Range closest-pair search in higher dimensions*

Timothy M. Chan[1], Saladi Rahul[2], and Jie Xue[3]

[1] University of Illinois at Urbana-Champaign, Urbana, IL, USA
tmc@illinois.edu
[2] University of Illinois at Urbana-Champaign, Urbana, IL, USA
saladi.rahul@gmail.com
[3] University of Minnesota - Twin Cities, Minneapolis, Minnesota, USA
xuexx193@umn.edu

Abstract. Range closest-pair (RCP) search is a range-search variant of the classical closest-pair problem, which aims to store a given set S of points into some space-efficient data structure such that when a query range Q is specified, the closest pair in $S \cap Q$ can be reported quickly. RCP search has received attention over years, but the primary focus was only on $\mathbb{R}^2$. In this paper, we study RCP search in higher dimensions. We give the first nontrivial RCP data structures for orthogonal, simplex, halfspace, and ball queries in $\mathbb{R}^d$ for any constant d. Furthermore, we prove a conditional lower bound for orthogonal RCP search for $d \geq 3$.

1 Introduction

The closest-pair problem is one of the most fundamental problems in computational geometry and finds numerous applications in various areas, such as collision detection, traffic control, etc. In many scenarios, instead of finding the global closest-pair, people want to know the closest pair contained in some specified ranges. This results in the notion of *range closest-pair* (RCP) search. RCP search is a range-search variant of the classical closest-pair problem, which aims to store a given set S of points into some space-efficient data structure such that when a query range Q is specified, the closest pair in $S \cap Q$ can be reported quickly. RCP search has received considerable attention over the years [1, 4, 10, 11, 17, 18, 21, 20, 22, 23].

Unlike most traditional range-search problems, RCP search is *non-decomposable*. That is, if we partition the dataset S into S_1 and S_2, given a query range Q, the closest pair in $S \cap Q$ cannot be obtained efficiently from the closest pairs in $S_1 \cap Q$ and $S_2 \cap Q$. Due to the non-decomposability, many traditional range-search techniques are inapplicable to RCP search, which makes the problem quite challenging. As such, despite of much effort made on this topic, most known results

* A full version of the paper is available at [6]. Work by the first author has been partially supported by NSF Grant CCF-1814026. Work by the third author has been partially supported by a Doctoral Dissertation Fellowship from the Graduate School of the University of Minnesota.

are restricted to the plane case, i.e., RCP search in $\mathbb{R}^2$. Beyond $\mathbb{R}^2$, only very specific query types have been studied, such as 2-sided box queries.

In this paper, we investigate RCP search in higher dimensions. We consider four widely-studied query types: orthogonal queries, simplex queries, halfspace queries, and ball queries. We are interested in designing efficient RCP data structures (in terms of space cost, query time, and preprocessing time) for these kinds of query ranges, and proving conditional lower bounds for these problems.

Related work. The closest-pair problem and range search are both well-studied problems in computational geometry; see [2, 19] for surveys of these two topics.

RCP search was for the first time introduced by Shan et al. [17] and subsequently studied in [1, 4, 10, 11, 18, 21, 20, 22, 23]. In $\mathbb{R}^2$, the query types studied include quadrants, strips, rectangles, and halfplanes. RCP search with these query ranges can be solved using near-linear space with poly-logarithmic query time. The best known data structures were given by Xue et al. [22], and we summarize the bounds in Table 1. For *fat* rectangles queries (i.e., rectangles of constant aspect ratio), Bae and Smid [4] showed an improved RCP data structure using $O(n \log n)$ space and $O(\log n)$ query time. In a recent work [20], Xue considered a colored version of RCP search in which the goal is to report the bichromatic closest pair contained in a query range, and proposed efficient data structures for orthogonal colored approximate RCP search (mainly in $\mathbb{R}^2$).

Query type	Space cost	Query time	Preprocessing time
Quadrant	$O(n)$	$O(\log n)$	$O(n \log^2 n)$
Strip	$O(n \log n)$	$O(\log n)$	$O(n \log^2 n)$
Rectangle	$O(n \log^2 n)$	$O(\log^2 n)$	$O(n \log^7 n)$
Halfplane	$O(n)$	$O(\log n)$	$O(n \log^2 n)$

Table 1. Best known results in $\mathbb{R}^2$

Beyond $\mathbb{R}^2$, the problem is quite open. To our best knowledge, the only known results are the orthogonal RCP data structure given by Gupta et al. [10] which only has guaranteed average-case performance and the approximate colored RCP data structures given by Xue [20] which can only handle restricted query types (dominance query in $\mathbb{R}^3$ and 2-sided box query in $\mathbb{R}^d$).

A key ingredient in existing solutions for RCP search in $\mathbb{R}^2$ is the *candidate-pair* method. Roughly speaking, this method tries to show that among the $\Omega(n^2)$ point pairs, only a few (called *candidate pairs*) can be the answer of some query. If this can be shown, then it suffices to store the candidate pairs and search the answer among them. Unfortunately, it is quite difficult to generalize this method to higher dimensions, as the previous approaches for proving the number of candidate pairs heavily rely on the fact that the data points are given in the plane. This might be the main reason why RCP search can be efficiently solved in $\mathbb{R}^2$, while remaining open in higher dimensions.

Our contributions. In this paper, we give the first non-trivial RCP data structures for orthogonal, simplex, halfspace, and ball queries in $\mathbb{R}^d$, for any constant d. The performances of our new data structures are summarized in Table 2, where the notation $\tilde{O}(\cdot)$ hides $\log n$ factors. All these data structures have near-linear

space cost, sub-linear query time, and sub-quadratic preprocessing time. For example, we obtain $\tilde{O}(n^{7/8})$ query time for two-dimensional triangular ranges, and $\tilde{O}(n^{2/3})$ query time for three-dimensional halfspaces and two-dimensional balls (i.e., disks).[4]

Furthermore, we complement these results by establishing a conditional lower bound, implying that our $\tilde{O}(\sqrt{n})$ query time bound for orthogonal RCP search in $\mathbb{R}^d$ for any $d \geq 3$ is likely the best possible (and in particular explaining why polylogarithmic solution seems not possible beyond two dimensions). Specifically, we show that orthogonal RCP search in $\mathbb{R}^3$ is at least as hard as the *set intersection query problem*, which is conjectured to require $\tilde{\Omega}(\sqrt{n})$ query time for linear-space structures.

