Two-Dimensional Range Diameter Queries

Pooya Davoodi[1], Michiel Smid[2], and Freek van Walderveen[1]

[1] MADALGO*, Department of Computer Science, Aarhus University, Denmark.
[2] School of Computer Science, Carleton University, Ottawa, Canada.

Abstract. Given a set of n points in the plane, range diameter queries ask for the furthest pair of points in a given axis-parallel rectangular range. We provide evidence for the hardness of designing space-efficient data structures that support range diameter queries by giving a reduction from the set intersection problem. The difficulty of the latter problem is widely acknowledged and is conjectured to require nearly quadratic space in order to obtain constant query time, which is matched by known data structures for both problems, up to polylogarithmic factors. We strengthen the evidence by giving a lower bound for an important sub-problem arising in solutions to the range diameter problem: computing the diameter of two convex polygons, that are separated by a vertical line and are preprocessed independently, requires almost linear time in the number of vertices of the smaller polygon, no matter how much space is used. We also show that range diameter queries can be answered much more efficiently for the case of points in convex position by describing a data structure of size $O(n \log n)$ that supports queries in $O(\log n)$ time.

1 Introduction

Measuring the extent of a set of points in the plane finds a variety of applications in clustering, collision detection, shape-fitting, data mining, etc. [2, 6, 12, 16, 17]. For example in clustering algorithms, the *diameter* of a point set, the largest distance between any pair of points, is used as a measure of the spread of points in a cluster [4, 7]. On the other hand, computing aggregate functions on a subset of points contained in a query range, that is *range aggregate queries*, is interesting from the perspective of computational geometry and database applications, and has attracted the attention of researchers from both research communities [1, 14, 15, 18, 23]. Examples of such range aggregate queries include reporting the closest pair, furthest pair (diameter), width, and the radius of the minimum enclosing disk of points contained in a rectangular range query [14, 15, 20]. In this paper, we primarily study range diameter queries asking for the furthest pair of points within a given rectangular range (see Figure 1). More formally, we study the problem of preprocessing a set of points from $\mathbb{R}^2$ into a data structure such that given an orthogonal range query, we can find the furthest pair of points within the query range.

* Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation.

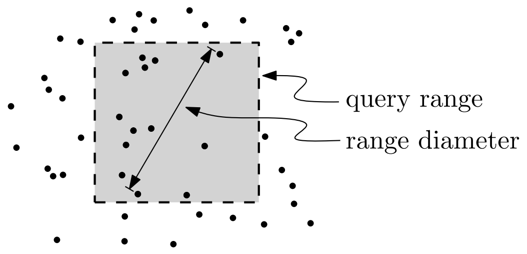

Fig. 1. The answer to a range diameter query is a pair of points that are furthest away in a given orthogonal range.

Unlike many other range queries (such as range counting, range reporting, and range maximum), the diameter of a set of points is "holistic" in the sense that it cannot be computed by dividing the point set into subsets, computing the diameter for each subset, and then aggregating the diameters [14]. In fact, the main difficulty in such a divide and conquer strategy is to combine the partial results, that is, to find the diameter of two point sets. Storing the answers to all such subproblems would yield constant query time per subproblem, but uses at least quadratic space in total. We can reduce the amount of space as follows. Let S_1 and S_2 be two disjoint subsets of the input point set, where $|S_1| \leq |S_2|$. We can find the largest distance between each point in S_1 and all points in S_2 in $O(|S_1| \log |S_2|)$ query time by using a furthest point Voronoi diagram of S_2 to find the furthest point in S_2 for each of the points in S_1 [15]. This data structure requires only linear space per subset, but in case both sets are big a query will take $\Omega(n \log n)$ time per subproblem. The currently best known solution to the range diameter problem stores the diameter for pairs of large subsets, while using Voronoi diagrams for finding the diameter between other pairs of subsets [15]. This results in a trade-off with $O((n+(n/k)^2) \log^2 n)$ space and $O(k \log^5 n)$ query time, for a parameter k where $1 \leq k \leq n$. Therefore, the holistic property of the diameter raises the subject of space-efficiency of data structures supporting range diameter queries, which is the subject of this paper.

