New Bounds for Range Closest-Pair Problems

Jie Xue
Department of Computer Science & Engineering, University of Minnesota
Minneapolis, MN, USA
http://cs.umn.edu/~xuexx193
xuexx193@umn.edu

Yuan Li
Facebook Inc.
Seattle, WA, USA
lydxlx@fb.com

Saladi Rahul
Department of Computer Science, University of Illinois
Urbana, IL, USA
http://cs.umn.edu/~rahuls
saladi.rahul@gmail.com

Ravi Janardan
Department of Computer Science & Engineering, University of Minnesota
Minneapolis, MN, USA
http://cs.umn.edu/~janardan
janardan@umn.edu

Abstract

Given a dataset S of points in $\mathbb{R}^2$, the range closest-pair (RCP) problem aims to preprocess S into a data structure such that when a query range X is specified, the closest-pair in $S \cap X$ can be reported efficiently. The RCP problem can be viewed as a range-search version of the classical closest-pair problem, and finds applications in many areas. Due to its non-decomposability, the RCP problem is much more challenging than many traditional range-search problems. This paper revisits the RCP problem, and proposes new data structures for various query types including quadrants, strips, rectangles, and halfplanes. Both worst-case and average-case analyses (in the sense that the data points are drawn uniformly and independently from the unit square) are applied to these new data structures, which result in new bounds for the RCP problem. Some of the new bounds significantly improve the previous results, while the others are entirely new.

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Data structures design and analysis

Keywords and phrases Closest-pair, Range search, Candidate pairs, Average-case analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.73

Related Version A full version of this paper is available at [12], http://arxiv.org/abs/1712.09749.

1 Introduction

The closest-pair problem is one of the most fundamental problems in computational geometry and finds many applications, e.g., collision detection, similarity search, traffic control, etc. In this paper, we study a range-search version of the closest-pair problem called the *range closest-pair* (RCP) problem. Let $\mathcal{X}$ be a certain collection of ranges called *query space*. The

RCP problem with query space $\mathcal{X}$ (or the $\mathcal{X}$-RCP problem for short) aims to preprocess a given dataset S of points into a low-space data structure such that when a query range $X \in \mathcal{X}$ is specified, the closest-pair in $S \cap X$ can be reported efficiently. The motivation for the RCP problem is clear and similar to that of range search: in many situations, one is interested in local information (i.e., local closest-pairs) inside specified ranges rather than global information (i.e., global closest-pair) of the dataset.

The RCP problem is quite challenging due to a couple of reasons. First, in the RCP problem, the objects of interest are in fact point-pairs instead of single points, and in a dataset there is a quadratic number of point-pairs to be dealt with. Moreover, the RCP problem is non-decomposable in the sense that even if the query range $X \in \mathcal{X}$ can be written as $X = X_1 \cup X_2$, the closest-pair in $S \cap X$ cannot be computed from the closest-pairs in $S \cap X_1$ and $S \cap X_2$. The non-decomposability makes many traditional range-search techniques inapplicable to the RCP problem, and thus makes the problem much more challenging.

The RCP problem in $\mathbb{R}^2$ has been studied in prior work over the last fifteen years, e.g., [1, 5, 6, 9, 10]. In this paper, we revisit this problem and make significant improvements to the existing solutions. Following the existing work, the query types considered in this paper are orthogonal queries (specifically, quadrants, strips, rectangles) and halfplane query.

1.1 Our contributions, techniques, and related work

The closest-pair problem and range search are both classical topics in computational geometry; see [2, 11] for references. The RCP problem is relatively new. The best existing bounds in $\mathbb{R}^2$ and our new results are summarized in Table 1 (Space refers to space cost and Qtime refers to query time), and we give a brief explanation below.

Table 1 Summary of the best existing bounds and our new results for the RCP problem in $\mathbb{R}^2$ (each row corresponds to an RCP data structure for the corresponding query space).

Query	Source	Worst-case		Average-case	
		Space	Qtime	Space	Qtime
Quadrant	[6]	$O(n \log n)$	$O(\log n)$	-	-
	Theorem 3	$\boldsymbol{O(n)}$	$\boldsymbol{O(\log n)}$	$\boldsymbol{O(\log^2 n)}$	$\boldsymbol{O(\log \log n)}$
Strip	[10]	$O(n \log^2 n)$	$O(\log n)$	-	-
	Theorem 6	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(\log n)}$	$\boldsymbol{O(n)}$	$\boldsymbol{O(\log n)}$
Rectangle	[6]	$O(n \log^5 n)$	$O(\log^2 n)$	-	-
	[10]	$O(n \log^3 n)$	$O(\log^3 n)$	-	-
	[6]	-	-	$O(n \log^4 n)$	$O(\log^4 n)$
	Theorem 15	$\boldsymbol{O(n \log^2 n)}$	$\boldsymbol{O(\log^2 n)}$	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(\log n)}$
Halfplane	[1]	$O(n \log n)$	$O(n^{0.5+\varepsilon})$	-	-
	Theorem 18	$\boldsymbol{O(n)}$	$\boldsymbol{O(\log n)}$	$\boldsymbol{O(\log^2 n)}$	$\boldsymbol{O(\log \log n)}$

Related work. The RCP problem for orthogonal queries was studied in [6, 9, 10]. The best known solution for quadrant query was given by [6], while [10] gave the best known solution for strip query. For rectangle query, there are two best known solutions (in terms of worst-case bounds) given by [6] and [10] respectively. The above results only considered worst-case performance of the data structures. The authors of [6] for the first time applied average-case analysis to RCP data structures in the model where the data points are drawn independently and uniformly from the unit square. Unfortunately, [6] only gave a rectangle

RCP data structure with low average-case *preprocessing* time, while its average-case space cost and query time are even higher than the worst-case counterparts of the data structure given by [10] (even worse, its worst-case space cost is super-quadratic). In fact, in terms of space cost and query time, no nontrivial average-case bounds were known for any kind of query before this paper. The RCP problem for halfplane query was studied in [1]. Two data structures were proposed. We only present the first one in Table 1. The second one (not in the table), while having higher space cost and query time than the first one, can be built in $O(n \log^2 n)$ time. Both data structures require (worst-case) super-linear space cost and polynomial query time.

Our contributions. In this paper, we improve all the above results by giving new RCP data structures for various query types. The improvements can be seen in Table 1. In terms of worst-case bounds, the highlights are our rectangle RCP data structure which simultaneously improves the two best known results (given by [6] and [10]) and our halfplane RCP data structure which is *optimal* and significantly improves the bounds in [1]. Furthermore, by applying average-case analysis to our new data structures, we establish the first nontrivial average-case bounds for all the query types studied. Our average-case analysis applies to datasets generated in not only the unit square but also an arbitrary axis-parallel rectangle. These average-case bounds demonstrate that our new data structures might have much better performance in practice than one can expect from the worst-case bounds. Finally, we also give an $O(n \log^2 n)$-time algorithm to build our halfplane RCP data structure, matching the preprocessing time in [1]. The preprocessing for our orthogonal RCP data structures is not considered in this paper; we are still in the process of investigating this.

Our techniques. An important notion in our techniques is that of a *candidate pair*, i.e., a pair of data points that is the answer to some RCP query. Our solutions for the quadrant and strip RCP problems use the candidate pairs to construct a planar subdivision and take advantage of point-location techniques to answer queries. The data structures themselves are simple, and our main technical contribution here occurs in the average-case analysis of the data structures. The analysis requires a study of the expected number of candidate pairs in a random dataset, which is of both geometric and combinatorial interest. Our data structure for the rectangle RCP problem is subtle; it is constructed by properly combining two simpler data structures, each of which partially achieves the desired bounds. The high-level framework of the two simpler data structures is identical: it first "decomposes" a rectangle query into four quadrant queries and then simplifies the problem via some geometric observations similar to those in the standard divide-and-conquer algorithm for the classical closest-pair problem. Also, the analysis of the data structures is technically interesting. Our solution for the halfplane RCP problem applies the duality technique to map the candidate pairs to wedges in the dual space and form a planar subdivision, which allows us to solve the problem by using point-location techniques on the subdivision, similarly to the approach for the quadrant and strip RCP problems. However, unlike the quadrant and strip cases, to bound the complexity of the subdivision here is much more challenging, which requires non-obvious observations using the properties of duality and the problem itself. The average-case bounds of the data structure follow from a technical result bounding the expected number of candidate pairs, which also involves a nontrivial proof.