Query type	Source	Space cost	Query time	Preprocessing time
Orthogonal	Theorem 1	$\tilde{O}(n)$	$\tilde{O}(\sqrt{n})$	$\tilde{O}(n\sqrt{n})$
Simplex	Theorem 3	$\tilde{O}(n)$	$\tilde{O}(n^{1-1/(2d^2)})$	$\tilde{O}(n^{(3d^2+1)/(2d^2+1)})$
Halfspace	Theorem 4	$\tilde{O}(n)$	$\tilde{O}(n^{1-1/(d\lfloor d/2 \rfloor)})$	$\tilde{O}(n^{2-1/(2d^2)})$
Ball	Full version [6]	$\tilde{O}(n)$	$\tilde{O}(n^{1-1/((d+1)\lceil d/2 \rceil)})$	$\tilde{O}(n^{2-1/(2(d+1)^2)})$

Table 2. Performances of our new RCP data structures in $\mathbb{R}^d$

Overview of our techniques. Our approach for designing these new data structures is quite different from those in the previous work. We avoid using the aforementioned candidate-pair method. Instead, our RCP data structures solve the problems as follows (roughly). For a given query range Q, the data structure first partitions the points in $S \cap Q$ into two subsets, say K and L. The size of L is guaranteed to be small, while K may have a large size. Then the data structure computes the closest pair ϕ in K using some pre-stored information and computes the closest pair ϕ' in L using the standard closest-pair algorithm (which can be done efficiently as L is small). If the two points of the closest pair ϕ^* in $S \cap Q$ are both in K or both in L, we are done. The only remaining case is that one point of ϕ^* is in K while the other point is in L. The data structure handles this case by finding the nearest neighbor of a in Q for every $a \in L$ via reporting all the points in Q that are "near" a. Using a packing argument, we can show that one only needs to report a constant number of points for each $a \in L$, and hence this procedure can be completed efficiently (since L is small).

To implement this strategy, we incorporate a number of existing geometric data structuring techniques. For orthogonal RCP, we use *range trees* and adapt an idea from Gupta et al. [10] of classifying nodes as "heavy" and "light" (originally for solving a different problem, two-dimensional orthogonal *range diameter*, in near-linear space and $\tilde{O}(\sqrt{n})$ query time). For simplex RCP, we use *simplicial partitions* instead of range trees. For halfspace RCP, we switch to dual space and use *cuttings*, similar to an idea from Chan et al. [7] (for solving a different problem, halfspace *range mode*, in near-linear space and $\tilde{O}(n^{1-1/d^2})$ time). Overall, the combination of existing and new ideas is nontrivial (and interesting, in our

[4] Gupta et al. [10] obtained $\tilde{O}(\sqrt{n})$ query time for two-dimensional disks, but only for *uniformly distributed* point sets; the general problem was left open in their paper.

opinion). Our conditional lower bound proof for three-dimensional orthogonal RCP is similar to some previous work (for example, Davoodi et al.'s conditional lower bound for two-dimensional range diameter [9]), and along the way, we introduce a new variant of colored range searching, *color uniqueness query*, which may be of independent interest.

2 Preliminaries

The first two results we need are the well-known partition lemma and cutting lemma, both of which are extensively used for solving range-search problems.

Lemma 1. (Partition lemma [13]) *Given a set S of n points in $\mathbb{R}^d$ and a parameter $1 \leq r \leq n^{1-\delta}$ for an arbitrarily small constant $\delta > 0$, one can compute in $O(n \log n)$ time a partition $\{S_1, \ldots, S_r\}$ of S and r simplices $\Delta_1, \ldots, \Delta_r$ in $\mathbb{R}^d$ such that* **(1)** *$S_i \subseteq \Delta_i$ for all $i \in \{1, \ldots, r\}$,* **(2)** *$|S_i| = O(n/r)$ for all $i \in \{1, \ldots, r\}$, and* **(3)** *any hyperplane in $\mathbb{R}^d$ crosses $O(r^{1-1/d})$ simplices among $\Delta_1, \ldots, \Delta_r$.*

Lemma 2. (Cutting lemma [8]) *Given a set $\mathcal{H}$ of n hyperplanes in $\mathbb{R}^d$ and a parameter $1 \leq r \leq n$, one can compute in $O(nr^{d-1})$ time a cutting of $\mathbb{R}^d$ into $O(r^d)$ cells each of which is a constant-complexity polytope intersecting $O(n/r)$ hyperplanes in $\mathcal{H}$. In addition, the algorithm for computing the cutting stores the cells into an $O(r^d)$-space data structure which can report in $O(\log r)$ time, for a specified point in $x \in \mathbb{R}^d$, the cell containing x.*

We shall also use the standard range-reporting data structures for orthogonal, simplex, and halfspace queries, stated in the following lemma:

Lemma 3. *Given a set S of n points in $\mathbb{R}^d$, one can build in $O(n \log^{O(1)} n)$ time an $O(n \log^{O(1)} n)$-space data structure which can*

(a) **(Orthogonal range reporting [5])** *report, for a specified orthogonal box B in $\mathbb{R}^d$, the points in $S \cap B$ in $O(\log^{O(1)} n + k)$ time where $k = |S \cap B|$;*
(b) **(Simplex range reporting [13])** *report, for a specified simplex Δ in $\mathbb{R}^d$, the points in $S \cap \Delta$ in $O(n^{1-1/d} \log^{O(1)} n + k)$ time where $k = |S \cap \Delta|$;*
(c) **(Halfspace range reporting [14])** *report, for a specified halfspace H in $\mathbb{R}^d$, the points in $S \cap H$ in $O(n^{1-1/\lfloor d/2 \rfloor} \log^{O(1)} n + k)$ query time where $k = |S \cap H|$.*

Using a multi-level data structure that combines range trees with the above structures, we can obtain range-reporting structures for query ranges that are the intersections of an orthogonal box and a simplex/halfspace (see the full version [6] for a detailed proof).

Lemma 4. *Given a set S of n points in $\mathbb{R}^d$, one can build in $O(n \log^{O(1)} n)$ time an $O(n \log^{O(1)} n)$-space data structure which can*

(a) **(Box-simplex range reporting)** *report, for a specified orthogonal box B and simplex Δ in $\mathbb{R}^d$, the points in $S \cap B \cap \Delta$ in $O(\log^{O(1)} n + m^{1-1/d} \log^{O(1)} n + k)$ time where $m = |S \cap B|$ and $k = |S \cap B \cap \Delta|$;*

(b) **(Box-halfspace range reporting)** *report, for a specified orthogonal box B and halfspace H in $\mathbb{R}^d$, the points in $S \cap B \cap H$ in $O(\log^{O(1)} n + m^{1-1/\lfloor d/2 \rfloor} \log^{O(1)} n + k)$ time where $m = |S \cap B|$ and $k = |S \cap B \cap H|$.*

3 Orthogonal RCP queries

3.1 Data structure

Let S be a set of n points in $\mathbb{R}^d$. In this section, we show how to build a RCP data structure on S for orthogonal queries. First, we build a (standard) d-dimensional range tree $\mathcal{T}$ on S. Each node $\mathbf{u}$ of $\mathcal{T}$ corresponds to a *canonical subset* of S, which we denote by $S(\mathbf{u})$. We say $\mathbf{u}$ is a *heavy* node if $|S(\mathbf{u})| \geq \sqrt{n}$. For every pair $(\mathbf{u}, \mathbf{v})$ of heavy nodes, we compute the closest pair $\phi_{\mathbf{u},\mathbf{v}}$ in $S(\mathbf{u}) \cup S(\mathbf{v})$; denote by Φ the set of all these pairs. Then we build an orthogonal range-reporting data structure $\mathcal{D}(S)$ on S (Lemma 3(a)). Our orthogonal RCP data structure consists of the range tree $\mathcal{T}$, the data structure $\mathcal{D}(S)$, and the pair set Φ.

