $\gamma \in (0, 1)$  (Theorem 4). Therefore, the patching will require  $O\left(\frac{N}{k} \left(\frac{N}{B}\right)^\gamma\right)$  I/Os, which is expensive.

### 3.2 Nekrich and Rahul's construction (NR-construction)

At SODA'23, Nekrich and Rahul [13] designed a data structure for 4-D dominance reporting query by constructing several shallow cuttings for 3-D dominance ranges. However, for their application, the conflict lists of the cells were not needed, and only the cells needed to be constructed. They modified the AT-construction to design an algorithm whose running time is proportional to the number of cells constructed (specifically  $O(N \log_2 N)$  cells were constructed on different subsets of  $P$  in  $O(N \log_2^5 N)$  time).

For a cell  $C$  in the shallow cutting with apex point  $(x, y, z)$ , let  $\text{exp}(C)$  be the largest  $z$ -coordinate such that  $(x, y, \text{exp}(C))$  dominates less than  $k$  points of  $P$ . Then  $\text{exp}(C)$  is defined as the *expiry* of  $C$ . For a cell  $C$ , its expiry  $\text{exp}(C)$  can be computed via a 3-D dominance  $z$ -selection query on  $P$ . Similar to AT-construction approach, starting from  $z = +\infty$ , the algorithm sweeps a plane parallel to the  $xy$ -plane in the  $-z$  direction and the same invariant is maintained. The key difference is in the *event points* visited by the sweep-plane algorithm. In the NR-construction, the event points will be the  $\text{exp}(C)$  values of all cells  $C$  in the shallow cutting, which are stored in a simple priority-queue  $\mathcal{Q}$ .

At each time step, we find the event with the largest  $z$ -coordinate  $z_{\max}$  in  $\mathcal{Q}$  and set the  $z$ -coordinate of the sweep-plane to  $z_{\max}$ . When the cell  $C$  corresponding to  $z_{\max}$  expires, (inner and outer) corners corresponding to  $C$  have to be removed from the 2-D shallow cutting at  $z_{\max}$ . Therefore, the patching procedure is performed. After finishing the patching procedure, (a) the events corresponding to the new inner corners in the 2-D shallow cutting are inserted into  $\mathcal{Q}$  and the events corresponding to the deleted corners in the 2-D shallow cutting are removed from  $\mathcal{Q}$ , (b) for each newly created outer corner  $(x, y)$  in the patching procedure, a new cell with apex point  $(x, y, z_{\max})$  gets *created*. We define  $\text{cre}(C) = z_{\max}$  for each cell  $C$  created. The algorithm finishes when the  $z$ -coordinate of the sweep-plane reaches  $z = -\infty$ . We omit the details of the NR-construction algorithm that are not needed to obtain our result. The following lemma summarizes the properties of the NR-construction that are relevant for our algorithm.

► **Lemma 7.** *Nekrich and Rahul's construction [13] of  $k$ -level shallow cuttings for 3-D dominance ranges can be summarized as follows:*

1. *It involves construction of a priority queue and 3-D dominance  $x, y, z$ -selection data structures.<sup>1</sup>*
2. *The number of updates and queries on the priority-queue is  $O(N/k)$  and the number of 3-D dominance  $x, y, z$ -selection queries is  $O(N/k)$ .*
3. *In each selection query, if the query point is  $q = (q_x, q_y, q_z)$ , then the sweep-plane is at  $q_z$  and  $(q_x, q_y)$  is one of the corners in the 2-D shallow cutting of  $P$  at  $q_z$  after patching. The query integer  $k' \leq 10k$  will be such that for an  $x$ -selection query, the reported point  $(q'_x, q_y, q_z)$  will have  $q'_x > q_x$ . However, for the  $y$ -selection and the  $z$ -selection queries, the reported points  $(q_x, q'_y, q_z)$  and  $(q_x, q_y, q'_z)$ , respectively, will have  $q'_y < q_y$  and  $q'_z < q_z$ .*

<sup>1</sup> Nekrich and Rahul [13] originally used 3-D dominance range counting data structures as well, where the goal is to report  $|P \cap q|$  for a 3-D dominance range  $q$ . However, both queries are almost equivalent: we can answer a 3-D dominance range selection query using  $O(\log N)$  range counting queries (binary search). Conversely, we can also answer a 3-D dominance range counting query via  $O(\log N)$  dominance  $z$ -selection queries.