Organization. Section 1.2 presents the notations and preliminaries that are used throughout the paper. We suggest that the readers read this section carefully before moving on. Our solutions for quadrant, strip, rectangle, and halfplane queries are presented in Section 2, 3, 4, and 5, respectively. In Section 6, we conclude our results and give some open questions for future work. Due to space limitations, proofs are omitted in this paper and can be found in the full version [12]. For the convenience of the reader, for some technical lemmas and theorems, we give short proof sketches in this paper which provide an overview of the proofs. Also, the details of the preprocessing algorithm for our halfplane RCP data structure is presented in [12].

1.2 Notations and Preliminaries

We introduce the notations and preliminaries that are used throughout the paper.

Query spaces. The following notations denote various query spaces (i.e., collections of ranges in $\mathbb{R}^2$): $\mathcal{Q}$ quadrants, $\mathcal{P}$ strips, $\mathcal{U}$ 3-sided rectangles, $\mathcal{R}$ rectangles, $\mathcal{H}$ halfplanes (quadrants, strips, 3-sided rectangles, rectangles under consideration are all axis-parallel). Define $\mathcal{Q}^{\nearrow} = \{[x,\infty) \times [y,\infty) : x, y \in \mathbb{R}\} \subseteq \mathcal{Q}$ as the sub-collection of all northeast quadrants, and define $\mathcal{Q}^{\nwarrow}$, $\mathcal{Q}^{\searrow}$, $\mathcal{Q}^{\swarrow}$ similarly. Define $\mathcal{P}^{\text{v}} = \{[x_1, x_2] \times \mathbb{R} : x_1, x_2 \in \mathbb{R}\} \subseteq \mathcal{P}$ as the sub-collection of all vertical strips, and similarly $\mathcal{P}^{\text{h}}$ horizontal strips. If l is a vertical (resp., horizontal) line, an *l-anchored* strip is a vertical (resp., horizontal) strip containing l; define $\mathcal{P}_l \subseteq \mathcal{P}$ as the sub-collection of all l-anchored strips. Define $\mathcal{U}^{\downarrow} = \{[x_1, x_2] \times (-\infty, y] : x_1, x_2, y \in \mathbb{R}\} \subseteq \mathcal{U}$ as the sub-collection of all bottom-unbounded 3-sided rectangles, and define $\mathcal{U}^{\uparrow}$,$\mathcal{U}^{\leftarrow}$,$\mathcal{U}^{\rightarrow}$ similarly. If l is a non-vertical line, denote by $l^{\uparrow}$ (resp., $l^{\downarrow}$) the halfplane above (resp., below) l; define $\mathcal{H}^{\uparrow} = \{l^{\uparrow} : l \text{ is a non-vertical line}\} \subseteq \mathcal{H}$ (resp., $\mathcal{H}^{\downarrow} = \{l^{\downarrow} : l \text{ is a non-vertical line}\} \subseteq \mathcal{H}$).

Candidate pairs. For a dataset S and query space $\mathcal{X}$, a *candidate pair* of S with respect to $\mathcal{X}$ refers to a pair of points in S which is the closest-pair in $S \cap X$ for some $X \in \mathcal{X}$. We denote by $\Phi(S, \mathcal{X})$ the set of the candidate pairs of S with respect to $\mathcal{X}$. If l is a line, we define $\Phi_l(S, \mathcal{X}) \subseteq \Phi(S, \mathcal{X})$ as the subset consisting of the candidate pairs that cross l (i.e., whose two points are on opposite sides of l).

Data structures. For a data structure $\mathcal{D}$, we denote by $\mathcal{D}(S)$ the data structure instance of $\mathcal{D}$ built on the dataset S. The notations $\mathsf{Space}(\mathcal{D}(S))$ and $\mathsf{Qtime}(\mathcal{D}(S))$ denote the space cost and query time (i.e., the maximum time for answering a query) of $\mathcal{D}(S)$, respectively.

Random datasets. If X is a region in $\mathbb{R}^2$ (or more generally in $\mathbb{R}^d$), we write $S \propto X^n$ to mean that S is a dataset of n random points drawn independently from the uniform distribution $\mathsf{Uni}(X)$ on X. More generally, if $X_1, \dots, X_n$ are regions in $\mathbb{R}^2$ (or more generally in $\mathbb{R}^d$), we write $S \propto \prod_{i=1}^{n} X_i$ to mean that S is a dataset of n random points drawn independently from $\mathsf{Uni}(X_1), \dots, \mathsf{Uni}(X_n)$ respectively.

Other notions. For a point $a \in \mathbb{R}^2$, we denote by $a.x$ and $a.y$ the x-coordinate and y-coordinate of a, respectively. For $a, b \in \mathbb{R}^d$, we use $\text{dist}(a, b)$ to denote the Euclidean distance between a and b, and use $[a, b]$ to denote the segments connecting a and b (in $\mathbb{R}^1$ this coincides with the notation for a closed interval). We say $I_1, \dots, I_n$ are vertical (resp., horizontal) *aligned* segments in $\mathbb{R}^2$ if there exist $r_1, \dots, r_n, \alpha, \beta \in \mathbb{R}$ such that $I_i = \{r_i\} \times [\alpha, \beta]$ (resp., $I_i = [\alpha, \beta] \times \{r_i\}$). The *length* of a pair $\phi = (a, b)$ of points is the length of the segment $[a, b]$. For $S \subseteq \mathbb{R}^2$ of size at least 2, the notation $\kappa(S)$ denotes the *closest-pair distance* of S, i.e., the length of the closest-pair in S.

The following result regarding the closest-pair distance of a random dataset will be used to bound the expected number of candidate pairs with respect to various query spaces.

▶ **Lemma 1.** *Let R be a rectangle of size $\Delta \times \Delta'$ where $\Delta \leq \Delta'$, and $A \propto R^m$. Then*

$$\mathbb{E}[\kappa^p(A)] = \Theta\left(\max\left\{(\Delta'/m^2)^p, (\sqrt{\Delta\Delta'}/m)^p\right\}\right) \text{ for any constant } p > 1.$$

In particular, if R is a segment of length ℓ, then $\mathbb{E}[\kappa^p(A)] = \Theta((\ell/m^2)^p)$.

2 Quadrant query

We consider the RCP problem for quadrant queries, i.e., the $\mathcal{Q}$-RCP problem. In order to solve the $\mathcal{Q}$-RCP problem, it suffices to consider the $\mathcal{Q}^{\swarrow}$-RCP problem. Let $S \subseteq \mathbb{R}^2$ be a dataset of size n. Suppose $\Phi(S, \mathcal{Q}^{\swarrow}) = \{\phi_1, \dots, \phi_m\}$ where $\phi_i = (a_i, b_i)$, and assume $\phi_1, \dots, \phi_m$ are sorted in increasing order of their lengths. It was shown in [6] that $m = O(n)$. We construct a mapping $\Phi(S, \mathcal{Q}^{\swarrow}) \to \mathbb{R}^2$ as $\phi_i \mapsto w_i$ where $w_i = (\max\{a_i.x, b_i.x\}, \max\{a_i.y, b_i.y\})$, and observe that for a query range $Q \in \mathcal{Q}^{\swarrow}$, ϕ_i is contained in Q iff $w_i \in Q$. Let W_i be the northeast quadrant with vertex w_i. Then we further have $w_i \in Q$ iff $q \in W_i$ where q is the vertex of Q. As such, the closest-pair in $S \cap Q$ to be reported is ϕ_η for $\eta = \min\{i : q \in W_i\}$. We create a planar subdivision Γ, by successively overlaying $W_1, \dots, W_m$ (see Figure 1). Note that the complexity of Γ is $O(m)$, since overlaying each quadrant creates at most two vertices of Γ. By the above observation, the answer for Q is ϕ_i iff q is in the cell $W_i \backslash \bigcup_{j=1}^{i-1} W_j$. Thus, we can use the optimal planar point-location data structures (e.g., [4, 8]) to solve the problem in $O(m)$ space with $O(\log m)$ query time. Since $m = O(n)$, we obtain a $\mathcal{Q}$-RCP data structure using $O(n)$ space with $O(\log n)$ query time in worst-case.