Query procedure. Consider a query box B in $\mathbb{R}^d$. Our goal is to find the closest pair in $S \cap B$ using the data structure described above. By searching in the range tree $\mathcal{T}$, we can find $t = O(\log^{O(1)} n)$ canonical nodes $\mathbf{c}_1, \ldots, \mathbf{c}_t$ corresponding to B. We have $S \cap B = \bigcup_{i=1}^{t} S(\mathbf{c}_i)$. Let $I = \{i : \mathbf{c}_i \text{ is a heavy node}\}$ and $I' = \{1, \ldots, t\} \backslash I$. (See Figure 1(left).) For all $i, j \in I$, we obtain the pair $\phi_{\mathbf{c}_i,\mathbf{c}_j}$ from Φ and take the closest one $\phi \in \{\phi_{\mathbf{c}_i,\mathbf{c}_j} : i, j \in I\}$. On the other hand, we compute $L = \bigcup_{i \in I'} S(\mathbf{c}_i)$. We take the closest pair ϕ' in L. Let $\delta = \min\{|\phi|, |\phi'|\}$. For each $a \in L$, let $\square_a$ be the hypercube centered at a with side-length 2δ. We query, for each $a \in L$, the box range-reporting data structure $\mathcal{D}(S)$ with $\square_a \cap B$ to obtain the set $P_a = S \cap \square_a \cap B$. After this, for each $a \in L$, we compute a pair ψ_a consisting of a and the nearest neighbor of a in $P_a \backslash \{a\}$. We then take the closest one $\psi \in \{\psi_a : a \in L\}$. Finally, if $|\psi| < |\phi|$, then we return ψ as the answer; otherwise, we return ϕ as the answer.

We now verify the correctness of the above query procedure. Let $\phi^* = (a, b)$ be the closest pair in $S \cap B$. It suffices to show that $|\phi| \leq |\phi^*|$ or $|\psi| \leq |\phi^*|$. Suppose $a \in S(\mathbf{c}_i)$ and $b \in S(\mathbf{c}_j)$. If $i, j \in I$, then $|\phi| \leq |\phi_{\mathbf{c}_i,\mathbf{c}_j}| \leq |\phi^*|$ and we are done. Otherwise, either $i \in I'$ or $j \in I'$; assume $i \in I'$ without loss of generality. It follows that $a \in L$. Since ϕ^* is the closest pair in $S \cap B$, we have $|\phi^*| \leq |\phi|$ and $|\phi^*| \leq |\phi'|$, which implies that the distance between a and b is at most δ. Therefore, $b \in P_a$. Now we have $|\psi| \leq |\psi_a| \leq |\phi^*|$, which completes the proof of the correctness.

Analysis. We analyze the performance (space, query time, and preprocessing time) of our orthogonal RCP data structure. To this end, we first bound the number of the heavy nodes. The lemma below follows immediately from the well-known fact that the sum of sizes of the canonical subsets in a range tree is $O(n \log^d n)$.

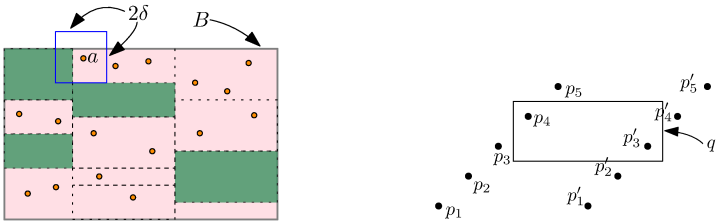


Fig. 1. (Left) The canonical nodes in the range tree $\mathcal{T}$ break the query box B into thirteen disjoint regions. The green regions correspond to set I (the heavy nodes). The orange points form the set L. For one of the points in L (denoted by a), the box $\square_a$ is shown in blue. The crucial property is that the number of points which lie in $B \cap \square_a$ is $O(1)$. (Right) Reduction from the set intersection query to the color uniqueness query. The set intersection query is to test if S_4 and S_3 are disjoint, and the query rectangle q for the color uniqueness query exactly contains points p_4 and p'_3.