The running time of the NR-construction can be summarized as follows:

$$T_2(N) = \underbrace{T_Q\left(\frac{N}{k}\right)}_{\text{queue}} + \underbrace{O\left(\frac{N}{k}\right) \cdot T_{sel}(N)}_{\text{bottleneck}} \quad (2)$$

Similar to the second bottleneck in the AT-construction, the bottleneck in the NR-construction is the I/Os required to perform the dominance selection queries (with  $O(\text{sort}(N))$  construction I/Os for the data structures, we have  $O\left(\frac{N}{k}\right) \cdot T_{sel}(N) = O\left(\frac{N}{k} \left(\frac{N}{B}\right)^\gamma\right)$  I/Os which is expensive).

### 3.3 Other constructions

At ICALP'14, Afshani, Chan, and Tsakalidis [3] proposed an AT-construction variant (by sweeping upwards) for constructing shallow cuttings for 3-D dominance ranges in  $O(N \log_2 \log_2 N)$  time in the word-RAM model. At STOC'96, Vengroff and Vitter [20] employed a structure that is similar to 3-D shallow cuttings to answer 3-D orthogonal range reporting queries. Unfortunately, the number of I/Os required to construct their data structure was not considered in their work.

Construction of shallow cuttings for halfspaces in 3-D has been another active area of research. However, typically the techniques and tools used for these problems (such as vertical decomposition,  $\varepsilon$ -cuttings, planar graph separators) have been different from the ones used for 3-D dominance ranges. For example, at SoCG'15, Chan and Tsakalidis [10] presented an optimal deterministic internal memory algorithm for the construction of shallow cuttings for halfspaces in 3-D. Their hierarchical construction relies on the fact that the cells of the shallow cutting do not intersect (to be precise, they are interior-disjoint). While this is true in the case of shallow cuttings for 3-D halfspaces, this property is not satisfied in our problem: cells of a  $k$ -level shallow cutting for 3-D dominance ranges are intersecting boxes. Also, in their hierarchy the levels go down by a constant factor which is not ideal for an I/O-efficient algorithm, whereas in our construction levels will go down at a faster rate.

## 4 Our algorithm for constructing $k$ -level shallow cuttings

We are now ready to prove Theorem 1. Our algorithm follows the general approach used in [5, 13] and combines it with several other ideas. First, we construct a *hierarchy* of shallow cuttings such that the conflict list size of cells decreases exponentially. Second, somewhat counterintuitively, we use data structures with high query cost and fast pre-processing for 3-D dominance selection queries. The final algorithm is a combination of six pieces.

### 4.1 1st piece: Hierarchy of shallow cuttings

The main idea is to construct a hierarchy of shallow cuttings. We want to construct a  $k$ -level shallow cutting of  $P$  where  $k \geq M/c$  and  $c$  is a sufficiently large constant. Let  $t_i = (N/B)^{\delta^i}$  and  $k_i = \max(Bt_i, k)$ . We select the constant  $\delta \in (0, 1)$  in such way that  $k_i \leq k_{i-1}/10$  (for any  $i$  satisfying  $k_i \geq M/c$ ). The algorithm works in stages (starting from stage 0). In the  $i$ -th stage, we construct the  $k_i$ -level shallow cutting. For the starting case of 0-th stage, since  $k_0 = N$ , the  $k_0$ -level shallow cutting consists of one cell containing all points of  $P$ . The algorithm terminates when  $k_i = k$ .

► **Lemma 8.** *The number of shallow cuttings constructed is  $r = O(\log_{M/B}(N/B))$ .*

**Proof.** Since  $k_i = Bt_i \geq M/c$ , we have  $t_i \geq M/Bc$ . Therefore,  $t_{i+1} = t_i^\delta = t_i \cdot \frac{1}{t_i^{1-\delta}} \leq \frac{t_i}{(M/cB)^{1-\delta}}$ . Unfolding the recursion, we obtain  $t_{i+1} \leq \frac{t_0}{(\frac{M}{cB})^{(i+1)(1-\delta)}}$ . Setting  $i \leftarrow \frac{c'}{1-\delta} \log_{M/cB}(N/B) - 1$ , for  $c' \geq 2$ , we observe that  $t_{i+1} \leq t_0/(N/B)^{c'} < 1$ . This implies that  $r = O(\frac{1}{1-\delta} \log_{M/B}(N/B)) = O(\log_{M/B}(N/B))$ . ◀

## 4.2 2nd piece: Existence of parent cells

We will start with some definitions. For all  $1 \leq i \leq r$ , we will say that the  $k_{i-1}$ -level shallow cutting is the *parent* shallow cutting of the  $k_i$ -level shallow cutting. When the sweep-plane reaches  $z$ , then a cell  $C$  is *active* at  $z$  if  $cre(C) \leq z \leq exp(C)$ .