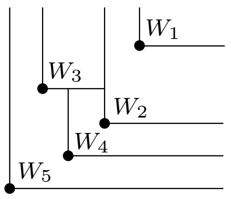


Figure 1 The subdivision induced by successively overlaying the quadrants.

Next, we analyze the average-case performance of the above data structure. In fact, it suffices to bound the expected number of the candidate pairs. Surprisingly, we have the following poly-logarithmic bound.

▶ **Lemma 2.** *For $S \propto R^n$ where R is an axis-parallel rectangle, $\mathbb{E}[|\Phi(S, \mathcal{Q})|] = O(\log^2 n)$.*

Proof Sketch. Assume $R = [0, 1] \times [0, \Delta]$ w.o.l.g. It suffices to show $\mathbb{E}[|\Phi(S, \mathcal{Q}^{\swarrow})|] = O(\log^2 n)$. Let $a_1, \dots, a_n$ be the n random points in S, and $E_{i,j}$ be the event $(a_i, a_j) \in \Phi(S, \mathcal{Q}^{\swarrow})$. By linearity of expectation, one can see that $\mathbb{E}[|\Phi(S, \mathcal{Q}^{\swarrow})|] = O(n^2 \cdot \Pr[E_{1,2}])$. So it suffices to bound $\Pr[E_{1,2}]$. Define random variables $x_{\max} = \max\{a_1.x, a_2.x\}$, $y_{\max} = \max\{a_1.y, a_2.y\}$, and define $x_{\min}, y_{\min}$ similarly. The quadrant $Q = (-\infty, x_{\max}] \times (-\infty, y_{\max}]$ is the *minimal* quadrant containing a_1, a_2, and thus $(a_1, a_2) \in \Phi(S, \mathcal{Q}^{\swarrow})$ iff (a_1, a_2) is the closest-pair in $S \cap Q$. Define $\Lambda = \{i \geq 3 : a_i \in Q\}$, which is a random subset of $\{3, \dots, n\}$.

We bound $\Pr[E_{1,2}]$ through several steps. First, we fix the values of $x_{\max}, y_{\max}, \Lambda$ and study the corresponding conditional probability of $E_{1,2}$. Fixing $\tilde{x} \in (0, 1]$, $\tilde{y} \in (0, \Delta]$, and nonempty $J \subseteq \{3, \dots, n\}$, let $C_{\tilde{x},\tilde{y},J}$ be the event $(x_{\max} = \tilde{x}) \wedge (y_{\max} = \tilde{y}) \wedge (\Lambda = J)$. Consider $\Pr[E_{1,2} \mid C_{\tilde{x},\tilde{y},J}]$. Let $\delta_x = x_{\max} - x_{\min}$ and $\delta_y = y_{\max} - y_{\min}$. We observe that under the condition $C_{\tilde{x},\tilde{y},J}$, $E_{1,2}$ happens only if $(\delta_x \leq \kappa(S_J)) \wedge (\delta_y \leq \kappa(S_J))$, where $S_J = \{a_j : j \in J\}$. Furthermore, under $C_{\tilde{x},\tilde{y},J}$, the $|J|$ random points in S_J can be viewed as independently drawn from the uniform distribution on the rectangle $[0, \tilde{x}] \times [0, \tilde{y}]$. As such, Lemma 1 can be applied to understand the behavior of $\kappa(S_J)$. By properly applying Lemma 1 and doing some careful calculations, we deduce that under the condition $C_{\tilde{x},\tilde{y},J}$, $(\delta_x \leq \kappa(S_J)) \wedge (\delta_y \leq \kappa(S_J))$ happens with probablity $O(1/|J|^2)$. Thus, $\Pr[E_{1,2} \mid C_{\tilde{x},\tilde{y},J}] = O(1/|J|^2)$. This further implies $\Pr[E_{1,2} \mid |\Lambda| = m] = O(1/m^2)$ for all $m \in \{1, \dots, n-2\}$. With this in hand, to bound $\Pr[E_{1,2}]$, it suffices to study $\Pr[|\Lambda| = m]$. To calculate $\Pr[|\Lambda| = m]$ is in fact a combinatorial problem, because $|\Lambda|$ only depends on the orderings of the x-coordinates and y-coordinates of $a_1,\dots,a_n$. Using combinatorial arguments, we obtain $\Pr[|\Lambda| = m] = O((m+1)\log n/n^2)$. Finally, applying the formula $\Pr[E_{1,2}] = \sum_{m=0}^{n-2} \Pr[|\Lambda| = m] \cdot \Pr[E_{1,2} \mid |\Lambda| = m]$ and the bounds achieved, we have $\Pr[E_{1,2}] = O(\log^2 n/n^2)$. As a result, $\mathbb{E}[|\Phi(S,\mathcal{Q}^{\swarrow})|] = O(\log^2 n)$ and $\mathbb{E}[|\Phi(S,\mathcal{Q})|] = O(\log^2 n)$. A complete proof can be found in [12]. ◀

Using the above lemma, we can immediately conclude that our data structure uses $O(\log^2 n)$ space in average-case. The average-case query time is in fact $O(\mathbb{E}[\log|\Phi(S,\mathcal{Q})|])$. Note that $\mathbb{E}[\log x] \le \log \mathbb{E}[x]$ for a positive random variable x, thus $\mathbb{E}[\log|\Phi(S,\mathcal{Q})|] = O(\log\log n)$.

▶ **Theorem 3.** *There exists a $\mathcal{Q}$-RCP data structure $\mathcal{A}$ such that*

- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{A}(S)) = O(n)$ and $\mathsf{Qtime}(\mathcal{A}(S)) = O(\log n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{A}(S))] = O(\log^2 n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{A}(S))] = O(\log\log n)$.*

3 Strip query

We consider the RCP problem for strip queries, i.e., the $\mathcal{P}$-RCP problem. In order to solve the $\mathcal{P}$-RCP problem, it suffices to consider the $\mathcal{P}^{\mathrm{v}}$-RCP problem. Let $S \subseteq \mathbb{R}^2$ be a dataset of size n. Suppose $\Phi(S,\mathcal{P}^{\mathrm{v}}) = \{\phi_1,\dots,\phi_m\}$ where $\phi_i = (a_i,b_i)$, and assume $\phi_1,\dots,\phi_m$ are sorted in increasing order of their lengths. It was shown in [10] that $m = O(n\log n)$. We construct a mapping $\Phi(S,\mathcal{P}^{\mathrm{v}}) \to \mathbb{R}^2$ as $\phi_i \mapsto w_i$ where $w_i = (\min\{a_i.x, b_i.x\}, \max\{a_i.x, b_i.x\})$, and observe that for a query range $P = [x_1,x_2]\times\mathbb{R} \in \mathcal{P}^{\mathrm{v}}$, ϕ_i is contained in P iff w_i is in the southeast quadrant $[x_1,\infty)\times(-\infty,x_2]$. Let W_i be the northwest quadrant with vertex w_i. Then we further have $w_i \in [x_1,\infty)\times(-\infty,x_2]$ iff $p \in W_i$ where $p = (x_1,x_2)$. As such, the closest-pair in $S \cap P$ is ϕ_η for $\eta = \min\{i : p \in W_i\}$. Thus, as in Section 2, we can successively overlay $W_1,\dots,W_m$ to create a planar subdivision, and use point-location to solve the problem in $O(m)$ space and $O(\log m)$ query time. Since $m = O(n\log n)$ here, we obtain a $\mathcal{P}$-RCP data structure using $O(n\log n)$ space with $O(\log n)$ query time in worst-case.

Next, we analyze the average-case performance of our data structure. Again, it suffices to bound the expected number of the candidate pairs. For later use, we study here a more general case in which the queries are 3-sided rectangles.

▶ **Lemma 4.** *Let $S \propto \prod_{i=1}^{n} I_i$ where $I_1,\dots,I_n$ are distinct vertical (resp., horizontal) aligned segments sorted from left to right (resp., from bottom to top). Suppose $a_i \in S$ is the point drawn on I_i. Then for $i,j \in \{1,\dots,n\}$ with $i<j$ and $\mathcal{X} \in \{\mathcal{U}^{\downarrow},\mathcal{U}^{\uparrow}\}$ (resp., $\mathcal{X} \in \{\mathcal{U}^{\leftarrow},\mathcal{U}^{\rightarrow}\}$),*

$$\Pr[(a_i,a_j) \in \Phi(S,\mathcal{X})] = O\left(\frac{\log(j-i)}{(j-i)^2}\right).$$

From the above lemma, a direct calculation gives us the following corollary.