Our Results. In Section 2, we investigate the hardness of range diameter queries by observing that the disjointness of two sets among a collection of sets, namely a *set intersection* query, can be verified using a range diameter query over a suitable set of points (reducing set intersection queries to other problems has been previously considered to study the hardness of approximate distance oracles [10, 22]). Obtaining space-efficient data structures that support set intersection queries is known to be hard. A folklore conjecture states that, for m sets of cardinality polylogarithmic in m, answering a set intersection query in $O(1)$ time requires an $\tilde{\Omega}(m^2)$-space data structure, and answering a set intersection query in polylogarithmic time (but asymptotically smaller than the maximum cardinality of the sets) requires $\tilde{\Omega}(m^{2-\varepsilon})$ space [22]. Our reduction in Section 2.1 implies that range diameter queries are as hard as set intersection queries in the real RAM without the floor function. We conjecture that $\tilde{\Omega}((n/k)^2)$ space is required to answer range diameter queries in $\tilde{O}(k)$ time. We use tilde notation to hide polylogarithmic factors.

As previously mentioned, in answering range diameter queries using a divide and conquer strategy, computing the diameter of two disjoint point sets arises as a subproblem. In 1985, Edelsbrunner [11, Section 4] considered the related problem of computing the diameter of two convex polygons that are separated by a vertical line and are preprocessed independently. He showed that if each polygon is represented as a list of vertices, then linear query time in the number of vertices of the larger polygon is required. In Section 2.2, we show a cell probe lower bound that is almost linear in the number of vertices of the smaller polygon, no matter how much space and preprocessing time is spent. This lower bound not only addresses an open problem mentioned by Edelsbrunner [11], but also may be a step forward in proving our lower bound conjecture in Section 2.1 for range diameter queries.

Our reduction from set intersection queries to range diameter queries is on a hard instance of the problem consisting of a set of points placed on a linear number of concentric circles, thus the set is not in convex position (see Fig. 2). In Section 3, we show that range diameter queries can be answered much more efficiently in case points are in convex position. In particular, we describe a data structure that stores a set of n points in convex position using $O(n \log n)$ space such that range diameter queries can be answered in $O(\log n)$ time.

The *width* of a set P of points in the plane is the minimum distance between any two parallel lines ℓ_1 and ℓ_2 enclosing P, that is, P and ℓ_1 are on the same side of ℓ_2, and P and ℓ_2 are on the same side of ℓ_1. In Section 3, we also sketch how to adapt and extend the range diameter data structure to create a data structure with the same space and query bounds for finding the width within any given query range of a set of points in convex position.

2 Reduction from Set Intersection Queries

In this section, we provide support for the hardness of range diameter queries by presenting a reduction from the set intersection problem. This reduction implies a lower bound for range diameter queries based on a generalization of a folklore conjecture for the set intersection problem. We finish this section by proving a lower bound for determining the diameter of two convex polygons that are separated by a vertical line. This problem usually arises as a subproblem in answering range diameter queries. We conclude that this lower bound may be a step forward in proving our conjecture for range diameter queries.

The set intersection problem is to preprocess m sets $S_1, S_2, \ldots, S_m$ of positive real numbers into a data structure that supports set intersection queries asking whether the sets S_i and S_j are disjoint, for given query indices i and j. Let N be the total number of elements in the sets, that is $N = \sum_{i=1}^{m} |S_i|$.

Theorem 1. *Given a data structure of size $s(n)$ that supports range diameter queries in $t(n)$ time on any point set of size n in the plane, we can build a data*