In the construction of the  $k_i$ -level shallow cutting, consider an outer corner  $(x_p, y_p)$  on the 2-D shallow cutting which got created when the sweep-plane is at position  $z$ . Then the *parent cell* of point  $p = (x_p, y_p, z)$  is the cell  $C$  in the parent shallow cutting which satisfies the following two properties: (a) the apex point of  $C$  dominates  $p$ , and (b) the apex point of  $C$  has the smallest  $y$ -coordinate among all cells that are active at  $z$ .

► **Lemma 9.** *In the construction of the  $k_i$ -level shallow cutting (for  $i \geq 1$ ), consider an outer corner  $(x_p, y_p)$  on the 2-D shallow cutting which got created when the sweep-plane is at position  $z$ . Then a parent cell always exists for the point  $(x_p, y_p, z)$ .*

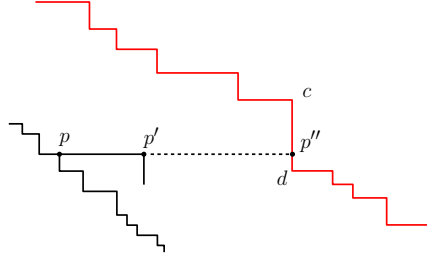
**Proof.** The depth of the point  $(x_p, y_p, z)$  will be less than or equal to  $10k_i$  (an invariant maintained by the AT-construction), which is at most  $k_{i-1}$ . In the  $k_{i-1}$ -level shallow cutting construction, consider the 2-D shallow cutting when the sweep-plane reached  $z$ . This 2-D shallow cutting will have at least one outer corner, say  $p'$ , which dominates  $(x_p, y_p)$  (by third property of  $k_{i-1}$ -level shallow cuttings). Let  $C$  be the 3-D cell in the  $k_{i-1}$ -level shallow cutting corresponding to the outer corner  $p'$ . Since  $C$  is active when the sweep-plane reached position  $z$ , it implies that the apex point of  $C$  dominates  $(x_p, y_p, z)$ . This establishes property (a) of a parent cell, and hence, a parent cell exists. ◀

## 4.3 3rd piece: Query-expensive structures on small-sized inputs

A data structure built on  $N'$  points is query-expensive if the the number of I/Os required to answer a query is of the form  $O((N'/B)^\gamma)$  or  $O((N'/B)^\gamma + t/B)$ . Our next idea is to construct such query-expensive data structures, but only on small-sized inputs. Specifically, for the  $k_i$ -level shallow cutting, we will ensure that the a 3-D dominance  $x, y, z$ -selection query requires only  $O(k_i/B)$  I/Os, and since the number of such selection queries performed is bounded by  $O(N/k_i)$  (point 2 in Lemma 7), overall the selection queries will require only  $O(N/B)$  I/Os. Similarly, we will ensure that generating the conflict list of a cell requires only  $O(k_i/B)$  I/Os, and since there are  $O(N/k_i)$  cells in the  $k_i$ -level cutting, generating all the conflict lists will require only  $O(N/B)$  I/Os. The existence of parent cells and the following lemma will let us achieve the goal of utilizing query-expensive data structures I/O-efficiently.

► **Lemma 10.** *In the construction of the  $k_i$ -level shallow cutting (for  $i \geq 1$ ), consider the sweep-plane at any time  $z$ . If patching is performed at time  $z$ , then let  $p = (x_p, y_p)$  be a new outer corner created with parent cell  $C$ . Then the 3-D dominance  $x, y, z$ -selection queries with query point  $p = (x_p, y_p, z)$  will output a point that lies inside  $C$ .*

**Proof.** This is trivial to see for a  $y$ -selection query since  $p$  already lies inside  $C$  and hence, the point reported by the query will also lie inside  $C$  (see Lemma 7). Similarly, it holds for a  $z$ -selection query as well. The non-trivial case is the  $x$ -selection query.