▶ **Corollary 5.** *For $S \propto R^n$ where R is an axis-parallel rectangle, $\mathbb{E}[|\Phi(S,\mathcal{U})|] = \Theta(n)$ and $\mathbb{E}[|\Phi(S,\mathcal{P})|] = \Theta(n)$.*

Using the above argument and our previous data structure, we conclude the following.

▶ **Theorem 6.** *There exists a $\mathcal{P}$-RCP data structure $\mathcal{B}$ such that*

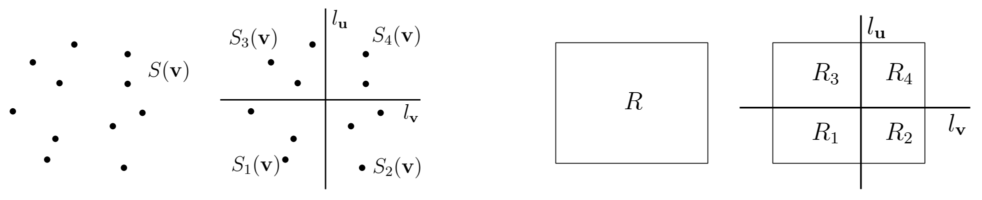
- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{B}(S)) = O(n\log n)$ and $\mathsf{Qtime}(\mathcal{B}(S)) = O(\log n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{B}(S))] = O(n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{B}(S))] = O(\log n)$.*

4 Rectangle query

We consider the RCP problem for rectangle queries, i.e., the $\mathcal{R}$-RCP problem. Interestingly, our final solution for the $\mathcal{R}$-RCP problem is a combination of two simpler solutions, each of which partially achieves the desired bounds.

We first describe the common part of our two solutions. Let $S \subseteq \mathbb{R}^2$ be a dataset of size n. The common component of our two data structures is a standard 2D range tree built on S [3]. The main tree (or primary tree) $\mathcal{T}$ is a range tree built on the x-coordinates of the points in S. Each node $\mathbf{u} \in \mathcal{T}$ corresponds to a subset $S(\mathbf{u})$ of x-consecutive points in S, called the *canonical subset* of $\mathbf{u}$. At $\mathbf{u}$, there is an associated secondary tree $\mathcal{T}_\mathbf{u}$, which is a range tree built on the y-coordinates of the points in $S(\mathbf{u})$. With an abuse of notation, for each node $\mathbf{v} \in \mathcal{T}_\mathbf{u}$, we still use $S(\mathbf{v})$ to denote the canonical subset of $\mathbf{v}$, which is a subset of y-consecutive points in $S(\mathbf{u})$. As in [6], for each (non-leaf) primary node $\mathbf{u} \in \mathcal{T}$, we fix a vertical line $l_\mathbf{u}$ such that the points in the canonical subset of the left (resp., right) child of $\mathbf{u}$ are to the left (resp., right) of $l_\mathbf{u}$. Similarly, for each (non-leaf) secondary node $\mathbf{v}$, we fix a horizontal line $l_\mathbf{v}$ such that the points in the canonical subset of the left (resp., right) child of $\mathbf{v}$ are above (resp., below) $l_\mathbf{v}$. Let $\mathbf{v} \in \mathcal{T}_\mathbf{u}$ be a secondary node. Then at $\mathbf{v}$ we have two lines $l_\mathbf{v}$ and $l_\mathbf{u}$, which partition $\mathbb{R}^2$ into four quadrants. We denote by $S_1(\mathbf{v}), \dots, S_4(\mathbf{v})$ the subsets of $S(\mathbf{v})$ contained in these quadrants; see Figure 2a for the correspondence. In order to solve the problem, we need to store some additional data structures at the nodes of the tree (called *sub-structures*). At each secondary node $\mathbf{v}$, we store four $\mathcal{Q}$-RCP data structures $\mathcal{A}(S_1(\mathbf{v})), \dots, \mathcal{A}(S_4(\mathbf{v}))$ (Theorem 3).

Now let us explain what we can do by using this 2D range tree (with the sub-structures). Let $R = [x_1, x_2] \times [y_1, y_2] \in \mathcal{R}$ be a query rectangle. We first find in $\mathcal{T}$ the *splitting* node $\mathbf{u} \in \mathcal{T}$ corresponding to the range $[x_1, x_2]$, which is by definition the LCA of all the leaves whose corresponding points are in $[x_1, x_2] \times \mathbb{R}$. Then we find in $\mathcal{T}_\mathbf{u}$ the splitting node $\mathbf{v} \in \mathcal{T}_\mathbf{u}$ corresponding to the range $[y_1, y_2]$. If either of the splitting nodes does not exist or is a leaf node, then $|S \cap R| \leq 1$ and nothing should be reported. So assume $\mathbf{u}$ and $\mathbf{v}$ are non-leaf nodes. By the property of splitting node, we have $S \cap R = S(\mathbf{v}) \cap R$, and the lines $l_\mathbf{u}$ and $l_\mathbf{v}$ both intersect R. Thus, $l_\mathbf{u}$ and $l_\mathbf{v}$ decompose R into four smaller rectangles $R_1, \dots, R_4$; see Figure 2b for the correspondence. By construction, we have $S(\mathbf{v}) \cap R_i = S_i(\mathbf{v}) \cap R_i$. In order to find the closest-pair in $S \cap R$, we first try to compute the closest-pair in $S \cap R_i$ for all $i \in \{1, \dots, 4\}$. This can be done by querying the sub-structures stored at $\mathbf{v}$. Indeed, $S \cap R_i = S(\mathbf{v}) \cap R_i = S_i(\mathbf{v}) \cap R_i = S_i(\mathbf{v}) \cap Q_i$, where Q_i is the quadrant obtained by removing the two sides of R_i that coincide with $l_\mathbf{u}$ and $l_\mathbf{v}$. Therefore, we can query $\mathcal{A}(S_i(\mathbf{v}))$ with Q_i to find the closest-pair in $S \cap R_i$. Once the four closest-pairs are computed, we take the shortest one (i.e., the one of the smallest length) among them and denote it by ϕ.



(a) Illustrating the subsets $S_1(\mathbf{v}), \dots, S_4(\mathbf{v})$. **(b)** Illustrating the rectangles $R_1, \dots, R_4$.

Figure 2 Illustrating $S_1(\mathbf{v}), \dots, S_4(\mathbf{v})$ and $R_1, \dots, R_4$.

Clearly, ϕ is not necessarily the closest-pair in $S \cap R$ as the two points in the closest-pair may belong to different R_i's. However, as we will see, with ϕ in hand, finding the closest-pair in $S \cap R$ becomes easier. Suppose $l_{\mathbf{u}} : x = \alpha$ and $l_{\mathbf{v}} : y = \beta$, where $x_1 \leq \alpha \leq x_2$ and $y_1 \leq \beta \leq y_2$. Let δ be the length of ϕ. We define $P_\alpha = [\alpha - \delta, \alpha + \delta] \times \mathbb{R}$ (resp., $P_\beta = \mathbb{R} \times [\beta - \delta, \beta + \delta]$) and $R_\alpha = R \cap P_\alpha$ (resp., $R_\beta = R \cap P_\beta$); see Figure 3. We have the following key observation.

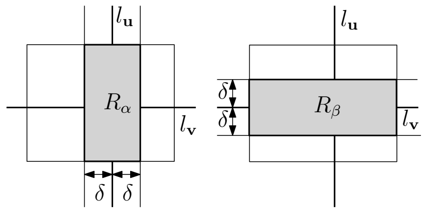


Figure 3 Illustrating the rectangles R_α and R_β.

▶ **Lemma 7.** *The closest-pair in $S \cap R$ is the shortest one among $\{\phi, \phi_\alpha, \phi_\beta\}$, where ϕ_α (resp., ϕ_β) is the closest-pair in $S \cap R_\alpha$ (resp., $S \cap R_\beta$).*

Due to the above lemma, it now suffices to compute ϕ_α and ϕ_β. Note that R_α and R_β are rectangles, so computing ϕ_α and ϕ_β still requires rectangle RCP queries. Fortunately, there are some additional properties which make it easy to search for the closest-pairs in $S \cap R_\alpha$ and $S \cap R_\beta$. For a set A of points in $\mathbb{R}^2$ and $a, b \in A$, we define the *x-gap* (resp., *y-gap*) between a and b in A as the number of the points in $A \backslash \{a, b\}$ whose x-coordinates (resp., y-coordinates) are in between $a.x$ and $b.x$ (resp., $a.y$ and $b.y$).