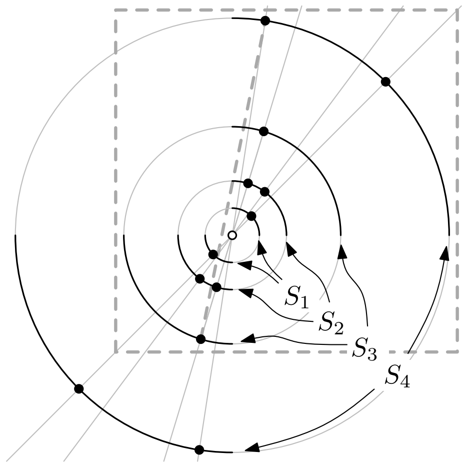

Fig. 2. Example of the reduction in Theorem 1. Each element e is shown as a line $y = ex$, with points at the intersections with circle c_i if and only if $e \in S_i$. The dashed query rectangle is used to report whether S_3 and S_4 intersect. As the diameter (indicated by the dashed line) is less than $r_3 + r_4$, $S_3 \cap S_4 = \emptyset$. Note that $S_2 \cap S_3 \neq \emptyset$.

structure of size $s(2N)$ supporting set intersection queries in $t(2N)$ time, for input sets containing N elements in total.

Proof. Let $S_1, S_2, \ldots, S_m$ be a collection of sets, and recall that $N = \sum_{i=1}^{m} |S_i|$. We transform the sets into a point set of size $2N$ in the plane, and we show that each set intersection query can be answered using a range diameter query.

Let $r_i = 2^{i-1}$ for $i = 1, \ldots, m$. We map each $e \in S_i$ to two points positioned on the first and third quadrant of the circle c_i with radius r_i centered on $(0, 0)$. The positions are determined by the two intersection points of the line $y = ex$ with c_i (see Fig. 2). Notice that for $e \in S_i$ and $e \in S_j$, the distance between the corresponding points on the first quadrant of c_i and the third quadrant of c_j is $r_i + r_j$. By the triangle inequality, for $e \in S_i$ and $e' \in S_j$, where $e \neq e'$, the distance between the point corresponding to e on the first quadrant of c_i and the point corresponding to e' on the third quadrant of c_j is less than $r_i + r_j$. Therefore, to verify the disjointness of S_i and S_j, we ask a range diameter query over the rectangle with bottom-left point $(-r_i, -r_i)$ and top-right point (r_j, r_j). If the diameter of the points within this rectangle is $r_i + r_j$, then $S_i \cap S_j \neq \emptyset$, and if the diameter is smaller than $r_i + r_j$, then $S_i \cap S_j = \emptyset$ (they are disjoint). □

2.1 Conditional Lower Bound

We can naively solve the set intersection problem with $O(1)$ query time using $O(m^2)$ space by tabulating the answer of all queries. Cohen and Porat [10] presented a data structure of size $O((N/k)^2)$ that supports set intersection queries in $O(k \log N)$ time, for a parameter k where $1 \leq k \leq N$. They tabulate the answers of queries where each query set has at least k elements. To verify the disjointness of S_i and S_j, if w.l.o.g. S_i has less than k elements then they search for each element of S_i in S_j in logarithmic time (the query time can be improved to $O(k)$ using linear perfect hashing in the word RAM [9, 10]). Note that this same approach was used in [15] to obtain the currently best known data structure for range diameter queries that was mentioned in Section 1. Pătraşcu and Roditty [22] mentioned a folklore conjecture stating that $\tilde{\Omega}(m^2)$ space is required to support set intersection queries in $O(1)$ time, for a universe of size polylogarithmic in m. They also strengthened the conjecture to polylogarithmic query time (but asymptotically smaller than the maximum cardinality of the sets) and a space lower bound of $\Omega(m^{2-\varepsilon})$ in the cell probe model. The following is a generalized version of their conjecture, which would imply that the best known upper bound of Cohen and Porat [10] is optimal up to polylogarithmic factors.

Conjecture 1. Given a collection of m sets of N real numbers in total, where the maximum cardinality of the sets is polylogarithmic in m, any real-RAM data structure that supports set intersection queries in $\tilde{O}(k)$ time without using the floor function, requires $\tilde{\Omega}((N/k)^2)$ space, for $1 \leq k \leq N$.

From Theorem 1 and Conjecture 1, we conclude the following.