■ **Figure 4** An example of  $x$ -selection query in Lemma 10. The point  $p$  is an outer corner of a 2-D shallow cutting of the  $k_i$ -level. The corner  $c$  is the outer corner of the parent staircase corresponding to the parent cell  $C$  of  $p$ , since  $c$  has the smallest  $y$ -coordinate among all apex points of the parent staircase dominating  $p$ .

Let  $p''$  denote the point where the horizontal ray from  $p$  hits the 2-D shallow cutting of the parent at time  $z$ . See Figure 4. Let  $d$  be the inner corner immediately below  $p''$  and  $c$  be the outer corner immediately above  $p''$ . Since the depth of  $d$  is at least  $k_{i-1}$ , the depth of  $p''$  will also be at least  $k_{i-1}$ . Consider the 3-D dominance  $x$ -selection query with query point  $p$  and parameter  $k' \leq 10k_i \leq k_{i-1}$ . This implies that the output,  $p'$ , will have an  $x$ -coordinate less than or equal to  $p''$ . Now the cell corresponding to the outer corner  $c$  is the parent cell of  $p$  and contains  $p'$  as well. ◀

► **Lemma 11.** Assume that the  $k_{i-1}$ -level shallow cutting of  $P$  has been constructed in the  $(i-1)$ -th stage. Ignoring the I/Os required to find the parent cells, in the  $i$ -th stage, the  $k_i$ -level shallow cutting can be constructed in  $O\left(\frac{N}{k_i} \log_B \frac{N}{k_i} + \frac{N}{B} \log_{M/B} \frac{k_{i-1}}{B}\right)$  I/Os.

**Proof.** For each cell in the  $k_{i-1}$ -level shallow cutting, based on its conflict list we construct data structures supporting 3-D dominance reporting and 3-D dominance  $x, y, z$ -selection queries (Theorem 4) that requires  $O((N/k_{i-1}) \cdot \frac{k_{i-1}}{B} \log_{M/B}(\frac{k_{i-1}}{B})) = O\left(\frac{N}{B} \log_{M/B} \frac{k_{i-1}}{B}\right)$  I/Os. By point 2 in Lemma 7,  $O(N/k_i)$   $x, y, z$ -selection queries will be performed. For each selection query, if the cost of finding the parent cell of the query point is ignored, then by Lemma 10 the answer to the query can be obtained by querying the  $x, y, z$ -selection data structure built at the parent cell. The query I/Os of each selection query will then be  $O((k_{i-1}/B)^\gamma)$ . Therefore, the number of I/Os performed to answer the selection queries is  $O\left(\frac{N}{k_i} \left(\frac{k_{i-1}}{B}\right)^\gamma\right) = O\left(\frac{N}{B t_i} \cdot t_{i-1}^\gamma\right) = O\left(\frac{N}{B t_i} \cdot t_{i-1}^\delta\right) = O\left(\frac{N}{B}\right)$ , by choosing  $\gamma < \delta$ .

By point 1 in Lemma 7, the number of operations on the priority-queue are  $O(N/k_i)$  and using a vanilla B-tree as a priority-queue  $O\left(\frac{N}{k_i} \log_B \frac{N}{k_i}\right)$  I/Os are performed.

Finally, for each cell  $C$  in the  $k_i$ -level cutting, its conflict list is generated by querying its parent cell's 3-D dominance reporting data structure. The number of I/Os performed is  $O\left(\left(\frac{k_{i-1}}{B}\right)^\gamma + \frac{k_i}{B}\right) = O(k_i/B)$ . Since the number of cells in the  $k_i$ -level cutting is  $O(N/k_i)$ , the total number of I/Os required to generate the conflict lists is  $O(N/B)$ . ◀

#### 4.4 4th piece: Finding parent cells efficiently

By point 2 in Lemma 7,  $O(N/k_i)$   $x, y, z$ -selection queries are performed in the construction of a  $k_i$ -level shallow cutting. To perform each selection query, we first need an efficient routine to find the parent cell. Once the parent cell is found, then the selection query is performed on the data structure corresponding to the parent cell.