▶ **Lemma 8.** *There exists a constant integer k such that the y-gap (resp., x-gap) between the two points of ϕ_α (resp., ϕ_β) in $S \cap R_\alpha$ (resp., $S \cap R_\beta$) is at most k.*

We shall properly use the above lemma to help compute ϕ_α and ϕ_β. At this point, our two solutions diverge.

4.1 Preliminary: extreme point data structures

Before presenting our solutions, we introduce the so-called *top/bottom extreme point* (TBEP) and *left/right extreme point* (LREP) data structures. For a query space $\mathcal{X}$ and a constant integer k, an $(\mathcal{X}, k)$-TBEP (resp. $(\mathcal{X}, k)$-LREP) data structure stores a given set S of points in $\mathbb{R}^2$ and can report the k topmost/bottommost (resp., leftmost/rightmost) points in $S \cap X$ for a query range $X \in \mathcal{X}$.

▶ **Lemma 9.** *Let k be a constant integer. There exists a $(\mathcal{P}^{\mathrm{v}}, k)$-TBEP data structure $\mathcal{K}^{\mathrm{v}}$ such that for any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{K}^{\mathrm{v}}(S)) = O(n)$ and $\mathsf{Qtime}(\mathcal{K}^{\mathrm{v}}(S)) = O(\log n)$. Symmetrically, there also exists a $(\mathcal{P}^{\mathrm{h}}, k)$-LREP data structure $\mathcal{K}^{\mathrm{h}}$ satisfying the same bounds.*

▶ **Lemma 10.** *Let l be a vertical (resp., horizontal) line and k be a constant integer. There exists a $(\mathcal{P}_l, k)$-TBEP (resp., $(\mathcal{P}_l, k)$-LREP) data structure $\mathcal{K}_l$ such that for $S \propto \prod_{i=1}^{n} I_i$ where $I_1, \dots, I_n$ are distinct vertical (resp., horizontal) aligned segments, $\mathbb{E}[\mathsf{Space}(\mathcal{K}_l(S))] = O(\log n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{K}_l(S))] = O(\log \log n)$.*

We remark here that the TBEP (resp., LREP) data structures are essentially top-k data structures when using the y-coordinates (resp., x-coordinates) as weights. Therefore, a 1D top-k data structure (see for example [7]) can be used to prove Lemma 9. For completeness, we also give a proof in the full version [12].

4.2 First solution

We now introduce our first solution, which achieves the desired worst-case bounds. Let k be the constant integer in Lemma 8. In our first solution, besides the 2D range tree presented before, we build additionally two 1D range trees $\mathcal{T}'$ and $\mathcal{T}''$ on S, where $\mathcal{T}'$ (resp., $\mathcal{T}''$) is built on y-coordinates (resp., x-coordinates). For $\mathbf{u}' \in \mathcal{T}'$ (resp., $\mathbf{u}'' \in \mathcal{T}''$), we still use $S(\mathbf{u}')$ (resp., $S(\mathbf{u}'')$) to denote the canonical subset of $\mathbf{u}'$ (resp., $\mathbf{u}'' \in \mathcal{T}''$). At each node $\mathbf{u}' \in \mathcal{T}'$, we store a $\mathcal{P}$-RCP data structure $\mathcal{B}(S(\mathbf{u}'))$ (Theorem 6) and a $(\mathcal{P}^{\mathrm{v}}, k)$-TBEP data structure $\mathcal{K}^{\mathrm{v}}(S(\mathbf{u}'))$ (Lemma 9). Similarly, at each node $\mathbf{u}'' \in \mathcal{T}''$, we store a $\mathcal{P}$-RCP data structure $\mathcal{B}(S(\mathbf{u}''))$ (Theorem 6) and a $(\mathcal{P}^{\mathrm{h}}, k)$-LREP data structure $\mathcal{K}^{\mathrm{h}}(S(\mathbf{u}''))$ (Lemma 9).

We now explain how to compute ϕ_α and ϕ_β. Suppose $R_\alpha = [x_\alpha, x'_\alpha] \times [y_\alpha, y'_\alpha]$. Let $P_x = [x_\alpha, x'_\alpha] \times \mathbb{R}$ and $P_y = \mathbb{R} \times [y_\alpha, y'_\alpha]$. To compute ϕ_α, we first find in $\mathcal{T}'$ the $t = O(\log n)$ canonical nodes $\mathbf{u}'_1, \dots, \mathbf{u}'_t \in \mathcal{T}'$ corresponding to the range $[y_\alpha, y'_\alpha]$. Then $\bigcup_{i=1}^{t} S(\mathbf{u}'_i) = S \cap P_y$, and each $S(\mathbf{u}'_i)$ is a set of y-consecutive points in $S \cap P_y$. Furthermore, $S \cap R_\alpha = \bigcup_{i=1}^{t} S(\mathbf{u}'_i) \cap P_x$. We query the sub-structures $\mathcal{B}(S(\mathbf{u}'_1)), \dots, \mathcal{B}(S(\mathbf{u}'_t))$ with P_x to find the closest-pairs $\phi_1, \dots, \phi_t$ in $S(\mathbf{v}_1) \cap P_x, \dots, S(\mathbf{v}_t) \cap P_x$, respectively. We also query $\mathcal{K}^{\mathrm{v}}(S(\mathbf{u}'_1)), \dots, \mathcal{K}^{\mathrm{v}}(S(\mathbf{u}'_t))$ with P_x to obtain the k topmost and bottommost points in $S(\mathbf{u}'_1) \cap P, \dots, S(\mathbf{u}'_t) \cap P$, respectively; we denote by K the set of the $2kt$ reported points. Then we find the closest-pair ϕ_K in K using the standard divide-and-conquer algorithm. We claim that ϕ_α is the shortest one among $\{\phi_1, \dots, \phi_t, \phi_K\}$. Suppose $\phi_\alpha = (a, b)$. If the two points of ϕ_α are both contained in some $S(\mathbf{u}'_i)$, then clearly $\phi_\alpha = \phi_i$. Otherwise, by Lemma 8 and the choice of k, the two points of ϕ_α must belong to K and hence $\phi_\alpha = \phi_K$. It follows that $\phi_\alpha \in \{\phi_1, \dots, \phi_t, \phi_K\}$. Furthermore, because the pairs $\phi_1, \dots, \phi_t, \phi_K$ are all contained in R_α, ϕ_α must be the shortest one among $\{\phi_1, \dots, \phi_t, \phi_K\}$. Therefore, with $\phi_1, \dots, \phi_t, \phi_K$ in hand, ϕ_α can be easily computed. The pair ϕ_β is computed symmetrically using $\mathcal{T}''$. Finally, taking the shortest one among $\{\phi, \phi_\alpha, \phi_\beta\}$, the query R can be answered.

The 2D range tree together with the two 1D range trees $\mathcal{T}'$ and $\mathcal{T}''$ forms an $\mathcal{R}$-RCP data structure, which is our first solution. A straightforward analysis gives us the worst-case space cost and query time of this data structure.

▶ **Theorem 11.** *There exists an $\mathcal{R}$-RCP data structure $\mathcal{D}_1$ such that for any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{D}_1(S)) = O(n \log^2 n)$ and $\mathsf{Qtime}(\mathcal{D}_1(S)) = O(\log^2 n)$.*

Our first solution itself already achieves the desired worst-case bounds, which simultaneously improves the results given in [6] and [10].