Theorem 2. *Assuming Conjecture 1, any real-RAM data structure that supports range diameter queries on a set of n points from $\mathbb{R}^2$ in $\tilde{O}(k)$ time without using the floor function, requires $\tilde{\Omega}((n/k)^2)$ space, for $1 \leq k \leq n$.*

Remark. In our reduction in Section 2, we transform a collection of sets into a set of points which have exponentially large coordinates. As a result, lower bounds for the set intersection problem imply lower bounds for range diameter queries, only in a computational model where working with unbounded numbers is allowed (like real RAM). An interesting open problem is giving a transformation algorithm in the word RAM, implying that cell probe lower bounds for the set intersection problem also apply to range diameter queries.

2.2 Diameter of Two Convex Polygons

We prove a lower bound for the problem of representing two convex polygons P and Q in the plane, that are separated by a vertical line (the preprocessing of each polygon into its representation is oblivious to the other polygon), such that we can determine the furthest pair of points in $P \cup Q$ using the two representations. This problem often arises as a subproblem when answering range diameter queries, in case we divide a query into disjoint subqueries and then combine the answers of the subqueries. Our lower bound essentially shows that it is hard to combine the answers of two subqueries if we do not store any information about both subqueries together. This may be a step forward in proving Theorem 2 unconditionally.

In 1985 Edelsbrunner [11, Theorem 4.1] showed that if we represent a polygon as a list of vertices, then $\Omega(|P|+|Q|)$ is a lower bound on the worst-case time complexity of determining the diameter of $P \cup Q$. He raised the question of determining the complexity of the problem for other representations of polygons. We address this open problem by proving a lower bound of $\tilde{\Omega}(\min\{|P|,|Q|\})$ for any representation of the polygons, derived by a reduction from the asymmetric (lopsided) version of the *set disjointness* problem in communication complexity. Our reduction is similar to the reduction from set disjointness to computing the diameter of a planar point set [21]. The latter reduction implies a lower bound of $\Omega(n \log n)$ time to compute the diameter of a planar point set in the algebraic computation tree model [21].

The asymmetric set disjointness problem in communication complexity is for Alice and Bob to verify the disjointness of sets A and B, after Alice receives A and Bob receives B, where $A, B \subseteq [n]$, and $|A| < |B| < n/2$. It is known that Alice and Bob need to communicate $\Omega(|A|)$ bits to determine whether the sets are disjoint [19]. This lower bound implies that for any representation of two given sets A and B, $\tilde{\Omega}(|A|)$ time is required to verify the disjointness of A and B in the cell probe model. Now we use the latter lower bound to prove the following.

Theorem 3. *For any independent representation of two convex polygons P and Q that are vertically separated, finding the furthest pair of points in $P \cup Q$ requires $\tilde{\Omega}(\min\{|P|,|Q|\})$ time, in the cell probe model.*

Proof. As previously mentioned, $\tilde{\Omega}(|A|)$ time is required to verify the disjointness of A and B, for any representation of given sets A and B, where $A, B \subseteq [n]$, and $|A| < |B| < n/2$. We construct two vertically-separated point sets P and Q corresponding to A and B respectively, and then we show that the disjointness of A and B can be verified by finding the diameter of $P \cup Q$.

We map each element $e \in A$ to a point positioned on the intersection point of the line $y = ex$ with the first quadrant of the unit circle. Similarly, we map each element $e \in B$ to a point positioned on the intersection point of the line $y = ex$ with the third quadrant of the unit circle. Hence, P has $|A|$ points and Q has $|B|$ points. It is clear that there exists an element e belonging to both A and B if and only if there exist a point $p \in P$ and a point $q \in Q$ such that the distance between p and q is 2. We compute the diameter of $P \cup Q$. If the diameter is 2 then there is a common element in A and B, and otherwise (the diameter is less than 2) A and B are disjoint. □

3 Points in Convex Position

As it appears unlikely that we can get polylogarithmic query time when using $O(n^{2-\varepsilon})$ space for range diameter queries on sets of n points, we consider in this section the case of sets of points in convex position. For this case we describe data structures with polylogarithmic query time using near-linear space. The precise bounds on space and query time depend on the choice of underlying

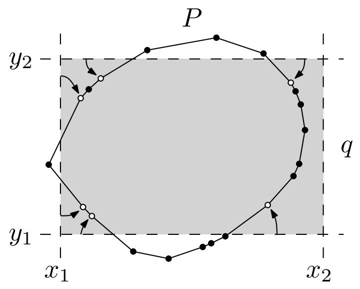

Fig. 3. Range diameter query on the vertices of a convex polygon. In this example, the query range covers three disjoint sections. Predecessor (and successor) queries are indicated by arrows. The white vertices are within the query range and determine the sections of $q \cap P$.

data structures. We also describe a data structure for the range width problem with the same bounds.