We will use partially persistent B-trees [7] to store the cells in the  $k_{i-1}$ -level shallow cutting. Versions of the partially persistent data structure are parameterized by  $z$ -coordinates: a cell is inserted (resp. deleted) at time  $t$  if the corresponding outer corner is added to the 2-D shallow cutting (resp. removed from the 2-D shallow cutting) when the  $z$ -coordinate of the sweep-plane is equal to  $t$ . This data structure supports successor queries with respect to  $y$ -coordinates of its apex points: for any pair  $(z_q, y_q)$  we can find the cell with the smallest  $y$ -coordinate among all cells of the parent shallow cutting that are active at time  $z_q$  and whose  $y$ -coordinate is not smaller than  $y_q$ .

► **Lemma 12.** *In the construction of  $k_i$ -level shallow cutting, the number of I/Os performed to compute the parent cells is  $\left(\frac{N}{k_i} \log_B \frac{N}{k_{i-1}}\right)$ .*

**Proof.** The number of cells in the  $k_{i-1}$ -level shallow cutting is  $O(N/k_{i-1})$  and using persistent B-trees, a successor query and an update query can be supported in  $O(\log_B(N/k_{i-1}))$  I/Os. By point 2 in Lemma 7, the total number of  $x, y, z$ -selection queries will be  $O(N/k_i)$ , and hence, the number of I/Os performed to compute the parent cells is  $\left(\frac{N}{k_i} \log_B \frac{N}{k_{i-1}}\right)$ . ◀

#### 4.5 5th piece: Obtaining logarithmic bounds with the base $M/B$

To obtain  $O(\text{sort}(N))$  I/Os bound, we need the base in the logarithm to be  $M/B$ . To circumvent the fact that finding parent cells and operations on the priority queue require  $O(\log_B(N/k))$  query I/Os, we exploit the fact that  $k$  is large.

► **Lemma 13.** *For  $k = \Omega(M)$ , we have  $O\left(\frac{N}{k} \log_B \frac{N}{k}\right) = O\left(\frac{N}{B} \log_{M/B} \frac{N}{B}\right)$ .*

**Proof.** First, we will establish that  $B \log(M/B) \leq cM \log B$ , where  $c$  is a sufficiently large constant. When  $M \leq B^c$ , we have  $cM \log B = M \log B^c \geq M \log M \geq B \log(M/B)$ . When  $M > B^c$ , we have  $cM \log B \geq cB \cdot M^{1-1/c} \log B \geq cB \cdot M^{1-1/c} \geq cB \log M \geq cB \log(M/B)$ . Using this fact, we have

$$\frac{N}{k} \cdot \frac{\log(N/k)}{\log B} \leq \frac{N}{M} \cdot \frac{\log(N/B)}{\log B} \leq O\left(\frac{N}{B} \log_{M/B} \frac{N}{B}\right). \quad \blacktriangleleft$$

In summary, the running time of our algorithm is

$$T_3(N) = O\left(T_{\mathcal{Q}}\left(\frac{N}{k}\right) + \sum_{i=1}^r \frac{N}{k_i} \cdot T_{\text{sel}}(k_{i-1}) + \frac{N}{k} \cdot \log_B \frac{N}{k}\right) \quad (3)$$

► **Lemma 14.** *For  $k = \Omega(M)$ , the  $k$ -level shallow cutting for 3-D dominance ranges on  $N$  points can be constructed in  $O\left(\frac{N}{B} \log_{M/B} \left(\frac{N}{B}\right)\right)$  I/Os.*

**Proof.** By combining Lemma 11 and Lemma 12, the number of I/Os required to construct a  $k_i$ -level shallow cutting is  $O\left(\frac{N}{k_i} \log_B \frac{N}{k_i} + \frac{N}{B} \log_{M/B} \frac{k_{i-1}}{B}\right)$ . Therefore,

$$\sum_{i=1}^r O\left(\frac{N}{k_i} \log_B \frac{N}{k_i} + \frac{N}{B} \log_{M/B} \frac{k_{i-1}}{B}\right) = O\left(\frac{N}{k} \log_B \frac{N}{k} + \frac{N}{B} \log_{M/B} \frac{N}{B}\right)$$

By Lemma 13, this can be bounded by  $O\left(\frac{N}{B} \log_{M/B} \frac{N}{B}\right)$ . ◀

#### 4.6 Final piece: Handling Small Values of $k$

We will need a different procedure to handle  $k = O(M)$ . We will perform a two-level cutting. First, we will construct an  $(M/c)$ -level shallow cutting of  $P$  using Lemma 14. For each cell, say  $C$ , in the  $(M/c)$ -level, load its *entire* conflict list into the internal memory. This can be done I/O-efficiently since the size of the conflict list is  $O(M)$ . Next run an internal memory algorithm to compute the  $k$ -level shallow cutting of the conflict list of  $C$ . The final output is the union of the shallow cuttings obtained from each cell in the  $(M/c)$ -level.