4.3 Second solution

We now introduce our second solution, which has the desired average-case space cost and an $O(\log n)$ query time (even in worst-case). In our second solution, we only use the 2D range tree presented before, but we need some additional sub-structures stored at each secondary node. Let k be the constant integer in Lemma 8. Define $S_\blacktriangle(\mathbf{v}) = S_3(\mathbf{v}) \cup S_4(\mathbf{v})$ (resp., $S_\blacktriangledown(\mathbf{v}) = S_1(\mathbf{v}) \cup S_2(\mathbf{v})$) as the subset of $S(\mathbf{v})$ consisting of the points above (resp., below) $l_\mathbf{v}$. Similarly, define $S_\blacktriangleleft(\mathbf{v})$ and $S_\blacktriangleright(\mathbf{v})$ as the subsets to the left and right of $l_\mathbf{u}$, respectively. Let $\mathbf{v} \in \mathcal{T}_\mathbf{u}$ be a secondary node. Besides $\mathcal{A}(S_1(\mathbf{v})), \dots, \mathcal{A}(S_4(\mathbf{v}))$, we store at $\mathbf{v}$ two $(\mathcal{P}_{l_\mathbf{u}}, k)$-TBEP data structures $\mathcal{K}_{l_\mathbf{u}}(S_\blacktriangle(\mathbf{v})), \mathcal{K}_{l_\mathbf{u}}(S_\blacktriangledown(\mathbf{v}))$ (Lemma 10) and two $(\mathcal{P}_{l_\mathbf{v}}, k)$-LREP data structures $\mathcal{K}_{l_\mathbf{v}}(S_\blacktriangleleft(\mathbf{v})), \mathcal{K}_{l_\mathbf{v}}(S_\blacktriangleright(\mathbf{v}))$ (Lemma 10). Furthermore, we need a new kind of sub-structures called *range shortest-segment* (RSS) data structures. For a query space $\mathcal{X}$, an $\mathcal{X}$-RSS data structure stores a given set of segments in $\mathbb{R}^2$ and can report the shortest segment contained in a query range $X \in \mathcal{X}$. We have the following observation.

▶ **Lemma 12.** *There exists a $\mathcal{U}$-RSS data structure $\mathcal{C}$ such that for any set G of m segments in $\mathbb{R}^2$, $\mathsf{Space}(\mathcal{C}(G)) = O(m^2)$ and $\mathsf{Qtime}(\mathcal{C}(G)) = O(\log m)$.*

Define $\Phi_{\blacktriangle}(\mathbf{v}) = \Phi_{l_\mathbf{u}}(S_{\blacktriangle}(\mathbf{v}),\mathcal{U}^{\downarrow})$, $\Phi_{\blacktriangledown}(\mathbf{v}) = \Phi_{l_\mathbf{u}}(S_{\blacktriangledown}(\mathbf{v}),\mathcal{U}^{\uparrow})$, $\Phi_{\blacktriangleleft}(\mathbf{v}) = \Phi_{l_\mathbf{v}}(S_{\blacktriangleleft}(\mathbf{v}),\mathcal{U}^{\rightarrow})$, $\Phi_{\blacktriangleright}(\mathbf{v}) = \Phi_{l_\mathbf{v}}(S_{\blacktriangleright}(\mathbf{v}),\mathcal{U}^{\leftarrow})$. We can view $\Phi_{\blacktriangle}(\mathbf{v}), \Phi_{\blacktriangledown}(\mathbf{v}), \Phi_{\blacktriangleleft}(\mathbf{v}), \Phi_{\blacktriangleright}(\mathbf{v})$ as four sets of segments by identifying each point-pair (a, b) as a segment $[a, b]$. Then we store at $\mathbf{v}$ four $\mathcal{U}$-RSS data structures $\mathcal{C}(\Phi_{\blacktriangle}(\mathbf{v})), \mathcal{C}(\Phi_{\blacktriangledown}(\mathbf{v})), \mathcal{C}(\Phi_{\blacktriangleleft}(\mathbf{v})), \mathcal{C}(\Phi_{\blacktriangle}(\mathbf{v}))$ (Lemma 12).

We now explain how to compute ϕ_α and ϕ_β. Let us consider ϕ_α. Recall that ϕ_α is the closest-pair in $S \cap R_\alpha$, i.e., in $S(\mathbf{v}) \cap R_\alpha$. Let P be the $l_\mathbf{u}$-anchored strip obtained by removing the top/bottom bounding line of R_α. If the two points of ϕ_α are on opposite sides of $l_\mathbf{v}$, then by Lemma 8 its two points must be among the k bottommost points in $S_{\blacktriangle}(\mathbf{v}) \cap P$ and the k topmost points in $S_{\blacktriangledown}(\mathbf{v}) \cap P$ respectively. Using $\mathcal{K}_{l_\mathbf{u}}(S_{\blacktriangle}(\mathbf{v}))$ and $\mathcal{K}_{l_\mathbf{u}}(S_{\blacktriangledown}(\mathbf{v}))$, we report these $2k$ points, and compute the closest-pair among them by brute-force. If the two points of ϕ_α are on the same side of $l_\mathbf{v}$, then they are both contained in either $S_{\blacktriangle}(\mathbf{v})$ or $S_{\blacktriangledown}(\mathbf{v})$. So it suffices to compute the closest-pairs in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$ and $S_{\blacktriangledown}(\mathbf{v}) \cap R_\alpha$. Without loss of generality, we only need to consider the closest-pair in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$. We denote by U the 3-sided rectangle obtained by removing the bottom boundary of R_α, and by Q_1 (resp., Q_2) the quadrant obtained by removing the right (resp., left) boundary of U. We query $\mathcal{A}(S_1(\mathbf{v}))$ with Q_1, $\mathcal{A}(S_2(\mathbf{v}))$ with Q_2, and $\mathcal{C}(\Phi_{\blacktriangle}(\mathbf{v}))$ with U. Clearly, the shortest one among the three answers is the closest-pair in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$. Indeed, the three answers are all point-pairs in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$. If the two points of the closest-pair in $S_{\blacktriangle}(\mathbf{v}) \cap R_\alpha$ are both to the left (resp., right) of $l_\mathbf{u}$, $\mathcal{A}(S_1(\mathbf{v}))$ (resp., $\mathcal{A}(S_2(\mathbf{v}))$) reports it; otherwise, the closest-pair crosses $l_\mathbf{u}$, and $\mathcal{C}(\Phi_{\blacktriangle}(\mathbf{v}))$ reports it. Now we see how to compute ϕ_α, and ϕ_β can be computed symmetrically. Finally, taking the shortest one among $\{\phi, \phi_\alpha, \phi_\beta\}$, the query R can be answered.

A straightforward analysis shows that the overall query time is $O(\log n)$ even in worst-case. The worst-case space cost is not near-linear, as the $\mathcal{U}$-RSS data structure $\mathcal{C}$ may occupy quadratic space by Lemma 12. However, we can show that the average-case space cost is in fact $O(n \log n)$. The crucial thing is to bound the average-case space of the sub-structures stored at the secondary nodes. The intuition for bounding the average-case space of the $\mathcal{Q}$-RCP and TBEP/LREP sub-structures comes directly from the average-case performance of our $\mathcal{Q}$-RCP data structure (Theorem 3) and TBEP/LREP data structure (Lemma 10). However, to bound the average-case space of the $\mathcal{U}$-RSS sub-structures is more difficult. By our construction, the segments stored in these sub-structures are 3-sided candidate pairs that cross a line. As such, we have to study the expected number of such candidate pairs in a random dataset. To this end, we recall Lemma 4. Let l be a vertical line, and $S \propto \prod_{i=1}^{n} I_i$ be a random dataset drawn from vertical aligned segments $I_1, \dots, I_n$ as in Lemma 4. Suppose we build a $\mathcal{U}$-RSS data structure $\mathcal{C}(\Phi)$ on $\Phi = \Phi_l(S,\mathcal{U}^{\downarrow})$. Using Lemma 4, a direct calculation gives us $\mathbb{E}[|\Phi_l(S,\mathcal{U}^{\downarrow})|] = O(\log^2 n)$. Unfortunately, this is not sufficient for bounding the average-case space of $\mathcal{C}(\Phi)$, because $\mathbb{E}[\mathsf{Space}(\mathcal{C}(\Phi))] = O(\mathbb{E}[|\Phi_l(S,\mathcal{U}^{\downarrow})|^2])$ and in general $\mathbb{E}[|\Phi_l(S,\mathcal{U}^{\downarrow})|^2] \neq \mathbb{E}^2[|\Phi_l(S,\mathcal{U}^{\downarrow})|]$. Therefore, we need a bound for $\mathbb{E}[|\Phi_l(S,\mathcal{U}^{\downarrow})|^2]$, which can also be obtained using Lemma 4, but requires more work.