Lemma 5. *There are $O(\sqrt{n}\log^{O(1)} n)$ heavy nodes in $\mathcal{T}$.*

By the above lemma, the space of the data structure is $O(n\log^{O(1)} n)$. Indeed, the range tree $\mathcal{T}$ and the data structure $\mathcal{D}(S)$ both occupy $O(n\log^{d-1} n)$ space, and the pair-set Φ takes $O(n\log^{2d-2} n)$ space as there are $O(\sqrt{n}\log^{O(1)} n)$ heavy nodes. The preprocessing time is $O(n\sqrt{n}\log^{O(1)} n)$. Indeed, building the range tree $\mathcal{T}$ and the data structure $\mathcal{D}(S)$ takes $O(n\log^{O(1)} n)$ time. We claim that the pair-set Φ can be computed in $O(n\sqrt{n}\log^{O(1)} n)$ time. We first find the set $\mathcal{H}$ of heavy nodes, which can be done in $O(n\log^{O(1)} n)$ time by simply checking every node of $\mathcal{T}$. For two pairs $(\mathbf{u},\mathbf{v})$ and $(\mathbf{u}',\mathbf{v}')$ of nodes in $\mathcal{H}$, we write $(\mathbf{u},\mathbf{v}) \preceq (\mathbf{u}',\mathbf{v}')$ if $|S(\mathbf{u})|+|S(\mathbf{v})| \leq |S(\mathbf{u}')|+|S(\mathbf{v}')|$. Then "$\preceq$" is a partial order on $\mathcal{H}\times\mathcal{H}$. We consider the pairs of heavy nodes in this partial order from the smallest to the greatest. For each pair $(\mathbf{u},\mathbf{v})$, we compute $\phi_{\mathbf{u},\mathbf{v}}$ as follows. If $|S(\mathbf{u})| < 2\sqrt{n}$ and $|S(\mathbf{v})| < 2\sqrt{n}$, we explicitly compute $S(\mathbf{u}) \cup S(\mathbf{v})$ and then compute $\phi_{\mathbf{u},\mathbf{v}}$ using the standard closest-pair algorithm in $O(\sqrt{n}\log n)$ time. Otherwise, either $|S(\mathbf{u})| \geq 2\sqrt{n}$ or $|S(\mathbf{v})| \geq 2\sqrt{n}$. Without loss of generality, assume $|S(\mathbf{u})| \geq 2\sqrt{n}$. Then the two children $\mathbf{u}_1$ and $\mathbf{u}_2$ of $\mathbf{u}$) are both heavy. Note that $\phi_{\mathbf{u},\mathbf{v}}$ is the closest one among $\phi_{\mathbf{u}_1,\mathbf{v}}, \phi_{\mathbf{u}_2,\mathbf{v}}, \phi_{\mathbf{u}_1,\mathbf{u}_2}$ by construction. Also note that $(\mathbf{u}_1,\mathbf{v}) \preceq (\mathbf{u},\mathbf{v})$, $(\mathbf{u}_2,\mathbf{v}) \preceq (\mathbf{u},\mathbf{v})$, $(\mathbf{u}_1,\mathbf{u}_2) \preceq (\mathbf{u},\mathbf{v})$, thus $\phi_{\mathbf{u}_1,\mathbf{v}}, \phi_{\mathbf{u}_2,\mathbf{v}}, \phi_{\mathbf{u}_1,\mathbf{u}_2}$ have already been computed when considering $(\mathbf{u},\mathbf{v})$. With $\phi_{\mathbf{u}_1,\mathbf{v}}, \phi_{\mathbf{u}_2,\mathbf{v}}, \phi_{\mathbf{u}_1,\mathbf{u}_2}$ in hand, we can compute $\phi_{\mathbf{u},\mathbf{v}}$ in $O(1)$ time. In sum, $\phi_{\mathbf{u},\mathbf{v}}$ can be computed in $O(\sqrt{n}\log n)$ time in any case. Since $|\mathcal{H}\times\mathcal{H}| = O(n\log^{O(1)} n)$, we can compute Φ in $O(n\sqrt{n}\log^{O(1)} n)$ time. This completes the discussion of the preprocessing time. Next, we analyze the query time. Finding the canonical nodes $\mathbf{c}_1,\ldots,\mathbf{c}_t$ takes $O(\log^{O(1)} n)$ time, so does computing the index sets I and I'. Obtaining the set $\{\phi_{\mathbf{c}_i,\mathbf{c}_j} : i,j \in I\}$ and computing ϕ takes $O(\log^{O(1)} n)$ time since $|I| \leq t$ and $t = O(\log^{O(1)} n)$. Computing ϕ' requires $O(\sqrt{n}\log^{O(1)} n)$ time, because $|L| = O(t\sqrt{n}) = O(\sqrt{n}\log^{O(1)} n)$. For a point $a \in L$, reporting the points in P_a takes $O(\log^{O(1)} n + |P_a|)$ time. Therefore, computing all the P_a's can be done

in $O(|L| \log^{O(1)} n + \sum_{a \in L} |P_a|)$ time. To bound this quantity, we observe the following fact.

Lemma 6. *$|P_a| = O(1)$ for all $a \in L$.*

Proof. We have $S \cap B = (\bigcup_{i \in I} S(\mathbf{u}_i)) \cup L$. It suffices to show that $|(\bigcup_{i \in I} S(\mathbf{u}_i)) \cap \square_a| = O(1)$ and $|L \cap \square| = O(1)$. Both facts follow from the pigeonhole principle readily. Indeed, we have $|(\bigcup_{i \in I} S(\mathbf{u}_i)) \cap \square_a| = O(1)$ because ϕ is the closest pair in $\bigcup_{i \in I} S(\mathbf{u}_i)$ and $|\phi| \geq \delta$. We have $|L \cap \square| = O(1)$ because ϕ' is the closest pair in L and $|\phi'| \geq \delta$. This completes the proof. □

By the above lemma and the fact $|L| = O(\sqrt{n} \log^{O(1)} n)$, we can compute all the P_a's in $O(\sqrt{n} \log^{O(1)} n)$ time. The pair ψ can be directly obtained after knowing all the P_a's, hence the total query time is $O(\sqrt{n} \log^{O(1)} n)$. We conclude the following.

Theorem 1. *Given a set S of n points in $\mathbb{R}^d$, one can construct in $\tilde{O}(n\sqrt{n})$ time an orthogonal RCP data structure on S with $\tilde{O}(n)$ space and $\tilde{O}(\sqrt{n})$ query time.*

3.2 Conditional hardness

In this subsection, we prove a conditional lower-bound for the orthogonal RCP query, which shows that the upper bound given in Theorem 1 is tight, ignoring $\log n$ factors. First, we define the following problem [15].

Problem 1. **(Set intersection query)** The input is a collection of sets $S_1, S_2, \ldots, S_m$ of positive reals such that $\sum_{i=1}^{m} |S_i| = n$. Given query indices i and j, report if S_i and S_j are disjoint, or not?

This problem can be viewed as a query version of Boolean matrix multiplication, and is conjectured to be *hard*: in the cell-probe model without the floor function and where the cardinality of each set S_i is upper-bounded by $\log^{O(1)} m$, any data structure to answer the set intersection problem in $\tilde{O}(\alpha)$ time requires $\tilde{\Omega}((n/\alpha)^2)$ space, for $1 \leq \alpha \leq n$ [9, 15]. In particular, any linear-space structure is believed to require $\tilde{\Omega}(\sqrt{n})$ time.

Next we introduce an intermediate geometric problem, which may be of independent interest:

Problem 2. **(Color uniqueness query)** The input is a set S of n *colored* points in $\mathbb{R}^2$. Specifically, let C be a collection of distinct colors, and each point $p \in S$ is associated with some color from C. Given a query rectangle q, report if all the colors are unique in $S \cap q$? In other words, is there a color which has at least two points in $S \cap q$?

We will perform a two-step reduction: first, reduce the set intersection query to the color uniqueness query, and then reduce the two-dimensional color uniqueness query to the three-dimensional orthogonal RCP query.

Reduction from set intersection to color uniqueness in $\mathbb{R}^2$. Given an instance of the set intersection query, we will construct an instance of the color uniqueness query. Let $p_1 = (1, 1), p_2 = (2, 2), \ldots, p_m = (m, m)$, and $p'_1 = (m + 1, 1), p'_2 = (m + 2, 2), \ldots, p'_m = (2m, m)$. Next, assign a unique color to each distinct element in $S_1 \cup S_2 \cup \ldots \cup S_m$. Now replace each point p_i with $|S_i|$ *new* points such that (a) the new points are within a distance of $\varepsilon \ll 1$ from p_i, and (b) each new point picks a distinct color from the colors assigned to the elements in S_i. Perform a similar operation for points p'_i. Let P be the collection of these $2n$ new points.