Let a *section* of a convex polygon be a sequence of consecutive vertices of that polygon. We first describe how to find the (at most four) disjoint sections containing all the vertices covered by a query range (see Fig. 3). Second, we show how to solve the problem of finding the furthest pair of points between two given sections. For the description and analysis of our approach to answering such *section–section queries* we review a characteristic of convex polygons called *modality* and a derivative thereof that we use in our analysis. We show that section–section queries can be answered efficiently using two data structures: one storing the distances between a set of $O(n)$ selected point pairs explicitly, and one for answering *point–section queries*, a special case of section–section queries in which one section contains only one point.

3.1 Reduction to Section–Section Queries

The following lemma is easy to prove using predecessor data structures (Fig. 3).

Lemma 1. *A convex polygon $P = (p_1, p_2, \dots, p_n)$ can be preprocessed to obtain a linear-space data structure for finding the at most four sections of P intersecting a given query range $q = [x_1 : x_2] \times [y_1 : y_2]$ in $O(\log n)$ time.*

Let S_a be the sequence of points in the ath section of $q \cap P$. The diameter of the points in $q \cap P$ can be found by taking the maximum of all point pair distances in all pairs of sections: $\max_{a,b}\{\max_{p \in S_a, q \in S_b}\{d(p,q)\}\}$, where $d(p,q)$ is the Euclidean distance between points p and q. We can therefore focus on determining the maximum point pair distance between two (possibly equal) sections S_a and S_b.

3.2 Section–Section Queries

The main complicating factor in designing algorithms for convex polygons appears to be the fact that for a given vertex of a convex polygon the sequence of distances to the other vertices in order around the polygon may contain more than one local maximum. The maximum number of local maxima in the distance sequence of any vertex of a polygon $P = (p_1, p_2, \ldots, p_n)$ is called the *modality* of P. More formally, take $p_0 := p_n$, $p_{n+1} := p_1$, and let $M_i := \{1 \leq j \leq n : d(p_i, p_{j-1}) < d(p_i, p_j) \text{ and } d(p_i, p_{j+1}) < d(p_i, p_j)\}$ be the set of local maxima for vertex p_i, then the modality of P is $\max_{i=1}^{n} |M_i|$. Avis et al. [5] show there exist polygons for which the *total modality* $\sum_{i=1}^{n} |M_i| = \Theta(n^2)$, so we cannot hope for a space-efficient data structure that stores the local maxima for all vertices.

Reciprocal modality. The main observation on which our solution to the section–section problem is based is that, given two sections S_a and S_b, in case we know that a point $p \in S_a$ is *not* a local maximum in the distance sequence of point $q \in S_b$, the distance $d(p, q)$ cannot be the maximum distance between S_a and S_b unless p is either the first or the last point in S_a, because otherwise both neighbours of p in P are in the query range and the distance from q to one of those neighbours is larger than $d(p, q)$. This observation implies that for any pair of sections S_a and S_b, the furthest point pair is either a *pair of reciprocal local maxima* for which both points are local maxima in each other's distance sequence, or one of the points is the first or the last in its section. The largest distance between S_a and S_b is therefore equal to the maximum of

1. the distance of the furthest pair of reciprocal local maxima of which one point is in S_a and the other point is in S_b, and
2. the distance from the first/last point in S_b (S_a) to the furthest point in S_a (S_b).