► **Lemma 15.** *Suppose that we are given an  $(M/c)$ -level shallow cutting of  $P$  for some constant  $c > 1$ . Then for  $k < M/c$ , the  $k$ -level shallow cutting can be constructed in  $O(N/B)$  I/Os.*

**Proof.** We will establish that the three properties of a  $k$ -level shallow cutting of  $P$  are satisfied by the algorithm. The number of cells constructed in the final cutting is  $O(N/M) \cdot O(M/k) = O(N/k)$  (first property). It is easy to verify that the number of points of  $P$  lying inside each cell in the final cutting is  $O(k)$  (second property). Consider a point  $q$  that dominates less than or equal to  $k$  points in  $P$ . By the third property of  $M/c$ -level shallow cutting of  $P$ , there exists a cell  $C$  that contains  $q$ . The number of points in the conflict list of  $C$  dominated by  $q$  will still be less than or equal to  $k$ . Therefore, in the  $k$ -level shallow cutting of the conflict list of  $C$ , there will exist some cell that contains  $q$  (third property).

Loading the conflict lists of all the cells in the  $(M/c)$ -level shallow cutting requires  $O(N/M) \cdot O(M/B) = O(N/B)$  I/Os. For each cell in the  $(M/c)$ -level shallow cutting, writing the  $k$ -level shallow cutting of its conflict-list into external memory requires  $O(M/B)$  I/Os. Thus, over all the cells in the  $(M/c)$ -level shallow cutting, it will require  $O(N/M) \cdot O(M/B) = O(N/B)$  I/Os. ◀

Combining Lemma 14 and Lemma 15 completes the proof of Theorem 1.

### 5 Application-1: Offline 3-D dominance approximate counting

Recall that  $P$  is a set of  $N$  points in 3-D and  $Q$  is a set of query 3-D dominance ranges. Consider a parameter  $\varepsilon$  with a fixed value in  $(0, 1)$ . In the offline 3-D dominance approximate counting problem, the goal is to report a value in the range  $[(1 - \varepsilon)|P \cap q|, |P \cap q|]$ , for all  $q \in Q$ . We will prove Theorem 2 in this section.

#### 5.1 Data structure and query algorithm

The overall data structure is recursive and we start at the root with pointset  $P$ . Construct a  $(N^{2/3}B^{1/3})$ -level cutting  $\mathcal{C}$  based on  $P$  (Theorem 1). Also, construct an additive error counting structure based on  $P$  (Theorem 5). Finally, for each cell  $C \in \mathcal{C}$ , recurse on  $P_C$ , where  $P_C$  is the conflict list of  $C$ . The recursion stops when the size of pointset becomes less than or equal to  $M/c$ , where  $c$  is a sufficiently large constant.

The query algorithm will start from the root of the data structure. A point in  $Q$  is *deep* if no cell in  $\mathcal{C}$  contains it; otherwise, it is *shallow*. Using the offline-find-any procedure on  $\mathcal{C}$  and  $Q$ , we will classify each point in  $Q$  as shallow or deep. The result for the deep query points are obtained by querying the additive error counting structure. If a point  $q \in Q$  is shallow, then it is *assigned* to one of the cells containing it. The shallow points are handled recursively. Specifically, for each cell  $C \in \mathcal{C}$ , recurse on  $Q_c$ , where  $Q_c \subseteq Q$  is the set of shallow query points assigned to  $C$ .

To handle the base case, where  $|P| \leq M/c$ , first the points in  $P$  are loaded into the memory. Then the query objects are streamed into the main memory one block at a time, and the results for these queries are obtained without any additional I/Os and written to the external memory.

## 5.2 Analysis

We will first bound the number of I/Os performed by the algorithm.