To answer if S_i and S_j are disjoint ($j < i$), we ask a color uniqueness query on P with an axis-aligned rectangle $q = [i - \varepsilon, m + j + \varepsilon] \times [j - \varepsilon, i + \varepsilon]$ (see Figure 1(right)). If there is a color which contains two points, then we report that S_i and S_j are not disjoint; otherwise, we report that S_i and S_j are disjoint. The correctness is easy to see: the key observation is that q exactly contains the points of S_i and S_j. Therefore, S_i and S_j are disjoint iff all the colors are unique in $P \cap q$. Reductions of this flavor have been performed before [3, 9, 12, 16].

Reduction from color uniqueness in $\mathbb{R}^2$ to orthogonal RCP in $\mathbb{R}^3$. Given an instance of the color uniqueness query, we will now construct an instance of the orthogonal RCP query in $\mathbb{R}^3$. Let $d_{\max}$ be the maximum Euclidean distance between any two points in S, and let $c_1, c_2, \ldots, c_{|C|}$ be the $|C|$ colors in the dataset. Then each point $p = (p_x, p_y) \in S$ with color c_i is mapped to a 3-d point $p' = (p_x, p_y, 2 \cdot i \cdot d_{\max})$. Let P be the collection of these n newly mapped points.

To answer the color uniqueness query for a rectangle q, we will ask an orthogonal RCP query on P with the query box $q \times (-\infty, \infty)$. If the closest-pair distance is less than or equal to $d_{\max}$, then we report that there is a color which contains at least two points inside q; otherwise, we report that all the colors are unique inside q. Once again, the correctness is easy to see: the key observation is that the distance between points of different colors in P is at least $2 \cdot d_{\max}$.

The above two reductions together implies our conditional lower bound, which is presented in the following theorem.

Theorem 2. *The orthogonal RCP query is at least as hard as the set intersection query.*

4 Simplex RCP queries

Let S be a set of n points in $\mathbb{R}^d$, and r be a parameter to be specified shortly. In this section, we show how to build a RCP data structure on S for simplex queries. First, we use Lemma 1 to compute a partition $\{S_1, \ldots, S_r\}$ of S and r simplices $\Delta_1, \ldots, \Delta_r$ in $\mathbb{R}^d$ satisfying the conditions in the lemma. For every $i, j \in \{1, \ldots, r\}$, we compute the closest pair $\phi_{i,j}$ in $S_i \cup S_j$; denote by Φ the set of all these pairs. Then we build a box-simplex range-reporting data structure $\mathcal{D}'(S)$ on S (Lemma 4(a)). Our simplex RCP data structure consists of the partition $\{S_1, \ldots, S_r\}$, the simplices $\Delta_1, \ldots, \Delta_r$, the data structure $\mathcal{D}'(S)$, and the pair set Φ.

Query procedure. Consider a query simplex Δ in $\mathbb{R}^d$. Our goal is to find the closest pair in $S \cap \Delta$ using the data structure described above. We first compute two index sets $I = \{i : \Delta_i \subseteq \Delta\}$, $I' = \{i : \Delta_i \not\subseteq \Delta \text{ and } \Delta_i \cap \Delta \neq \emptyset\}$. (See Figure 2.) These index sets are computed by explicitly considering the r simplices $\Delta_1, \ldots, \Delta_r$. For all $i, j \in I$, we obtain the pair $\phi_{i,j}$ from Φ and take the closest one $\phi \in \{\phi_{i,j} : i, j \in I\}$. On the other hand, we compute a set $L = (\bigcup_{i \in I'} S_i) \cap \Delta$ by simply checking, for every $i \in I'$ and every $a \in S_i$, whether $a \in \Delta$. We take the closest pair ϕ' in L. Let $\delta = \min\{|\phi|, |\phi'|\}$. For each $a \in L$, let $\square_a$ be the hypercube centered at a with side length 2δ. We query, for each $a \in L$, the box-simplex range-reporting data structure $\mathcal{D}'(S)$ with $\square_a$ and Δ to obtain the set $P_a = S \cap \square_a \cap \Delta$. After this, for each $a \in L$, we compute a pair ψ_a consisting of a and the nearest neighbor of a in $P_a \backslash \{a\}$. We then take the closest one $\psi \in \{\psi_a : a \in L\}$. Finally, if $|\psi| < |\phi|$, then we return ψ as the answer; otherwise, we return ϕ as the answer.

We now verify the correctness of the above query procedure. Let $\phi^* = (a, b)$ be the closest pair in $S \cap \Delta$. It suffices to show that $|\phi| \leq |\phi^*|$ or $|\psi| \leq |\phi^*|$. Suppose $a \in S_i$ and $b \in S_j$. We first notice that $i, j \in I \cup I'$. Indeed, if $i \notin I \cup I'$ (resp., $j \notin I \cup I'$), then $\Delta_i \cap \Delta = \emptyset$ (resp., $\Delta_j \cap \Delta = \emptyset$) and hence $S_i \cap \Delta = \emptyset$ (resp., $S_j \cap \Delta = \emptyset$), which contradicts the fact that $a \in S_i \cap \Delta$ (resp., $b \in S_i \cap \Delta$). If $i, j \in I$, then $|\phi| \leq |\phi_{i,j}| \leq |\phi^*|$ and we are done. Otherwise, either $i \in I'$ or $j \in I'$; assume $i \in I'$ without loss of generality. It follows that $a \in L$. Since ϕ^* is the closest pair in $S \cap \Delta$, we have $|\phi^*| \leq |\phi|$ and $|\phi^*| \leq |\phi'|$, which implies that the distance between a and b is at most δ. Therefore, $b \in P_a$. Now we have $|\psi| \leq |\psi_a| \leq |\phi^*|$, which completes the proof of the correctness.

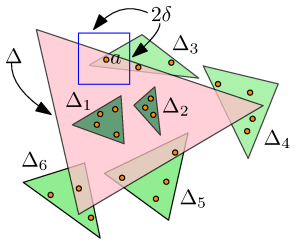


Fig. 2. $I = \{\Delta_1, \Delta_2\}$ and $I' = \{\Delta_3, \Delta_4, \Delta_5, \Delta_6\}$.