▶ **Lemma 13.** *Let l be a vertical (resp., horizontal) line and $S \propto \prod_{i=1}^{n} I_i$ where $I_1, \dots, I_n$ are distinct vertical (resp., horizontal) aligned segments. Then for $\mathcal{X} \in \{\mathcal{U}^{\downarrow},\mathcal{U}^{\uparrow}\}$ (resp., $\mathcal{X} \in \{\mathcal{U}^{\leftarrow},\mathcal{U}^{\rightarrow}\}$) , $\mathbb{E}[|\Phi_l(S,\mathcal{X})|] = O(\log^2 n)$ and $\mathbb{E}[|\Phi_l(S,\mathcal{X})|^2] = O(\log^4 n)$.*

Now we are ready to prove the bounds of our second solution.

▶ **Theorem 14.** *There exists an $\mathcal{R}$-RCP data structure $\mathcal{D}_2$ such that*

- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Qtime}(\mathcal{D}_2(S)) = O(\log n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{D}_2(S))] = O(n \log n)$.*

Proof sketch. The query time can be shown via a direct analysis. To bound the average-case space cost, let $S \propto R^n$. Since a 2D range tree built on a dataset of n points has a fixed tree structure independent of the dataset (while depending on the number n), $\mathcal{D}_2(S)$ can be viewed as a fixed 2D range tree with random sub-structures. Let $\mathbf{v} \in \mathcal{T}_\mathbf{u}$ be a secondary node. We want to bound the expected space cost of the sub-structures stored at $\mathbf{v}$. To this end, we take advantage of Theorem 3, Lemma 10, and Lemma 13. However, before this, there is a crucial issue to be handled. We notice that Theorem 3, Lemma 10, and Lemma 13 assume the random dataset is independently and uniformly generated from either an axis-parallel rectangle or a set of aligned segments. Unfortunately, the underlying datasets of the sub-structures stored at $\mathbf{v}$, which are $S_1(\mathbf{v}), \dots, S_4(\mathbf{v})$ and $S_\blacktriangle(\mathbf{v}), S_\blacktriangledown(\mathbf{v}), S_\blacktriangleleft(\mathbf{v}), S_\blacktriangleright(\mathbf{v})$, are neither (independently and uniformly) generated from a rectangle nor generated from aligned segments. To handle this issue is the technical part of this proof, and we only give some intuition here (the details can be found in the complete proof). The key idea is to fix a configuration of the points outside the random dataset under consideration, which makes the random dataset distributed independently and uniformly on a certain rectangle. For instance, if we fix a configuration of $S \backslash S_1(\mathbf{v})$, then $S_1(\mathbf{v})$ can be viewed as independently and uniformly generated from a rectangle and thus Theorem 3 applies to bound the expected space cost of $\mathcal{A}(S_1(\mathbf{v}))$. In this way, we show that the expected space cost of the sub-structures stored at $\mathbf{v}$ is poly-logarithmic in $|S(\mathbf{v})|$. It follows immediately that $\mathbb{E}[\mathsf{Space}(\mathcal{D}_2(S))] = O(n \log n)$. A complete proof can be found in [12]. ◀

4.4 Combining the two solutions

We now combine the two data structures $\mathcal{D}_1$ (Theorem 11) and $\mathcal{D}_2$ (Theorem 14) to obtain a *single* data structure $\mathcal{D}$ that achieves the desired worst-case and average-case bounds simultaneously. For a dataset $S \subseteq \mathbb{R}^2$ of size n, if $\mathsf{Space}(\mathcal{D}_2(S)) \geq n \log^2 n$, we set $\mathcal{D}(S) = \mathcal{D}_1(S)$, otherwise we set $\mathcal{D}(S) = \mathcal{D}_2(S)$. The worst-case bounds of $\mathcal{D}$ follows directly, while the average-case bounds follows from an analysis using Markov's inequality.

▶ **Theorem 15.** *There exists an $\mathcal{R}$-RCP data structure $\mathcal{D}$ such that*
- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{D}(S)) = O(n \log^2 n)$ and $\mathsf{Qtime}(\mathcal{D}(S)) = O(\log^2 n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{D}(S))] = O(n \log n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{D}(S))] = O(\log n)$.*

5 Halfplane query

We consider the RCP problem for halfplane queries, i.e., the $\mathcal{H}$-RCP problem. In order to solve the $\mathcal{H}$-RCP problem, it suffices to consider the $\mathcal{H}^\uparrow$-RCP problem. Let $S \subseteq \mathbb{R}^2$ be the dataset of size n.

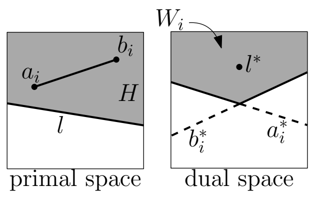


Figure 4 Illustrating the upward-open wedge W_i.

We shall apply the standard duality technique [3]. A non-vertical line $l : y = ux + v$ in $\mathbb{R}^2$ is dual to the point $l^* = (u, -v)$ and a point $p = (s, t) \in \mathbb{R}^2$ is dual to the line $p^* : y = sx - t$. A basic property of duality is that $p \in l^\uparrow$ (resp., $p \in l^\downarrow$) iff $l^* \in (p^*)^\uparrow$ (resp., $l^* \in (p^*)^\downarrow$). To make the exposition cleaner, we distinguish between *primal space* and *dual space*, which are two copies of $\mathbb{R}^2$. The dataset S and query ranges are assumed to lie in the primal space, while their dual objects are assumed to lie in the dual space. Duality allows us to transform the $\mathcal{H}^\uparrow$-RCP problem into a point location problem as follows. Let $H = l^\uparrow \in \mathcal{H}^\uparrow$ be a query range. The line l bounding H is dual to the point l^* in the dual space; for convenience, we also call l^* the dual point of H. If we decompose the dual space into "cells" such that the query ranges whose dual points lie in the same cell have the same answer, then point location techniques can be applied to solve the problem directly. Note that this decomposition must be a polygonal subdivision Γ of $\mathbb{R}^2$, which consists of vertices, straight-line edges, and polygonal faces (i.e., cells). This is because the cell-boundaries must be defined by the dual lines of the points in S. In order to analyze the space cost and query time, we need to study the complexity $|\Gamma|$ of Γ. An $O(n^2)$ trivial upper bound for $|\Gamma|$ follows from the fact that the subdivision formed by the n dual lines of the points in S has an $O(n^2)$ complexity. In what follows, we shall show $|\Gamma| = O(n)$ by using the additional properties of the problem, which is a key ingredient of our result in this section.

Suppose $\Phi(S, \mathcal{H}^\uparrow) = \{\phi_1, \dots, \phi_m\}$ where $\phi_i = (a_i, b_i)$ and $\phi_1, \dots, \phi_m$ are sorted in increasing order of their lengths. It was shown in [1] that $m = O(n)$, and the candidate pairs do not cross each other, i.e., the segments $[a_i, b_i]$ and $[a_j, b_j]$ do not cross for any $i \neq j$. The non-crossing property of the candidate pairs is important and will be used later for proving Lemma 16. With this in hand, we now consider the subdivision Γ. Let $H = l^\uparrow \in \mathcal{H}^\uparrow$ be a query range. By the property of duality, ϕ_i is contained in H iff $l^* \in (a_i^*)^\uparrow$ and $l^* \in (b_i^*)^\uparrow$, i.e., l^* is in the upward-open *wedge* W_i generated by the lines a_i^* and b_i^* (in the dual space); see Figure 4.

As such, the closest-pair in $S \cap H$ to be reported is ϕ_η for $\eta = \min\{i : l^* \in W_i\}$. Therefore, Γ can be constructed by successively overlaying the wedges $W_1, \dots, W_m$ (similarly to what we see in Section 2). Formally, we begin with a trivial subdivision Γ_0 of $\mathbb{R}^2$, which consists of only one face, the entire plane. Suppose Γ_{i-1} is constructed, which has an *outer face* F_{i-1} equal to the complement of $\bigcup_{j=1}^{i-1} W_j$ in $\mathbb{R}^2$. Now we construct a new subdivision Γ_i by "inserting" W_i to Γ_{i-1}. Specifically, Γ_i is obtained from Γ_{i-1} by decomposing the outer face F_{i-1} via the wedge W_i; that is, we decompose F_{i-1} into several smaller faces: one is $F_{i-1} \backslash W_i$ and the others are the connected components of $F_{i-1} \cap W_i$. Note that $F_{i-1} \backslash W_i$ is the complement of $\bigcup_{j=1}^{i} W_j$, which is connected (as one can easily verify) and becomes the outer face F_i of Γ_i. In this way, we construct $\Gamma_1, \dots, \Gamma_m$ in order, and it is clear that $\Gamma_m = \Gamma$. The linear upper bound for $|\Gamma|$ follows from the following technical result.