► **Lemma 16.** *The number of I/Os required to construct the data structure is  $O\left(\frac{|P|}{B} \left(\log_{M/B} \frac{|P|}{B}\right)^{O(1)}\right)$ .*

**Proof.** The  $(N^{2/3}B^{1/3})$ -level cutting (Theorem 1) and the additive error counting structure (Theorem 5) can be constructed in  $O(\text{sort}(|P|))$  I/Os. Therefore, the total number of I/Os required to construct the data structure is:

$$T(N) = \begin{cases} c_1 \left(\frac{N}{B}\right)^{1/3} \cdot T(N^{2/3}B^{1/3}) + \frac{c_2 N}{B} \log_{M/B} \frac{N}{B}, & \text{if } N \geq M/c \\ O(N/B), & \text{otherwise} \end{cases}$$

where  $c_1(N/B)^{1/3}$  is the number of cells in the cutting and  $c_2$  is the constant inside  $O(\text{sort}(N))$ . We will establish that  $T(N) \leq \frac{c_3 N}{B} \left(\log_{M/B} \frac{N}{B}\right)^{c_4}$ , where  $c_3 = 2c_2$  and  $c_1 \left(\frac{2}{3}\right)^{c_4} = \frac{1}{2}$ . As such,

$$\begin{aligned} T(N) &\leq c_1 \left(\frac{N}{B}\right)^{1/3} \cdot c_3 \frac{N^{2/3}B^{1/3}}{B} \left(\log_{M/B} \frac{N^{2/3}B^{1/3}}{B}\right)^{c_4} + \frac{c_2 N}{B} \log_{M/B} \frac{N}{B} \\ &\leq c_1 c_3 \frac{N}{B} \cdot \left(\frac{2}{3}\right)^{c_4} \left(\log_{M/B} \frac{N}{B}\right)^{c_4} + \frac{c_2 N}{B} \log_{M/B} \frac{N}{B} \\ &\leq \frac{c_2 N}{B} \left(\log_{M/B} \frac{N}{B}\right)^{c_4} + \frac{c_2 N}{B} \left(\log_{M/B} \frac{N}{B}\right)^{c_4} = \frac{c_3 N}{B} \left(\log_{M/B} \frac{N}{B}\right)^{c_4} \quad \blacktriangleleft \end{aligned}$$

► **Lemma 17.** *The number of I/Os performed by the query algorithm is  $O_\varepsilon \left(\frac{|P|}{B} \left(\log_{M/B} \frac{|P|}{B}\right)^{O(1)} + \frac{|Q|}{B} \log_{M/B} \frac{N}{B}\right)$ .*

**Proof.** By Theorem 5 and Lemma 6, the number of I/Os performed by the query algorithm at the root is  $O_\varepsilon \left(\text{sort}(N) + \text{sort}(|Q|) + \left(\frac{|C|+|Q|}{B}\right) \log_{M/B} \frac{|C|}{B}\right)$ , which can be bounded by  $O_\varepsilon \left(\text{sort}(N) + \frac{|Q|}{B} \log_{M/B} \frac{N}{B}\right)$ . The  $O_\varepsilon(\text{sort}(N))$  I/Os can be charged to the construction of the data structure, and hence, by the proof of Lemma 16, it will be bounded by  $O_\varepsilon \left(\frac{|P|}{B} \left(\log_{M/B} \frac{|P|}{B}\right)^{O(1)}\right)$  in the overall query algorithm.

Therefore, we will focus on the  $O_\varepsilon \left(\frac{|Q|}{B} \log_{M/B} \frac{N}{B}\right)$  term. As such, the amortized number of I/Os performed for a *single* query point is  $O_\varepsilon \left(\frac{1}{B} \log_{M/B} \frac{N}{B}\right)$ . Including recursion, the number of I/Os performed to answer a single query is:

$$q(N) \leq q(N^{2/3}B^{1/3}) + O_\varepsilon \left(\frac{1}{B} \log_{M/B} \frac{N}{B}\right) = O_\varepsilon \left(\frac{1}{B} \log_{M/B} \frac{N}{B}\right),$$

which holds because we end up with a geometrically decreasing series. ◀

► **Lemma 18.** For each  $q \in Q$ , the algorithm returns a  $(1 - \varepsilon)$ -approximation of  $|P \cap q|$ .