Analysis. We analyze the performance (space, query time, and preprocessing time) of our simplex RCP data structure. The space of the data structure is $O(n \log^{O(1)} n + r^2)$, because $\mathcal{D}'(S)$ occupies $O(n \log^{O(1)} n)$ space and Φ occupies $O(r^2)$ space. The preprocessing time is $O(nr \log^{O(1)} n)$. Indeed, computing the partition $\{S_1, \ldots, S_r\}$ and the simplices $\Delta_1, \ldots, \Delta_r$ takes $O(n \log n)$ time by Lemma 1. Computing $\phi_{i,j}$ for some fixed $i, j \in \{1, \ldots, r\}$ can be done in $O((n/r) \log(n/r))$ time using the standard closest-pair algorithm, because $|S_i \cup S_j| = O(n/r)$. It follows that computing Φ takes $O(nr \log n)$ time. Fi-

nally, building the data structure $\mathcal{D}'(S)$ requires $O(n \log^{O(1)} n)$ time. As such, our simplex RCP data structure can be constructed in $O(nr \log^{O(1)} n)$ time. Next, we analyze the query time. The index sets I and I' are computed in $O(r)$ time. Obtaining the set $\{\phi_{i,j} : i, j \in I\}$ and computing ϕ requires $O(r^2)$ time. The set L is computed by explicitly considering all the points in $\bigcup_{i \in I'} S_i$ in $O(\sum_{i \in I'} |S_i|)$ time. We notice that $|I'| = O(r^{1-1/d})$, since each facet of Δ only intersects $O(r^{1-1/d})$ simplices among $\Delta_1, \ldots, \Delta_r$ by Lemma 1. It follows that $\sum_{i \in I'} |S_i| = O(n/r^{1/d})$, because $|S_i| = O(n/r)$. That says, L can be computed in $O(n/r^{1/d})$ time and in particular, $|L| = O(n/r^{1/d})$. Once L is obtained, ϕ' can be computed in $O((n/r^{1/d}) \log(n/r^{1/d}))$ time using the standard closest-pair algorithm. For a point $a \in L$, reporting the points in P_a takes $O(\log^{O(1)} n + m_a^{1-1/d} \log^{O(1)} m_a + |P_a|)$ time where $m_a = |S \cap \square_a|$, by Lemma 4(a). Therefore, computing all the P_a's can be done in $O(\sum_{a \in L} m_a^{1-1/d} \log^{O(1)} n + \sum_{a \in L} |P_a|)$ time. To bound this quantity, we observe the following fact.

Lemma 7. *$\sum_{a \in L} m_a = O(n)$ and $|P_a| = O(1)$ for all $a \in L$.*

Proof. We first prove $\sum_{a \in L} m_a = O(n)$. Consider a point $p \in S$. Let $\square_p$ be the hypercube centered at p with side-length 2δ. Note that $p \in P_a$ only if $a \in \square_p$ for all $a \in L$. Since ϕ' is the closest pair in L and $|\phi'| \geq \delta$, we have $L \cap \square_p = O(1)$ by the pigeonhole principle. Therefore, only a constant number of points in L is contained in p. In other words, any point $p \in S$ is contained in P_a for only a constant number of $a \in L$, which implies $\sum_{a \in L} m_a = O(n)$. Next, we prove that $|P_a| = O(1)$ for all $a \in L$. Clearly, $S \cap \Delta = (\bigcup_{i \in I} S_i) \cup L$. So it suffices to show that $|(\bigcup_{i \in I} S_i) \cap \square_a| = O(1)$ and $|L \cap \square_a| = O(1)$. Both facts follow from the pigeonhole principle readily. Indeed, we have $|(\bigcup_{i \in I} S_i) \cap \square_a| = O(1)$ because ϕ is the closest pair in $\bigcup_{i \in I} S_i$ and $|\phi| \geq \delta$. We have $|L \cap \square_a| = O(1)$ because ϕ' is the closest pair in L and $|\phi'| \geq \delta$. This completes the proof of $|P_a| = O(1)$. $\square$

By the above lemma and Hölder's inequality, we have

$$\sum_{a \in L} m_a^{1-1/d} \leq O(n^{1-1/d} |L|^{1/d}) = O\left(\frac{n}{r^{1/d^2}}\right),$$

which implies that computing all the P_a's takes $O((n \log^{O(1)} n)/r^{1/d^2})$ time. The pair ψ can be directly obtained after knowing all the P_a's. Hence, the total query time is $O(r^2 + (n \log^{O(1)} n)/r^{1/d^2})$. Setting $r = n^{d^2/(2d^2+1)}$ gives:

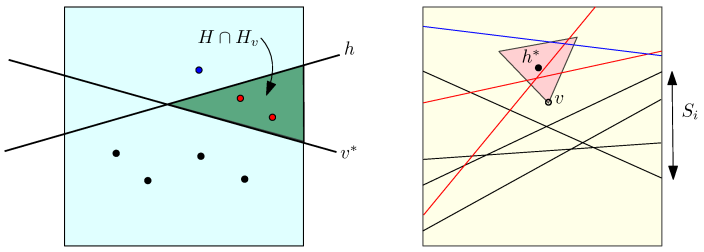
Theorem 3. *Given a set S of n points in $\mathbb{R}^d$, one can construct in $\tilde{O}(n^{(3d^2+1)/(2d^2+1)})$ time a simplex RCP data structure on S with $\tilde{O}(n)$ space and $\tilde{O}(n^{1-1/(2d^2)})$ query time.*

Note that our data structure above can also handle constant-complexity polytope RCP queries (with the same query procedure and query time). In other words, the data structure can be used to report, for specified $O(1)$ halfspaces $H_1, \ldots, H_c$ in $\mathbb{R}^d$, the closest pair in $S \cap (\bigcap_{i=1}^{c} H_i)$ in $\tilde{O}(n^{1-1/(2d^2)})$ time.

5 Halfspace RCP queries

Let S be a set of n points in $\mathbb{R}^d$, and r be a parameter to be specified shortly. In this section, we show how to build an RCP data structure on S for halfspace queries. The same method can also result in an RCP data structure for ball queries, using the standard lifting argument. Since halfspace query is a special case of simplex query, the simplex RCP data structure in the last section can be directly used to answer halfspace RCP queries. But in fact, for halfspace RCP queries, we can achieve better bounds.