▶ **Lemma 16.** $|\Gamma_i| - |\Gamma_{i-1}| = O(1)$ *for* $i \in \{1, \dots, m\}$. *In particular,* $|\Gamma| = O(m)$.

Proof sketch. We denote by ∂W_i the boundary of the wedge W_i, which consists of two rays (emanating from a point) contained in a_i^* and b_i^* respectively. We observe that, to prove $|\Gamma_i| - |\Gamma_{i-1}| = O(1)$, it suffices to show the number of the connected components of $\partial W_i \cap F_{i-1}$ is constant. This can be further reduced to considering one ray of ∂W_i (say the ray r contained in a_i^*). We notice that $r \cap F_{i-1} = r \backslash \bigcup_{j=1}^{i-1} (r \cap W_j)$, and each $r \cap W_j$ is a connected portion of r. Let l_i be the line through a_i, b_i and l_i' be the line through a_i that is perpendicular to l_i, then l_i^* is the initial point of r and $(l_i')^*$ is a point on a_i^*. The crucial observation here is that each $r \cap W_j$ for $j \in \{1, \dots, i-1\}$ satisfies at least one of the four conditions: **(i)** $r \cap W_j$ is empty; **(ii)** $r \cap W_j$ contains the initial point of r; **(iii)** $r \cap W_j$ contains the infinite end of r; **(iv)** $r \cap W_j$ contains the point $(l_i')^*$. The proof of this observation requires us to carefully analyze various cases using the properties of duality and the problem itself. Basically, we consider three cases: (1) $a_j, b_j \in l_i^\uparrow$; (2) $a_j, b_j \in l_i^\downarrow$; (3) one of a_j, b_j is strictly above l_i while the other is strictly below l_i. The first two cases are not very difficult, and are analyzed using duality and some geometry. The last case is the most subtle one. To handle it requires a careful use of the facts that ϕ_i, ϕ_j do not cross (i.e., the segments $[a_i, b_i]$ and $[a_j, b_j]$ do not cross) and ϕ_j is shorter than ϕ_i (because $j < i$ and $\phi_1, \dots, \phi_m$ are sorted in increasing order of their lengths), as well as some properties of duality. We omit the detailed analysis in this sketch. Once the observation is proved, it follows readily that $\bigcup_{j=1}^{i-1}(r \cap W_j)$ has at most three connected components and $r \cap F_{i-1}$ has at most two. As such, $|\Gamma_i| - |\Gamma_{i-1}| = O(1)$. A complete proof can be found in [12]. ◀

With the above result in hand, we can build an optimal point-location data structure for Γ using $O(m)$ space with $O(\log m)$ query time to solve the RCP problem. Since $m = O(n)$, we obtain an $\mathcal{H}$-RCP data structure using $O(n)$ space and $O(\log n)$ query time in worst-case.

Next, we analyze the average-case bounds of the above data structure. In fact, it suffices to bound the expected number of the candidate pairs. Similarly to the quadrant case, we can prove a poly-logarithmic bound.

▶ **Lemma 17.** *For $S \propto R^n$ where R is an axis-parallel rectangle, $\mathbb{E}[|\Phi(S, \mathcal{H})|] = O(\log^2 n)$.*

Now we are able to conclude the following.

▶ **Theorem 18.** *There exists an $\mathcal{H}$-RCP data structure $\mathcal{E}$ such that*

- *For any $S \subseteq \mathbb{R}^2$ of size n, $\mathsf{Space}(\mathcal{E}(S)) = O(n)$ and $\mathsf{Qtime}(\mathcal{E}(S)) = O(\log n)$.*
- *For a random $S \propto R^n$ where R is the unit square or more generally an arbitrary axis-parallel rectangle, $\mathbb{E}[\mathsf{Space}(\mathcal{E}(S))] = O(\log^2 n)$ and $\mathbb{E}[\mathsf{Qtime}(\mathcal{E}(S))] = O(\log \log n)$.*

Our data structure can be built in worst-case $O(n \log^2 n)$ time. We only give the high-level idea here, and the details can be found in [12]. We first observe that if the candidate pairs $\phi_1, \dots, \phi_m$ are already given, then the subdivision Γ can be constructed in $O(m \log m)$ time. The idea is to begin with Γ_0 and iteratively construct Γ_i from Γ_{i-1} by "inserting" the wedge W_i dual to ϕ_i. Each Γ_i can be constructed in *amortized* $O(\log m)$ time (from Γ_{i-1}) by using a (balanced) BST to maintain the outer face and properly exploiting the behavior of W_i observed in Lemma 16. It follows that Γ can be constructed in $O(m \log m)$ time. Now consider the general case in which $\phi_1, \dots, \phi_m$ are not given. We use an approach in [1] to compute in $O(n \log^2 n)$ time a set Ψ of $O(n \log n)$ point-pairs in S such that $\Phi(S, \mathcal{H}^{\uparrow}) \subseteq \Psi$. By considering the pairs in Ψ in increasing order of their lengths, we can efficiently verify whether each pair is a candidate pair or not, and update the subdivision (using the method above) whenever a candidate pair is recognized. The overall process takes $O(n \log^2 n)$ time.

6 Conclusion and future work

We revisited the range closest-pair (RCP) problem, which aims to preprocess a set S of points in $\mathbb{R}^2$ into a data structure such that for any query range X, the closest-pair in $S \cap X$ can be reported efficiently. We proposed new RCP data structures for various query types (including quadrants, strips, rectangles, and halfplanes). Both worst-case and average-case analyses were applied, resulting in new bounds for the RCP problem (see Table 1).

We now list some open questions for future study. First, as mentioned in Section 1.1, the preprocessing for our orthogonal RCP data structures remains open. It is not clear how to build these data structures in sub-quadratic time. Besides, the RCP problem for other query types is also open. One important example is the disk query, which is usually much harder than the rectangle query and halfplane query in traditional range search. For an easier version, we can focus on the case where the query disks have a fixed radius, or equivalently, the query ranges are *translates* of a fixed disk. Along this direction, one can also consider translation queries of some shape other than a disk. For instance, if the query ranges are translates of a fixed rectangle, can we have more efficient data structures than our rectangle RCP data structure in Section 4? Finally, the RCP problem in higher dimensions is quite open. To our best knowledge, the only known result for this is a simple data structure given in [6] constructed by explicitly storing all the candidate pairs, which only has guaranteed average-case performance.

References

1 Mohammad Ali Abam, Paz Carmi, Mohammad Farshi, and Michiel Smid. On the power of the semi-separated pair decomposition. In *Workshop on Algorithms and Data Structures*, pages 1–12. Springer, 2009.

2 Pankaj K Agarwal, Jeff Erickson, et al. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.

3 M. de Berg, M. van Kreveld, M. Overmars, and O. C. Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 2000.

4 Herbert Edelsbrunner, Leonidas J Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.

5 Prosenjit Gupta. Range-aggregate query problems involving geometric aggregation operations. *Nordic journal of Computing*, 13(4):294–308, 2006.

6 Prosenjit Gupta, Ravi Janardan, Yokesh Kumar, and Michiel Smid. Data structures for range-aggregate extent queries. *Computational Geometry: Theory and Applications*, 2(47):329–347, 2014.

7 Saladi Rahul and Yufei Tao. On top-k range reporting in 2D space. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 265–275. ACM, 2015.

8 Neil Sarnak and Robert E Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29(7):669–679, 1986.

9 Jing Shan, Donghui Zhang, and Betty Salzberg. On spatial-range closest-pair query. In *International Symposium on Spatial and Temporal Databases*, pages 252–269. Springer, 2003.

10 R Sharathkumar and Prosenjit Gupta. Range-aggregate proximity queries. *IIIT Hyderabad, Telangana*, 500032, 2007.

11 Michiel Smid. *Closest point problems in computational geometry.* Citeseer, 1995.

12 Jie Xue, Yuan Li, Saladi Rahul, and Ravi Janardan. New bounds for range closest-pair problems. *arXiv preprint arXiv:1712.09749*, 2017.