The furthest pair of reciprocal local maxima can be found in the following way. Let Q be a set containing a point (i, j) for each pair of reciprocal local maxima (p_i, p_j), that is, $Q := \{(i, j) \mid j \in M_i \text{ and } i \in M_j\}$. To each point (i, j) we assign a weight of $d(p_i, p_j)$. We create a two-dimensional range-maximum data structure over Q to be able to efficiently find the point with maximum weight within a given orthogonal query range, if it exists. Let f_a be the index of the first point in S_a and l_a the index of the last point in S_a. Note that the points in Q inside a range $[f_a : l_a] \times [f_b : l_b]$ (for $f_a \leq l_a$ and $f_b \leq l_b$) represent the pairs of reciprocal local maxima between S_a and S_b, so the point in this range with the highest weight corresponds to the furthest pair of reciprocal local maxima between S_a and S_b. In case $f_a > l_a$ or $f_b > l_b$, we take the maximum of two or four queries covering the whole query range.

To bound the amount of space necessary for the 2D range-maximum data structure, we show that the *reciprocal modality* $|Q| = O(n)$.

Lemma 2. *The reciprocal modality of the vertex set of any convex polygon $P = (p_1, p_2, \ldots, p_n)$ is $O(n)$.*

Proof. We show that any pair p, q of reciprocal local maxima is also an *antipodal pair*, that is, there exist parallel lines through p and q such that all other points of P lie between these lines. As the number of antipodal pairs is linear [21,

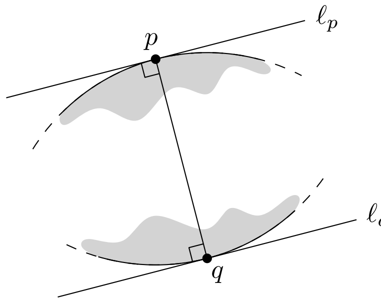

Fig. 4. If p and q are reciprocal local maxima, they also form an antipodal pair.

Section 4.2.3], the number of reciprocal local maxima, and hence the reciprocal modality, is $O(n)$.

Consider a pair p, q of reciprocal local maxima and draw parallel lines ℓ_p and ℓ_q through p and q, orthogonal to a line through both vertices (see Figure 4). As the vertices neighbouring p and q in the polygon must be closer to q and p, respectively, they must also be between ℓ_p and ℓ_q. By convexity, all points of P must be between ℓ_p and ℓ_q, so p and q form an antipodal pair. □

Point–section queries. Finally, we need a data structure for answering point–section queries. For this problem we can use a data structure of Aronov et al. [3] that uses $O(n \log^3 n)$ space to answer queries of the following type in $O(\log n)$ time: For a point q in the plane and a section of the polygon, find the point in this section furthest away from q. In our case q is always a vertex of the polygon, allowing us to design a data structure that uses less space.

Lemma 3. *The vertices of a convex polygon $P = (p_1, p_2, \ldots, p_n)$ can be preprocessed into a data structure of size $O(n \log n)$ such that queries of the following type can be answered in $O(\log n)$ time: given three indices i, j and k such that $j \leq k$, find the furthest point from p_i in the range $(p_j, p_{j+1}, \ldots, p_k)$.*

Proof. Our structure is a two-level structure, where the first level consists of a balanced binary search tree on the indices of the vertices of P with every vertex represented by a leaf. For every node v of the tree, let $P(v)$ denote the canonical set of v, that is, the set of vertices in the subtree rooted at v. Let $S(v, x)$ be the set of vertices $z \in P$ for which $x \in P(v)$ is the furthest vertex among all vertices in $P(v)$, that is, $S(v, x) := \{z \in P \mid x = \arg\max_{y \in P(v)} d(z, y)\}$ (see also Fig. 5). Because each $S(v, x)$ forms a consecutive subsequence of P we can store a list $L(v)$ of the indices of the first vertex in $S(v, x)$ for each x as a second-level data structure for each node v. This requires $O(n \log n)$ space in total.