It suffices to consider the halfspaces which are regions below non-vertical hyperplanes, namely, halfspaces of the form $x_d \leq a_1 x_1 + \cdots + a_{d-1} x_{d-1}$. By duality, a point $a \in S$ maps to a hyperplane a^* in the dual space (which is also a copy of $\mathbb{R}^d$). Also, a non-vertical hyperplane h in the primal $\mathbb{R}^d$ maps to a point h^* in the dual space. The property of duality guarantees that a is above (resp., below) h iff h^* is above (resp., below) a^* for all $a \in S$ and all hyperplanes h (see Figure 3). Define $\mathcal{H} = \{a^* : a \in S\}$. We use Lemma 2 to cut $\mathbb{R}^d$ (the dual space) into $R = O(r^d)$ cells $\Xi_1, \ldots, \Xi_R$ each of which is a constant-complexity polytope intersecting $O(n/r)$ hyperplanes in $\mathcal{H}$. For $i \in \{1, \ldots, R\}$, let $S_i = \{a : a^* \text{ is below } \Xi_i\}$. We associate to the cell Ξ_i the closest pair ϕ_i in S_i. Furthermore, we build a simplex range-reporting data structure $\mathcal{D}(S)$ on S (Lemma 3(b)) and a box-halfspace range-reporting data structure $\mathcal{D}'(S)$ in S (Lemma 4(b)). Our halfspace RCP data structure consists of the cells $\Xi_1, \ldots, \Xi_R$ (with the associated pairs $\phi_1, \ldots, \phi_r$) and the data structures $\mathcal{D}(S)$ and $\mathcal{D}'(S)$. The cells $\Xi_1, \ldots, \Xi_R$ are stored in the way mentioned in Lemma 2 (so that we can do point location efficiently).



(a) Primal plane (b) Dual plane

Fig. 3. The dataset shown in (a) consists of seven points. The dual h^* of the query hyperplane h lies inside the cell Ξ_i shown in pink in (b). The closest pair among the black points, ϕ_i, is computed in the preprocessing phase itself (since the dual of the black points is the set S_i). The red points belong to set L and are explicitly reported during the query procedure.

Query procedure. Consider a query halfspace H that is the region below a non-vertical hyperplane h. Our goal is to find the closest pair in $S \cap H$ using the data structure described above. To this end, we first find the cell Ξ_i such that

$h^* \in \Xi_i$. Let V be the set of the vertices of Ξ_i. We have $V = O(1)$ by Lemma 2. For every $v \in V$, let H_v be the halfspace above the non-vertical hyperplane v^* in the primal $\mathbb{R}^d$. Using $\mathcal{D}(S)$, we find the points in $S \cap (H \cap H_v)$ for all $v \in V$ and obtain the set $L = \bigcup_{v \in V} S \cap (H \cap H_v)$. We take the closest pair ϕ' in L. Let $\delta = \min\{|\phi_i|, |\phi'|\}$ (recall that ϕ_i is the pair associated to Ξ_i). For each $a \in L$, let $\square_a$ be the hypercube centered at a with side-length 2δ. We query, for each $a \in L$, the box-halfspace range-reporting data structure $\mathcal{D}'(S)$ with $\square_a$ and H to obtain the set $P_a = S \cap \square_a \cap H$. After this, for each $a \in L$, we compute a pair ψ_a consisting of a and the nearest neighbor of a in $P_a \backslash \{a\}$. We then take the closest one $\psi \in \{\psi_a : a \in L\}$. Finally, if $|\psi| < |\phi_i|$, then we return ψ as the answer; otherwise, we return ϕ_i as the answer.

We now verify the correctness of the above query procedure. First of all, we claim that $S \cap H = S_i \cup L$. Indeed, we have $L \subseteq S \cap H$ by definition and $S_i \subseteq S \cap H$ because a^* is below Ξ_i (and hence below h^*) for all $a \in S_i$; this implies $S_i \cup L \subseteq S \cap H$. To see $S \cap H \subseteq S_i \cup L$, let $a \in S \cap H$ be a point. If a^* is below Ξ_i, then $a \in S_i$. Otherwise, there exists $v \in V$ such that a^* is above v. It follows that $a \in S \cap (H \cap H_v) \subseteq L$. Therefore, $S \cap H \subseteq S_i \cup L$ and $S \cap H = S_i \cup L$. With this observation in hand, we first show that the returned answer is a pair in $S \cap H$. It suffices to show that both ϕ_i and ψ are pairs in $S \cap H$. The two points of ϕ_i are both in S_i and hence in $S \cap H$. To see ψ is a pair in $S \cap H$, suppose $\psi = \psi_a$ for $a \in L$. By definition, ψ_a consists of a and the nearest neighbor of a in $P_a \backslash \{a\}$. We have $a \in L \subseteq S \cap H$ and $P_a \subseteq L \subseteq S \cap H$, hence ψ is a pair in $S \cap H$. Next, we show that the returned answer is the closest pair in $S \cap H$. Let $\phi^* = (a, b)$ be the closest-pair in $S \cap H$. It suffices to show that $|\phi_i| \leq |\phi^*|$ or $|\psi| \leq |\phi^*|$. If $a, b \in S_i$, then $|\phi_i| \leq |\phi^*|$ and we are done. Otherwise, assume $a \notin S_i$ and thus $a \in L$, without loss of generality. Since ϕ^* is the closest pair in $S \cap H$, we have $|\phi^*| \leq |\phi_i|$, which implies that the distance between a and b is at most δ. Therefore, $b \in P_a$. Now we have $|\psi| \leq |\psi_a| \leq |\phi^*|$, which completes the proof of the correctness.

Analysis. We analyze the performance (space, query time, and preprocessing time) of our halfspace RCP data structure. The space of the data structure is $O(n \log^{O(1)} n + R)$, because $\mathcal{D}(S)$ occupies $O(n)$ space, $\mathcal{D}'(S)$ occupies $O(n \log^{O(1)} n)$ space, and storing $\Xi_1, \ldots, \Xi_R$ (with the associated pairs $\phi_1, \ldots, \phi_R$) requires $O(R)$ space. Next, we analyze the query time. Determining the cell Ξ_i takes $O(\log r)$ time by Lemma 2. For each $v \in V$, reporting the points in $S \cap (H \cap H_v)$ takes $O(n^{1-1/d} \log^{O(1)} n + k_v)$ time where $k_v = |S \cap (H \cap H_v)|$. We claim that a^* intersects Ξ_i for any $a \in S \cap (H \cap H_v)$. Indeed, a^* is below h because $a \in H$ and is above v because $a \in H_v$. Thus, a^* intersects the segment connecting h^* and v. Since $h^*, v \in \Xi_i$, a^* intersects Ξ_i. It follows that $k_v = O(n/r)$ by Lemma 2. Furthermore, because $V = O(1)$, L can be computed in $O(n^{1-1/d} \log^{O(1)} n + \sum_{v \in V} k_v) = O(n^{1-1/d} \log^{O(1)} n + n/r)$ time and $|L| = O(\sum_{v \in V} k_v) = O(n/r)$. Once L is obtained, ϕ' can be computed in $O((n/r) \log(n/r))$ time using the standard closest-pair algorithm. For a point $a \in L$, reporting the points in P_a takes $O(\log^{O(1)} n + m_a^{1-1/\lfloor d/2 \rfloor} \log^{O(1)} m_a + |P_a|)$

time where $m_a = |S \cap \square_a|$, by Lemma 4(b). By exactly the same argument in the proof of Lemma 7, we have the following observation:

Lemma 8. $\sum_{a \in L} m_a = O(n)$ *and* $|P_a| = O(1)$ *for all* $a \in L$.