**Proof.** For any deep query point  $q$  at the root, the value  $n_q$  reported by the additive error counting structure will be such that  $n_q \geq |P \cap q| - \varepsilon N^{2/3} B^{1/3} \geq (1 - \varepsilon)|P \cap q|$ , since  $|P \cap q| \geq N^{2/3} B^{1/3}$  for a deep point (third property of shallow cuttings). The same argument applies at each node in the data structure (w.r.t. deep queries at that node). ◀

---

## References

- 1 Peyman Afshani. On dominance reporting in 3D. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 41–51, 2008. doi:10.1007/978-3-540-87744-8\_4.
- 2 Peyman Afshani, Lars Arge, and Kasper Dalgaard Larsen. Orthogonal range reporting in three and higher dimensions. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 149–158, 2009. doi:10.1109/FOCS.2009.58.
- 3 Peyman Afshani, Timothy M. Chan, and Konstantinos Tsakalidis. Deterministic rectangle enclosure and offline dominance reporting on the RAM. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 77–88, 2014. doi:10.1007/978-3-662-43948-7\_7.
- 4 Peyman Afshani, Chris H. Hamilton, and Norbert Zeh. A general approach for cache-oblivious range reporting and approximate range counting. *Computational Geometry*, 43(8):700–712, 2010. doi:10.1016/J.COMGEO.2010.04.003.
- 5 Peyman Afshani and Konstantinos Tsakalidis. Optimal deterministic shallow cuttings for 3d dominance ranges. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1389–1398, 2014. doi:10.1137/1.9781611973402.102.
- 6 Pankaj K. Agarwal, Lars Arge, Jeff Erickson, Paolo Giulio Franciosa, and Jeffrey Scott Vitter. Efficient searching with linear constraints. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 169–178, 1998. doi:10.1145/275487.275506.
- 7 Bruno Becker, Stephan Gschwind, Thomas Ohler, Bernhard Seeger, and Peter Widmayer. An asymptotically optimal multiversion B-tree. *The VLDB Journal*, 5(4):264–275, 1996. doi:10.1007/S007780050028.
- 8 Michael A. Bender, Martin Farach-Colton, Mayank Goswami, Dzejla Medjedovic, Pablo Montes, and Meng-Tsung Tsai. The batched predecessor problem in external memory. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 112–124, 2014. doi:10.1007/978-3-662-44777-2\_10.
- 9 Timothy M. Chan, Yakov Nekrich, Saladi Rahul, and Konstantinos Tsakalidis. Orthogonal point location and rectangle stabbing queries in 3-d. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 31:1–31:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.31.
- 10 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. In *International Symposium on Computational Geometry (SoCG)*, pages 719–732, 2015. doi:10.4230/LIPIcs.SOCG.2015.719.
- 11 Roberto Grossi and Giuseppe F. Italiano. Efficient cross-trees for external memory. In *External Memory Algorithms*, volume 50, pages 87–106. DIMACS/AMS, 1998. doi:10.1090/DIMACS/050/04.
- 12 Kothuri Venkata Ravi Kanth and Ambuj K. Singh. Optimal dynamic range searching in non-replicating index structures. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 257–276, 1999. doi:10.1007/3-540-49257-7\_17.
- 13 Yakov Nekrich and Saladi Rahul. 4d range reporting in the pointer machine model in almost-optimal time. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1862–1876, 2023. doi:10.1137/1.9781611977554.CH71.
- 14 Mihai Patrascu. Succincter. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 305–313, 2008. doi:10.1109/FOCS.2008.83.

- 15 Saladi Rahul. Improved bounds for orthogonal point enclosure query and point location in orthogonal subdivisions in  $\mathbb{R}^3$ . In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 200–211, 2015. doi:10.1137/1.9781611973730.15.
- 16 Saladi Rahul and Yufei Tao. On top-k range reporting in 2d space. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 265–275, 2015. doi:10.1145/2745754.2745777.
- 17 Saladi Rahul and Yufei Tao. Efficient top-k indexing via general reductions. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 277–288, 2016. doi:10.1145/2902251.2902290.
- 18 John T. Robinson. The K-D-B-tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 10–18, 1981. doi:10.1145/582318.582321.
- 19 Yufei Tao. Indexability of 2d range search revisited: constant redundancy and weak indivisibility. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 131–142, 2012. doi:10.1145/2213556.2213577.
- 20 Darren Erik Vengroff and Jeffrey Scott Vitter. Efficient 3-d range searching in external memory. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 192–201, 1996. doi:10.1145/237814.237864.
- 21 Jeffrey Scott Vitter. Algorithms and data structures for external memory. *Foundation and Trends in Theoretical Computer Science*, 2(4):305–474, 2006. doi:10.1561/04000000014.