Queries for indices i, j and k can be answered as follows. Find the $O(\log n)$ nodes whose canonical sets together cover $(p_j, p_{j+1}, \ldots, p_k)$, but whose parents contain vertices outside the range. For each node v found in this way, do a binary search in $L(v)$ for i to obtain the furthest point from p_i among $P(v)$. By taking

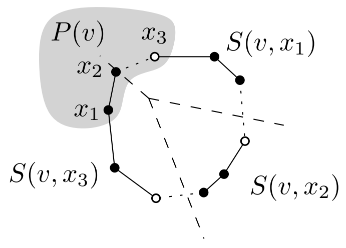

Fig. 5. Example subdivision of P into sets of points that have the same vertex in $P(v)$ as their furthest point, where the dashed diagram shows the regions of the plane with the same furthest point among the vertices of $P(v)$. The indices of the white vertices are saved in $L(v)$.

the maximum distance obtained from all these nodes we get the answer to the query in $O(\log^2 n)$ time. Since we search for the same value i in all $O(\log n)$ lists, we can apply fractional cascading to obtain $O(\log n)$ query time. □

Alternatively, we can use a linear-space data structure with higher query time.

Lemma 4. *The vertices of a convex polygon $P = (p_1, p_2, \ldots, p_n)$ can be preprocessed into a data structure of size $O(n)$ such that queries of the following type can be answered in $O(\log^2 n)$ time: given three indices i, j and k such that $j \leq k$, find the furthest point from p_i in the range $(p_j, p_{j+1}, \ldots, p_k)$.*

Proof. Build a range tree as for Lemma 3, but without the second level. Instead, we add a secondary key to each node v to support searching for the furthest point from query point p_i inside $P(v)$. The secondary key represents the range of vertices of P whose furthest point is in the left subtree of v.

For answering a query, we again find the $O(\log n)$ nodes whose canonical sets together cover $(p_j, p_{j+1}, \ldots, p_k)$. For each such node v, find the furthest point from p_i among $P(v)$ using the secondary keys. By taking the maximum distance obtained from all these nodes we can answer the query in $O(\log^2 n)$ time. □

Now that we have described all necessary components, we can put them together to obtain the main result of this section.

Theorem 4. *Given a convex point set, we can construct (1) in $O(n \log n)$ time an $O(n \log n)$-space data structure that answers range diameter queries in $O(\log n)$ time, or (2) an $O(n \log^{\varepsilon} n)$-space data structure with $O(\log^2 n)$ query time in the word RAM model.*

Proof. The predecessor structures of Lemma 1 use $O(n)$ space and take $O(\log n)$ time per query, and can be constructed in $O(n \log n)$ time. We construct set Q, containing pairs of indices of reciprocal local maxima, in $O(n)$ time by inspecting all antipodal pairs, which can be enumerated in linear time [21]. For result (1) we store Q in the 2D range maximum data structure of Gabow et al. [13], which answers queries in $O(\log n)$ time using $O(n \log n)$ space, and can be constructed in $O(n \log n)$ time. For result (2) we use a data structure of Chan et al. [8] that has $O(\log \log n)$ query time using $O(n \log^{\varepsilon} n)$ space in the word RAM model.

Point–section queries are answered by constructing in $O(n \log n)$ time, (1) the data structure of Lemma 3, using $O(n \log n)$ space and $O(\log n)$ query time, or (2) the data structure of Lemma 4, using $O(n)$ space and $O(\log^2 n)$ query time.

As described, range maximum queries can be answered using a constant number of queries on these data structures. □

3.3 Range Width

Recall that the width of a set of points in the plane is the smallest distance between any two parallel lines enclosing all points. For sets of points in convex position, it is easy to show that these lines are always incident to antipodal pairs, suggesting that we can follow a similar approach to answering range width queries in convex polygons as we did for answering range diameter queries. We now sketch how to build on the techniques described above to develop such a data structure.

We follow the same structure and first split the query into sections. For each pair of sections, we find the closest antipodal pair using a 2D range minimum data structure on the indices of the vertices forming antipodal pairs. We then only need to show how to answer point–section queries. The main difference between the two problems is that for range width, we cannot answer point–section queries in isolation: a valid pair of points and incident parallel lines may exist for a given point and section, while no parallel lines through these points exist that enclose all other points. Therefore, we shrink the section to only include points that allow valid parallel lines. As a preprocessing step, we use the rotating calipers algorithm to find for each edge of the polygon the one or two vertices that are furthest away from the line through that edge [21]. For a point–section query on indices i, j, and k, we first find the vertices $p_{j'}$ and $p_{k'}$ opposite edges (p_i, p_{i+1}) and (p_{i-1}, p_i) (where $p_0 := p_n$ and $p_{n+1} := p_1$), taking the vertices that are furthest apart in case of parallel edges. Then, we search for the closest point to p_i within the range $[j : k] \cap [j' : k']$.