By the above lemma and Hölder's inequality, we have

$$\sum_{a \in L} m_a^{1-1/\lfloor d/2 \rfloor} \leq O(n^{1-1/\lfloor d/2 \rfloor} |L|^{1/\lfloor d/2 \rfloor}) = O\left(\frac{n}{r^{1/\lfloor d/2 \rfloor}}\right),$$

which implies that computing all the P_a's takes $O(n \log^{O(1)} n / r^{1/\lfloor d/2 \rfloor})$ time. The pair ψ can be directly obtained after knowing all the P_a's. Hence, the total query time is $O(\log r + n \log^{O(1)} n / r^{1/\lfloor d/2 \rfloor})$. Finally, we analyze the preprocessing time. The data structures $\mathcal{D}(S)$ and $\mathcal{D}'(S)$ can both be constructed in $O(n \log^{O(1)} n)$ time by Lemma 3(b) and 4(b). The cells $\Xi_1, \ldots, \Xi_R$ can be computed in $O(nr^{d-1})$ time by Lemma 2. So it suffices to show how to compute the pairs $\phi_1, \ldots, \phi_R$ efficiently. To this end, we build a simplex RCP data structure on S as described in Theorem 3, which takes $\tilde{O}(n^{(3d^2+1)/(2d^2+1)})$ time. Fix $i \in \{1, \ldots, R\}$ and let V be the set of the $O(1)$ vertices of Ξ_i. For $v \in V$, let H'_v be the halfspace below the hyperplane v^* in the primal space. We claim that $S_i = S \cap (\bigcap_{v \in V} H'_v)$. To see this, consider a point $a \in S$. We have $a \in S_i$ iff a^* is below Ξ_i iff v is below a^* for all $v \in V$, or equivalently, $a \in H'_v$ for all $v \in V$. Thus, $S_i = S \cap (\bigcap_{v \in V} H'_v)$. We can then compute the closest pair ϕ_i in S_i using the simplex RCP data structure with the query range $\bigcap_{v \in V} H'_v$ (as mentioned at the end of Section 4, our simplex RCP data structure can handle queries which are intersections of constant number of halfspaces). Computing ϕ_i takes $O(n^{1-1/(2d^2)} \log^{O(1)} n)$ time, and hence computing all pairs $\phi_1, \ldots, \phi_R$ takes $O(Rn^{1-1/(2d^2)} \log^{O(1)} n)$ time. In sum, the preprocessing time of our halfspace RCP data structure is $O((nr^{d-1} + n^{(3d^2+1)/(2d^2+1)} + Rn^{1-1/(2d^2)}) \log^{O(1)} n)$. Setting $r = n^{1/d}$ gives:

Theorem 4. *Given a set* S *of* n *points in* $\mathbb{R}^d$*, one can construct in* $\tilde{O}(n^{2-1/(2d^2)})$ *time a halfspace RCP data structure on* S *with* $\tilde{O}(n)$ *space and* $\tilde{O}(n^{1-1/(d\lfloor d/2 \rfloor)})$ *query time.*

References

1. Mohammad Ali Abam, Paz Carmi, Mohammad Farshi, and Michiel Smid. On the power of the semi-separated pair decomposition. *Computational Geometry*, 46(6):631–639, 2013.
2. Pankaj K Agarwal and Jeff Erickson. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.
3. Pankaj K. Agarwal, Nirman Kumar, Stavros Sintos, and Subhash Suri. Range-max queries on uncertain data. *Journal of Computer and System Sciences*, 94:118–134, 2018.
4. Sang Won Bae and Michiel Smid. Closest-pair queries in fat rectangles. *CoRR*, arXiv:1809.10531, 2018.

5. Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2008.
6. Timothy Chan, Saladi Rahul, and Jie Xue. Range closest-pair search higher dimensions. *CoRR*, arXiv:1905.01029, 2019.
7. Timothy M. Chan, Stephane Durocher, Kasper Green Larsen, Jason Morrison, and Bryan T. Wilkinson. Linear-space data structures for range mode query in arrays. *Theory of Computing Systems*, 55(4):719–741, 2014.
8. Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9(2):145–158, 1993.
9. Pooya Davoodi, Michiel H. M. Smid, and Freek van Walderveen. Two-dimensional range diameter queries. In *Latin American Symposium on Theoretical Informatics (LATIN)*, pages 219–230, 2012.
10. P. Gupta, R. Janardan, Y. Kumar, and M. Smid. Data structures for range-aggregate extent queries. *Computational Geometry*, 2(47):329–347, 2014.
11. Prosenjit Gupta. Range-aggregate query problems involving geometric aggregation operations. *Nordic Journal of Computing*, 13(4):294–308, 2006.
12. Haim Kaplan, Natan Rubin, Micha Sharir, and Elad Verbin. Counting colors in boxes. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 785–794, 2007.
13. Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(3):315–334, 1992.
14. Jiří Matoušek. Reporting points in halfspaces. *Computational Geometry*, 2(3):169–186, 1992.
15. Mihai Pătraşcu and Liam Roditty. Distance oracles beyond the Thorup–Zwick bound. *SIAM Journal of Computing*, 43(1):300–311, 2014.
16. Saladi Rahul and Ravi Janardan. Algorithms for range-skyline queries. In *Proceedings of ACM Symposium on Advances in Geographic Information Systems (GIS)*, pages 526–529, 2012.
17. Jing Shan, Donghui Zhang, and Betty Salzberg. On spatial-range closest-pair query. In *Proceedings of Symposium on Advances in Spatial and Temporal Databases (SSTD)*, pages 252–269. Springer, 2003.
18. R. Sharathkumar and Prosenjit Gupta. Range-aggregate proximity queries. Technical Report TR/2007/80, IIIT Hyderabad, Telangana, 2007.
19. Michiel Smid. Closest point problems in computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 877–935. Elsevier Science, Amsterdam, 1999.
20. Jie Xue. Colored range closest-pair problem under general distance functions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 373–390, 2019.
21. Jie Xue, Yuan Li, and Ravi Janardan. Approximate range closest-pair search. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, pages 282–287, 2018.
22. Jie Xue, Yuan Li, Saladi Rahul, and Ravi Janardan. New bounds for range closest-pair problems. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 73:1–73:14. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2018.
23. Jie Xue, Yuan Li, Saladi Rahul, and Ravi Janardan. Searching for the closest-pair in a query translate. *CoRR*, arXiv:1807.09498, 2018.