Acknowledgements. We would like to thank Elad Verbin for introducing the set intersection problem, and Gerth Stølting Brodal, Jakob Truelsen, Konstantinos Tsakalidis, and Qin Zhang for informative discussions.

References

1. P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. AMS, 1999.
2. P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *Journal of the ACM*, 51(4):606–635, 2004.

3. B. Aronov, P. Bose, E. Demaine, J. I. Joachim Gudmundsson, S. Langerman, and M. Smid. Data structures for halfplane proximity queries and incremental voronoi diagrams. In *Proc. 7th LATIN*, pages 80–92, 2006.
4. D. Avis. Diameter partitioning. *Discrete & Computational Geometry*, 1(1):265–276, 1986.
5. D. Avis, G. T. Toussaint, and B. K. Bhattacharya. On the multimodality of distances in convex polygons. *Computers & Mathematics with Applications*, 8(2):153–156, 1982.
6. G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38(1):91–109, 2001.
7. V. Capoyleas, G. Rote, and G. J. Woeginger. Geometric clusterings. *Journal of Algorithms*, 12(2):341–356, 1991.
8. T. M. Chan, K. G. Larsen, and M. Pǎtraşcu. Orthogonal range searching on the RAM, revisited. In *Proc. 27th Symp. on Comp. Geometry*, pages 1–10, 2011.
9. H. Cohen and E. Porat. Fast set intersection and two-patterns matching. *Theoretical Computer Science*, 411(40-42):3795–3800, 2010.
10. H. Cohen and E. Porat. On the hardness of distance oracle for sparse graph. *The Computing Research Repository (arXiv)*, abs/1006.1117, 2010.
11. H. Edelsbrunner. Computing the extreme distances between two convex polygons. *Journal of Algorithms*, 6(2):213–224, 1985.
12. C. Faloutsos and K.-I. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 163–174, 1995.
13. H. N. Gabow, J. L. Bentley, and R. E. Tarjan. Scaling and related techniques for geometry problems. In *Proc. 16th STOC*, pages 135–143, 1984.
14. P. Gupta. Algorithms for range-aggregate query problems involving geometric aggregation operations. In *Proc. 16th ISAAC*, pages 892–901, 2005.
15. P. Gupta, R. Janardan, Y. Kumar, and M. H. M. Smid. Data structures for range-aggregate extent queries. In *Proc. 20th CCCG*, pages 7–10, 2008.
16. S. Har-Peled. A practical approach for computing the diameter of a point set. In *Proc. 17th Symp. on Comp. Geometry*, pages 177–186. ACM, 2001.
17. S. Har-Peled and Y. Wang. Shape fitting with outliers. *SIAM Journal on Computing*, 33(2):269–285, 2004.
18. S. Hong, B. Song, and S. Lee. Efficient execution of range-aggregate queries in data warehouse environments. In *Proc. 20th International Conference on Conceptual Modeling*, pages 299–310, 2001.
19. P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998.
20. Y. Nekrich and M. H. M. Smid. Approximating range-aggregate queries using coresets. In *Proc. 22nd CCCG*, pages 253–256, 2010.
21. F. Preparata and M. Shamos. *Computational geometry: an introduction.* Texts and monographs in computer science. Springer, 1991. Section 4.2.3.
22. M. Pǎtraşcu and L. Roditty. Distance oracles beyond the Thorup-Zwick bound. In *Proc. 51st FOCS*, pages 815–823, 2010.
23. S. Rahul, A. S. Das, K. S. Rajan, and K. Srinathan. Range-aggregate queries involving geometric aggregation operations. In *Proc. 5th WALCOM*, pages 122–133, 2011.